Published online DD MM YYYY

Nucleic Acids Research, YYYY, Vol. xx, No. xx ***1–17***
doi:10.1093/nar/gkn000

Feature selection for kernel methods in systems biology

Céline Brouard [1,*], **Jérôme Mariette** [1,*] **Rémi Flamary** [2] **and Nathalie Vialaneix** [1*]

[1] Université de Toulouse, INRAE, UR MIAT, F-31320, Castanet-Tolosan, France and [2] École Polytechnique, CMAP, F-91120, Palaiseau, France

Received YYYY-MM-DD; Revised YYYY-MM-DD; Accepted YYYY-MM-DD

ABSTRACT

The substantial development of high-throughput bio-technologies has rendered large-scale multi-omics datasets increasingly available. New challenges have emerged to process and integrate this large volume of information, often obtained from widely heterogeneous sources. Kernel methods have proven successful to handle the analysis of different types of datasets obtained on the same individuals. However, they usually suffer from a lack of interpretability since the original description of the individuals is lost due to the kernel embedding. We propose novel feature selection methods that are adapted to the kernel framework and go beyond the well established work in supervised learning by addressing the more difficult tasks of unsupervised learning and kernel output learning. The method is expressed under the form of a non-convex optimization problem with a ℓ_1 penalty, which is solved with a proximal gradient descent approach. It is tested on several systems biology datasets and shows good performances in selecting relevant and less redundant features compared to existing alternatives. It also proved relevant for identifying important governmental measures best explaining the time series of Covid-19 reproducing number evolution during the first months of 2020. The proposed feature selection method is embedded in the R package mixKernel version 0.7, published on CRAN.

INTRODUCTION

The recent development of high-throughput bio-technologies has rendered large-scale multi-omics datasets increasingly available. Biology has now entered the world of "big data", with a pressing need to manage, process and optimize the use of large-scale "-omics" sequencing measurements. In addition to the obvious challenge of having to deal with large volumes of information from widely heterogeneous sources, the underlying biological nature of such data poses the additional issue of high complexity, due to the multiple types of interactions existing within and across multiple levels in living organisms. This triggers the need to consider these multi-layered biological systems as a whole, and to develop accurate and innovative methods to integrate multiple and heterogeneous levels of information collected on the same individuals.

To address this challenge, kernel methods have proven useful and successful (1) because they offer a natural theoretical framework to handle the analysis of different types of data / features observed on the same individuals. They can also address the issue of the huge dimensionality by summarizing each level of information with a similarity matrix between individuals (whose number is generally small enough), providing a solution efficient in terms of computational cost and storage. Relevant kernels can embed expert knowledge and handle the high dimension much better than the Euclidean distance, notorious for behaving poorly when the number of features is large (2). They have been successfully used in computational biology for exploratory (3) or prediction (4, 5, 6) purposes and to integrate datasets in a supervised (7) or unsupervised fashion (8, 9).

However, as stated in (10, 11), kernel methods usually suffer from a lack of interpretability. The initial description of the individuals in terms of features is lost during the kernel embedding, which is known as the pre-image problem (12). In addition, the information of thousands of descriptors is often summarized in a few similarity measures, which can be strongly influenced by a large number of irrelevant descriptors.

To address these issues, feature selection is a widely used strategy: it consists in selecting the most promising features during or prior the analysis. The purpose of this work is to extend feature selection for kernel methods while addressing two rarely met purposes: unsupervised (exploratory) learning and multiple output or non numerical output predictions (which include multiple regression or multi-class supervised classification for instance).

Overview of feature selection.

Feature selection has attracted a large amount of attention during the past years: (13) made a tentative comprehensive overview and classification of the most popular methods for

*To whom correspondence should be addressed. These authors contributed equally to this work. Email: {celine.brouard,jerome.mariette}@inrae.fr

feature selection, accompanied by an open-source repository and benchmark datasets http://featureselection.asu.edu. In addition, implementations of the methods are provided in the Python package **scikit-feature**. These methods are generally organized in three main families: *embedded methods*, where the selection is embedded in a prediction method, *filter methods*, where the selection is made independently of any prediction purpose and, *wrapper methods*, where the selection is made in relation with but not embedded into a prediction method.

As visible through their description, embedded and wrapper methods are thus proposed in the framework of univariate supervised learning, where the selection is combined with the prediction objective to obtain a trade-off between number of selected features and accuracy of the prediction. This is the case, for instance, of the popular lasso approach (14) and its many variants. These embedded methods rely on the ℓ_1 penalty to performs feature selection. Other approaches are performed prior to model estimation but are based on the variable to predict (wrapper methods): **Relief** algorithm (15) computes the quality of attributes according to how well their values distinguish between individuals with different outcomes that they aim at predicting and the Conditional Informative Feature Extraction (**CIFE**, (16)) uses a similar idea, computing an information theory criterion. Also note that embedded methods can be adapted to the semi-supervised framework, exploiting the similarities between samples to propagate labels on unlabeled data (17).

On the contrary, filter methods, like the popular spectral feature selection (**SPEC**) approach (18), can address both unsupervised and supervised problems (and are usually very simple and fast) but they suffer from a major drawback because they perform selection by computing a score independently for each feature: they are thus very sensitive to redundancies in features and not able to account for this redundancy. For the two purposes that are addressed in this article (unsupervised learning and multiple output prediction), only a few alternatives to filter methods already exist.

Feature selection in the unsupervised setting.

For unsupervised learning, the issue of addressing feature selection globally among the set of features is usually handled in two main directions: some approaches, like the Multi-Cluster Feature Selection (**MCFS**, (19)), or the Convex Principal Feature Selection (**CPFS**, (20)), aim at selecting an ensemble of features that recover at best the projection of the data or the data reconstruction on the first axes of the Principal Component Analysis (PCA). Another direction is to incorporate the assumption that the data have an underlined cluster structure and to define a quality criteria that recovers this cluster structure. The first step of **MCFS** is based on this principle but other methods relying on it include, *e.g.*, the Nonnegative Discriminative Feature Selection (**NDFS**, (21)) and the Unsupervised Discriminative Feature Selection (**UDFS**, (22)).

All these methods are better designed than the simpler filter methods to address feature redundancy but they are also based on *a priori* assumptions that the relevant information on the data is contained in the first axis of the PCA or in an (unobserved) cluster structure. Closely related approaches can aim at the reconstruction of a local linear embedding of the data (as in (23)) to provide a more flexible setting. Also, more recent works (24, 25) use an unsupervised embedding approach in which the feature selection is incorporated in neural network auto-encoders.

Contrary to these works, the proposed approach will not aim at performing a data compression that optimizes the ability to reconstruct the original information but rather at preserving at best the original relations between individuals. This is a useful property for exploratory purpose where the user wants to be able to recover clusters, search for atypical individuals, or find a peculiar pattern between individuals, with no *a priori* knowledge.

Feature selection in the multiple output prediction setting.

The issue of feature selection for multiple output prediction has been studied at an even lesser extent than the unsupervised setting. Among the few existing works in this framework, most are not able to deal with non numeric outputs. Apart from the straightforward approach that consists in combining results of independent feature selections obtained for every output, the currently existing approaches are linear methods such as multivariate Gaussian lasso (26, 27), sparse Partial Least Squares (**sPLS**, (28)) or regularized Canonical Correlation Analysis (**CCA**) (29) that all select features in the predictors jointly associated to all outputs (or to a selection of outputs) by the addition of a ℓ_1 penalty to the loss function of the multiple output regression problem.

These approaches improve the interpretability of the results and were proven useful for a variety of applications, including eQTL studies (associations between SNP and gene expression, (30)), metabolomics data (31) or associations between gene expression and phenotypes (29) but they are restricted to linear relations between predictor features and outputs. More recent works go beyond the linear framework, extending feature selection in regression trees, Relief methods (32) or information theory methods (33) for multiple outputs. However, all these approaches are still limited to numeric or multiple class outputs.

Feature selection for kernel methods.

Among efficient methods that have been developed for feature selection, some explicitly use a kernel framework. (34) provide a comprehensive overview of these approaches, explicitly describing their relations and advantages or drawbacks. Most works on feature selection for kernel use a feature based kernel, *i.e.*, one kernel for each feature in the input dataset. The earliest work in this line is the Feature Vector Machine approach (35), where the feature kernel computes a similarity between a given kernel and all the other features. However, since the feature mapping maps each feature into a space having the same dimension than the original number of features, this approach is not adapted to cases where the number of features is larger than the number of observations. (36) proposes a sparse additive model (**SpAM**) approach, which is a linear model using the feature kernels for predictors and a group-lasso based penalty that performs feature selection. One weakness of this approach is that it only targets additive models.

Several feature selection approaches use the Hilbert-Schmidt independence criterion (**HSIC**). **Greedy HSIC** (37) uses the **HSIC** criterion for dependence maximization between the selected features and the outputs with forward / backward selection strategies that allow to compute a kernel based on multiple (selected) features. A remaining drawback of this approach is that forward/backward elimination strategies are heuristics that provide approximated solutions and are often far from the optimum in practice.

The Hilbert-Schmidt Feature Selection (**HSFS** (38)) can be seen as a continuous relaxation of **Greedy HSIC**. It first transforms the original features into a single vector by associating weights to every features (each observation is thus represented by a single value that is the weighted sum of its original feature values) and performs feature selection using a kernel based framework with ℓ_∞ penalty. **HSIC** lasso (34) extends the **Greedy HSIC** approach in order to avoid selecting multiple redundant features. **HSIC** lasso is based on the prediction of an output kernel by a linear model with a sparse penalty. This approach allows to predict any type of outputs and aims at selecting the features that would reproduce at best the relations between the observations, as described by the output kernel. **Block HSIC** lasso (39) is based on the **HSIC** lasso algorithm and uses the **Block HSIC** estimator in order to reduce the memory complexity of **HSIC** lasso.

Finally, the only earlier works that directly perform feature selection in multivariate kernels are (40, 41, 42). These methods train a kernel regression or classification in which the features are selected by weights obtained by minimization of a prediction error penalized with the ℓ_1 norm. Note that those methods are restricted to the supervised framework with numerical output.

Our contribution beyond the state-of-the-art.

In the current paper, these approaches are extended and a feature selection algorithm is proposed, which does not rely on any structural assumption on the data but explicitly takes advantage of the kernel structure in a multivariate manner. It both allows for unsupervised feature selection or for feature selection targeting an arbitrary (kernel based) output. In both situations, the main idea is to simultaneously learn weights, w_j, for each feature j, that correspond to the feature's relevance in the task at hand, as in (40, 41, 42).

The computation of the weights are obtained simultaneously for all features, in order to better account for colinearities or redundancies between features. A ℓ_1 penalty is added in the learning process to obtain a sparse $\mathbf{w}=(w_1,\ldots,w_p)$, which corresponds to a feature selection. This allows to define a flexible framework for feature selection, which is also able to incorporate *a priori* latent cluster structure or known relations between descriptors if needed. Simulations on various types of problems (both supervised and unsupervised) and various types of output datasets (multivariate numeric outputs or time series) show the effectiveness of the proposed methods and its great stability compared to alternative approaches.

In summary, the new approach for kernel method:

- extends the supervised case to the unsupervised setting and to arbitrary kernel output, without the need of structural or *a priori* assumptions on the data;
- accounts for colinearities between features to reduce redundancy in the selection;
- is based on a general framework flexible enough to incorporate *a priori* knowledge on the data if available.

Our feature selection method for kernels is implemented in Python, using numpy https://numpy.org/ and autograd https://github.com/HIPS/autograd and inspired by the PyOptim https://github.com/rflamary/PyOptim library. It is embedded in the `select.features` function of the R package **mixKernel** version 0.7 published on CRAN, using **reticulate** https://rstudio.github.io/reticulate.

MATERIALS AND METHODS

In this section, the proposal is described, by first presenting the common kernel framework before the two versions (unsupervised and kernel output feature selections) are developed. Then, a generic approach is provided, to extend these methods so as they can include *a priori* knowledge (structure between features or clusters, for instance). Finally, the experimental settings for the different simulations performed to evaluate the variants of the method are described.

Description of the kernel framework.

We consider a set of n observations $(\mathbf{x}_i)_{i=1,\ldots,n}$, taking values in $\mathcal{X}=\mathbb{R}^p$ $(\mathbf{x}_i=(x_{ij})_{j=1,\ldots,p})$. The observations are represented through their pairwise similarity using a kernel, K_x, such that $K_x:\mathbb{R}^p\times\mathbb{R}^p\to\mathbb{R}$ is symmetric $(\forall\mathbf{x},\mathbf{x}'\in\mathbb{R}^p,\ K_x(\mathbf{x},\mathbf{x}')=K_x(\mathbf{x}',\mathbf{x}))$ and positive $(\forall N\in\mathbb{N},\ \forall(\alpha_i)_{i=1,\ldots,N}\subset\mathbb{R},\ \forall(\mathbf{x}_i)_{i=1,\ldots,N}\subset\mathbb{R}^p,$ $\sum_{i,i'=1}^N\alpha_i\alpha_{i'}K_x(\mathbf{x}_i,\mathbf{x}_{i'})\geq 0)$. In the sequel, $\mathbf{K}_x$ will denote the symmetric positive semi-definite $(n\times n)$-matrix with entries $(K_x(\mathbf{x}_i,\mathbf{x}_{i'}))_{i,i'=1,\ldots,n}$. The feature map associated with K_x is $\phi:\mathbb{R}^p\to\mathcal{F}_x$, where $\mathcal{F}_x$ is the unique Hilbert space such that

$$\forall\mathbf{x},\mathbf{x}'\in\mathbb{R}^p,\ K_x(\mathbf{x},\mathbf{x}')=\langle\phi(\mathbf{x}),\phi(\mathbf{x}')\rangle_{\mathcal{F}_x}.$$

In some cases (output kernel prediction), a second set of observations $(y_i)_{i=1,\ldots,n}$ is associated to the $(\mathbf{x}_i)_i$ and observed on the same individuals $i\in\{1,\ldots,n\}$. The $(y_i)_i$ take values in an arbitrary space, $\mathcal{Y}$, with no specific requirements apart from the fact that objects in $\mathcal{Y}$ are also well described by another kernel, $K_y:\mathcal{Y}\times\mathcal{Y}\to\mathbb{R}$. This case includes any type of outputs, such as multiple numerical variables (for which the standard Euclidean scalar product can be used as a kernel for instance) or multiple class variables. Similarly to K_x, the feature map associated with K_y is denoted by ψ, the feature space by $\mathcal{F}_y$ and the kernel matrix by $\mathbf{K}_y\in\mathbb{R}^{n\times n}$.

Two distinct problems are then addressed: the first, described in Section "Unsupervised feature selection", relates to the selection of a subset of d features within the p original features in $\mathbb{R}^p$, such that $d\ll p$ and that the selected features limit the information loss. Contrary to most methods, which select the features by maximizing the prediction quality of a given quantity, this selection is done in an unsupervised setting, mostly aiming at preserving the topology structure of

the original kernel K_x (*i.e.*, the relations / similarities between individuals as described by K_x).

As stated in the introduction, this property is useful for exploratory purpose when one wants to recover clusters, find outliers, or design a topology of his/her samples. In addition to that appealing property, using a kernel K_x allows to handle very general similarities between input feature vectors $\mathbf{x}_i$, which are often more adapted than the standard Euclidean scalar product. This is particularly true when the dimension p is large, in which case the Euclidean distance is notoriously non informative (2).

The second issue, described in Section "Kernel output feature selection" extends feature selection to association studies. More precisely, the aim is to obtain a subset of $d \ll p$ features that best explain the $(y_i)_i$ or, rather, that best explain the way these individuals relate to each other as described by K_y. Again, this approach has several advantages: first, it allows to define prediction functions for predicting objects taking values in a very general space $\mathcal{Y}$ (as long as the pre-images of elements in $\mathcal{F}_y$ are easily accessible). In addition, again, it allows to incorporate in the model very general similarities between input and output vectors and these similarities can be more adapted to describe the relations between samples than the standard Euclidean scalar product.

We propose to address both problems by introducing a vector of p weights $\mathbf{w} = (w_j)_{j=1,\dots,p}$, taking values in $\{0,1\}^p$ and such that $w_j = 1$ is equivalent to select feature j. A new kernel, $K_x^{\mathbf{w}}$, with associated kernel matrix $\mathbf{K}_x^{\mathbf{w}}$, can be then defined from K_x by:

$$K_x^{\mathbf{w}}(\mathbf{x}_i, \mathbf{x}_{i'}) := K_x(\mathbf{w} \cdot \mathbf{x}_i, \mathbf{w} \cdot \mathbf{x}_{i'}),$$

in which " $\cdot$ " is the element-wise multiplication: $\mathbf{w} \cdot \mathbf{x} := (w_1 x_1, \dots, w_p x_p)^\top = \text{Diag}(\mathbf{w})\mathbf{x}$. In short, $K_x^{\mathbf{w}}$ is the restriction of K_x to the d features selected through the definition of $\mathbf{w}$. In the following, $\mathbf{D}_{\mathbf{w}} \in \mathbb{R}^{p \times p}$ is used as a shortcut for $\text{Diag}(\mathbf{w})$.

This approach gives a natural way to choose $\mathbf{w}$ by deriving a criterion that is optimized to obtain a trade-off between feature selection and quality of the objective (the preservation of the kernel structure of K_x or the association with the kernel K_y, respectively). However, the optimization problem associated with such an objective is a discrete optimization problem on $\{0,1\}^p$ that is usually hard to solve (the problem would be NP complete with an exhaustive search having to consider all the 2^p possible solutions). We thus consider a continuous relaxation with $w_j \geq 0$ and in which feature selection is conveniently handled using an ℓ_1 penalization that promotes a sparse solution. This approach was first presented in (40) for Support Vector Machine (SVM) with Gaussian kernels and further developed in (41, 42). It is also similar to the approach used in (25) for unsupervised feature selection with auto-encoders.

Notations used thorough the article are described in Table 1

Notation	Explanation
n	number of observations
p	number of features
d	number of selected features
$\mathcal{X} = \mathbb{R}^p$	input space
$\mathcal{Y}$	output space
$K_x : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$	input kernel
$K_y : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$	output kernel
$\mathbf{K}_x \in \mathbb{R}^{n \times n}$	input kernel matrix with entries $(K_x(\mathbf{x}_i, \mathbf{x}_{i'}))_{i,i'=1,\dots,n}$
$\mathbf{K}_y \in \mathbb{R}^{n \times n}$	output kernel matrix with entries $(K_y(y_i, y_{i'}))_{i,i'=1,\dots,n}$
$\mathcal{F}_x$	input feature space
$\mathcal{F}_y$	output feature space
$\phi : \mathbb{R}^p \to \mathcal{F}_x$	input feature map
$\psi : \mathcal{Y} \to \mathcal{F}_y$	output feature map
$\mathbf{w} \in (\mathbb{R}^+)^p$	vector of weights
$\mathbf{w}^{(k)} \in (\mathbb{R}^+)^p$	vector of weights at iteration k of the optimization algorithm
$\mathbf{D}_{\mathbf{w}} \in \mathbb{R}^{p \times p}$	shortcut for $\text{Diag}(\mathbf{w})$
$K_x^{\mathbf{w}}$	kernel defined as $K_x^{\mathbf{w}}(\mathbf{x}_i, \mathbf{x}_{i'}) := K_x(\mathbf{w} \cdot \mathbf{x}_i, \mathbf{w} \cdot \mathbf{x}_{i'})$
$\mathbf{K}_x^{\mathbf{w}}$	kernel matrix associated with $K_x^{\mathbf{w}}$

Table 1. Table of notations.

Unsupervised feature selection.

A penalized distortion criterion. As stated in the previous section, a natural way to select features in $\mathbb{R}^p$ that preserves at best the original kernel structure of K_x would be to search for $\mathbf{w} \in \{0,1\}^p$, solution of:

$$\underset{\mathbf{w} \in \{0,1\}^p}{\text{argmin}} \ \|\mathbf{K}_x^{\mathbf{w}} - \mathbf{K}_x\|_F^2 \qquad \textbf{(1)}$$

$$\text{with } \mathbf{w} \text{ such that } \sum_{j=1}^{p} w_j \leq d,$$

for a given chosen d controlling the sparsity of the solution and where $\|.\|_F$ stands for the Frobenius norm. But the problem above is NP hard, making it particularly difficult to solve in practice for binary constraints on the weights $\mathbf{w}$. In addition, binary constraints can be too stringent: removing a feature j might have an important effect to the full kernel matrix $\mathbf{K}_x^{\mathbf{w}}$ and this effect would be worth compensated by allowing continuous values $w_{j'} \neq 1$ for a feature j' that is correlated to feature j. This is why a relaxation of the discrete optimization problem of Equation (**1**) is proposed in favor of positivity constraints on the weights together with a regularization term that will promote sparsity in the weights.

The relaxation leads to the following optimization problem:

$$\mathbf{w}^* := \underset{\mathbf{w} \in (\mathbb{R}^+)^p}{\text{argmin}} \ \|\mathbf{K}_x^{\mathbf{w}} - \mathbf{K}_x\|_F^2 + \lambda \|\mathbf{w}\|_1, \qquad \textbf{(2)}$$

in which $\lambda > 0$ is a penalization parameter that controls the trade-off between the minimization of the distortion (Frobenius norm between the original kernel and the kernel based on selected features) and the sparsity of the solution (similarly to d in the original discrete optimization problem). To do so, the distortion is penalized using $\|.\|_1$, the ℓ_1 norm: $\|\mathbf{z}\|_1 := \sum_{j=1}^{p} |z_j|$.

The weights $\mathbf{w}$ are often called *scaling parameters* or *slack variables*. Equation (**2**) contains a highly non-convex data fitting term, $\|\mathbf{K}_x^{\mathbf{w}} - \mathbf{K}_x\|_F^2$, that depends on the kernel, but is, otherwise, very similar to the lasso estimator (14). The ℓ_1 norm is non-differentiable in 0, which promotes exact sparsity. When the regularization parameter, λ, is small, the solution of the problem converges to the original kernel with $\mathbf{w} = \mathbf{1}_p$, an all-ones vector, and when the regularization parameter increases, $\mathbf{w}$, becomes sparse and performs automatic feature selection. (25) shows that such a penalization of the weights efficiently enforces sparsity as compared to a direct penalization on the coefficients of their auto-encoder. In addition, it proved efficient in terms of controlling the redundancy of selected features, since feature selection is performed globally.

This unsupervised version of the proposed framework will be termed Unsupervised Kernel Feature Selection (**UKFS**) in the sequel.

Optimization of a non-convex/non-smooth problem. Problem (**2**) is a non-convex and non-smooth optimization problem. It can be reformulated as

$$\underset{\mathbf{w}\in(\mathbb{R}^+)^p}{\operatorname{argmin}} \quad f(\mathbf{w}) + \lambda g(\mathbf{w}) \tag{3}$$

where f is a smooth non-convex function ($f(\mathbf{w}) = \|\mathbf{K}_x^{\mathbf{w}} - \mathbf{K}_x\|_F^2$) and g is the non differentiable ℓ_1 norm promoting sparsity in $\mathbf{w}$ ($g(\mathbf{w}) = \|\mathbf{w}\|_1$).

We propose to solve the optimization problem using proximal gradient descent (43, 44) that is particularly well adapted to ℓ_1 regularized problems. Among those methods, a Forward-Backward Splitting (FBS) is used, which can be seen as a majorize-minimization (MM) algorithm. When the function f is gradient Lipschitz with a Lipschitz constant η, the variation of the function is limited and it can be bounded around a given $\tilde{\mathbf{w}}$ by

$$f(\mathbf{w}) + \lambda g(\mathbf{w}) \leq f(\tilde{\mathbf{w}}) + \nabla_{\mathbf{w}} f(\tilde{\mathbf{w}})^\top (\mathbf{w} - \tilde{\mathbf{w}}) + \frac{\eta}{2}\|\mathbf{w} - \tilde{\mathbf{w}}\|^2 + \lambda g(\mathbf{w}).$$

The FBS algorithm is actually equivalent to iteratively minimizing the upper bound of the previous equation and thus, each iteration leads to a decrease in the objective value. If $\mathbf{w}^{(k)}$ denotes the values of the weights at iteration k, $\mathbf{w}^{(k+1)}$ is thus obtained by minimizing the right hand side of the previous equation around $\tilde{\mathbf{w}} = \mathbf{w}^{(k)}$, which is also equivalent to solving

$$\mathbf{w}^{(k+1)} = \underset{\mathbf{w}\in(\mathbb{R}^+)^p}{\operatorname{argmin}} \quad \frac{1}{2}\left\|\mathbf{w} - \mathbf{w}^{(k+\frac{1}{2})}\right\|^2 + \frac{\lambda}{\eta} g(\mathbf{w}), \tag{4}$$

where $\mathbf{w}^{(k+\frac{1}{2})} = \mathbf{w}^{(k)} - \frac{1}{\eta}\nabla_{\mathbf{w}} f(\mathbf{w}^{(k)})$ can be seen as a gradient descent step *wrt* f.

The right hand side of Equation (**4**) is known as the *proximal operator of* $\frac{\lambda}{\eta} g$, noted $\operatorname{prox}_{\frac{\lambda}{\eta} g}(\mathbf{w}^{(k+\frac{1}{2})})$. When g is the ℓ_1 norm, this operator has the following explicit form: $\forall j = 1,\ldots,p$,

$$w_j^{(k+1)} = \operatorname{sign}\left(w_j^{(k+\frac{1}{2})}\right) \times \left(|w_j^{(k+\frac{1}{2})}| - \frac{\lambda}{\eta}\right)_+$$

with $\operatorname{sign}(u)$ the sign of u and $(u)_+ = \max(0, u)$ the positive part of u. This operator is known as the component-wise soft thresholding and one of its important property is that, due to the threshold, the iterations of the algorithm are sparse, which ensures the sparsity of the solution even in the case of early stopping along the iterations. It also allows for a clear feature selection, as opposed to other approaches proposed to solve the lasso that are based on re-weighting and provide sparsity only at convergence (45).

Also note that the Lipschitz constant is used for a $\frac{1}{\eta}$ gradient step: this choice ensures a decrease at each iteration but leads to a very slow convergence in practice. Some works have shown that using a larger step leads to an important speedup. For instance, one approach known to provide a good gradient step is the so-called *Barzilai-Borwein (BB) rule* (46). This rule first approximates the Hessian matrix as a scaled identity and then uses this estimation to provide a coarse optimal step. It was proposed for non-convex FBS in (47) in conjunction with a line search that ensures a decrease of the objective value at each iteration. Note that this strategy has been used on simulations using the smooth Gaussian kernel, but not on those based on the Bray-Curtis dissimilarity because the Hessian is undefined for non-smooth functions (see Section "Evaluation").

Finally, the non-convexity of problem (**3**) means that there is no hope to get a global minimum but, while FBS has been originally proposed for convex optimization, it has been used in several non-convex applications such as Iterative Hard Thresholding (48). It has also been shown that the algorithm converges to a stationary point on a wide class of non-convex optimization problems (49). Finally, note that optimizing scaling parameters in kernels has already been performed with success in the past using descent algorithms (40, 42) and seems to work well in practice despite the lack of convergence to a global minimum.

Kernel output feature selection.

A penalized association criterion. This section aims at selecting features best able to explain $(y_i)_i \in \mathcal{Y}$, described through the kernel K_y. This type of kernel association problem between $(\mathbf{x}_i)_i$ and $(y_i)_i$ has already been addressed in (50, 51), under the name of Input Output Kernel Regression (**IOKR**). It is equivalent to learning a function $h : \mathcal{X} \to \mathcal{F}_y$ that predicts the output feature vector $\psi(y_i) \in \mathcal{F}_y$ associated with a given $\mathbf{x}_i \in \mathcal{X}$ (see Figure 1, left, for an illustration of the method and notations). In the **IOKR** framework (50), this is done by choosing an operator-valued kernel (OVK), $\mathcal{K}_x$, which is a bilinear form from $\mathcal{X} \times \mathcal{X}$ into $\mathcal{B}(\mathcal{F}_y, \mathcal{F}_y)$, where $\mathcal{B}(\mathcal{F}_y, \mathcal{F}_y)$ is the set of all linear operators from $\mathcal{F}_y$ to $\mathcal{F}_y$. This kernel $\mathcal{K}_x$ allows to define a Reproducing Kernel Hilbert Space (RKHS), $\mathcal{H}$, that is a subspace of the linear operators from $\mathcal{X}$ into $\mathcal{F}_y$ in which h is taken. A simplification of this framework is obtained by defining the OVK from the feature

map ϕ induced by an input scalar kernel. In this case, h can be seen as a composition of the feature map ϕ with an operator V from $\mathcal{F}_x$ to $\mathcal{F}_y$.

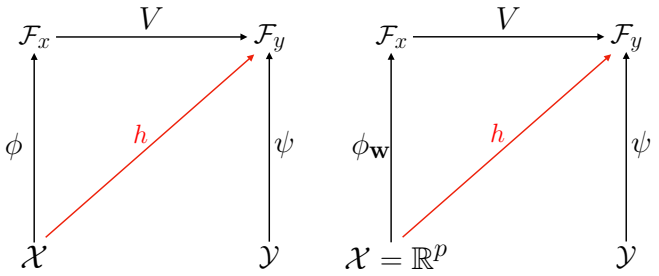

Figure 1. Comparison between the diagram of the simplified **IOKR** framework (left) and of the proposed approach (right). The latter jointly learns the vector of weights $\mathbf{w}$ associated with the features and a function $h : \mathbb{R}^p \to \mathcal{F}_y$ that approximates the output feature map ψ. In the proposed approach, h is modeled as $h(\mathbf{w} \cdot \mathbf{x}) = V(\phi_{\mathbf{w}}(\mathbf{x})), \forall \mathbf{x} \in \mathbb{R}^p$ where $\phi_{\mathbf{w}}(\mathbf{x}) = \phi(\mathbf{D}_{\mathbf{w}}\mathbf{x})$ and $V(.)$ is an operator from $\mathcal{F}_x$ to $\mathcal{F}_y$.

This approach has the advantage of being able to handle a set of possible functions for h, $\mathcal{H} \subset \{\mathcal{X} \to \mathcal{F}_y\}$, that is both very general and well characterized by the OVK RKHS theory. In particular, it is equipped with a scalar product $\|.\|_{\mathcal{H}}$, which is used to add a ridge penalty to a measure of the goodness-of-fit between $(\mathbf{x}_i)_i$ and $(y_i)_i$, as follows:

$$\underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{n} \|h(\mathbf{x}_i) - \psi(y_i)\|^2_{\mathcal{F}_y} + \lambda_1 \|h\|^2_{\mathcal{H}}. \tag{5}$$

Technical details, including the precise definition of $\mathcal{H}$, are left for the next section for the sake of clarity.

Following an idea similar to the one already described in the Section "Unsupervised feature selection", weights $\mathbf{w} \in (\mathbb{R}^+)^p$ are introduced to be jointly learned with h. These weights are used as scaling factors for the input features in $(\mathbf{x}_i)_i$, replacing $\mathbf{x}_i$ by $\mathbf{D}_{\mathbf{w}}\mathbf{x}_i$ in Equation (**5**) to perform the feature selection: a feature j is then not selected when $w_j = 0$ and selected otherwise. The proposed approach is illustrated in Figure 1.

Sparsity is induced on $\mathbf{w}$ by the use of a ℓ_1 penalization, which leads to the following regression problem:

$$\min_{h \in \mathcal{H}, \mathbf{w} \in (\mathbb{R}^+)^p} f(h, \mathbf{w}) + \lambda_1 \|h\|^2_{\mathcal{H}} + \lambda_2 \|\mathbf{w}\|_1, \tag{6}$$

where $f(h, \mathbf{w}) = \sum_{i=1}^{n} \|h(\mathbf{D}_{\mathbf{w}}\mathbf{x}_i) - \psi(y_i)\|^2_{\mathcal{F}_y}$ and $\lambda_1 > 0$ and $\lambda_2 > 0$ are two regularization parameters. The first regularization term is used to control the complexity of the function h and the ℓ_1 norm of the vector $\mathbf{w}$ performs feature selection. This version of the framework will be termed Kernel Output Kernel Feature Selection (**KOKFS**) in the sequel.

Similarly to the approach developed by (52) for their learning of anchor points in their convolutional kernel networks, the optimization scheme alternates between two steps (a) weights $\mathbf{w}$ are fixed and Equation (**6**) is minimized with respect to h; (b) h is fixed and weights $\mathbf{w}$ are updated using one pass of a proximal gradient descent using the explicit computation of $f(h, \mathbf{w})$. (52) perform one stochastic gradient descent step that is well suited to their ℓ_2 penalization

Algorithm 1 Algorithm for solving Problem (**6**).

Initialize $\mathbf{w}^{(0)}$ with $w_j^{(0)} \geq 0$ for $j = 1, \ldots, p$
For $k = 0, \ldots, T-1$ (or until convergence)
1. Learn $h^{(k+1)}$ with fixed $\mathbf{w}^{(k)}$:

$$h^{(k+1)} = \underset{h \in \mathcal{H}}{\operatorname{argmin}} f(h, \mathbf{w}^{(k)}) + \lambda_1 \|h\|^2_{\mathcal{H}} \tag{7}$$

2. Proximal gradient descent step on $\mathbf{w}^{(k+1)}$ with fixed $h^{(k+1)}$
Output: $\mathbf{w}^{(T)}$

whereas, here, a proximal gradient descent step is used because it is more adapted to the ℓ_1 penalization.

The sketch of the optimization approach is provided in Algorithm 1, while details on the two steps are provided in the next two sections. Note that this algorithm is also exactly a FBS that computes the minimum in $\mathbf{w} \in (\mathbb{R}^+)^p$ of $\min_{h \in \mathcal{H}} \left[f(h, \mathbf{w}) + \lambda_1 \|h\|^2_{\mathcal{H}} \right] + \lambda_2 \|\mathbf{w}\|_1$. The two steps described below thus describes the exact computation of the gradient (in $\mathbf{w}$) of this function, thanks to the envelop theorem (53). Similarly to the unsupervised case, one cannot hope to get a global minimum but the algorithm converges to a stationary point on a wide class of non-convex problems (49).

Solution in h for a fixed $\mathbf{w}^{(k)}$. When $\mathbf{w}^{(k)}$ is fixed, the optimization problem of Equation (**7**) is convex in h and has already been solved in (50, 51) in the setting of the RKHS theory for vector-valued functions. A vector-valued RKHS is uniquely characterized by an operator-valued kernel: $\mathcal{K}_x : \mathbb{R}^p \times \mathbb{R}^p \to \mathcal{B}(\mathcal{F}_y, \mathcal{F}_y)$. This framework extends the concept of scalar-valued kernel. Here, the special case of RKHS for vector valued-functions is used, with associated kernel of the form $\mathcal{K}_x(\mathbf{x}, \mathbf{x}') = K_x(\mathbf{x}, \mathbf{x}') I_{\mathcal{F}_y}, \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^p$, where $I_{\mathcal{F}_y}$ is the identity operator. The associated RKHS is denoted $\mathcal{H}$ and is a subset of linear operators from $\mathbb{R}^p$ to $\mathcal{F}_y$. Its specific simple form has been chosen because the size of the feature space $\mathcal{F}_y$ is not necessarily finite so the complexity of the set of all linear operators $\{\mathbb{R}^p \to \mathcal{F}_y\}$ can potentially be very large. The functions h in $\mathcal{H}$ can be written as: $h(\mathbf{z}) = V(\phi(\mathbf{z})), \forall \mathbf{z} \in \mathbb{R}^p$ where V is a linear operator from $\mathcal{F}_x$ to $\mathcal{F}_y$.

When using this operator-valued kernel, the closed-form solution of the optimization problem (**7**) for a fixed $\mathbf{w}^{(k)}$ is given by:

$$\forall \mathbf{z} \in \mathbb{R}^p, \qquad h^{(k+1)}(\mathbf{z}) = \sum_{i=1}^{n} \alpha_i(\mathbf{z}) \psi(y_i), \tag{8}$$

where $\boldsymbol{\alpha}(\mathbf{z}) = (\lambda_1 \mathbf{I}_n + \mathbf{K}_x^{\mathbf{w}^{(k)}})^{-1} \kappa_{\mathbf{w}^{(k)}}(\mathbf{z})$ with $\mathbf{I}_n$ the identity matrix of size n, $\mathbf{K}_x^{\mathbf{w}^{(k)}}$ the $(n \times n)$ kernel matrix with $[\mathbf{K}_x^{\mathbf{w}^{(k)}}]_{i,i'} = K_x(\mathbf{D}_{\mathbf{w}^{(k)}}\mathbf{x}_i, \mathbf{D}_{\mathbf{w}^{(k)}}\mathbf{x}_{i'})$ and $\kappa_{\mathbf{w}^{(k)}} : \mathbb{R}^p \to \mathbb{R}^n$ a function defined as $\kappa_{\mathbf{w}^{(k)}}(\mathbf{z}) = [K_x(\mathbf{D}_{\mathbf{w}^{(k)}}\mathbf{x}_1, \mathbf{z}), \ldots, K_x(\mathbf{D}_{\mathbf{w}^{(k)}}\mathbf{x}_n, \mathbf{z})]^\top, \forall \mathbf{z} \in \mathbb{R}^p$.

In practice, $h^{(k+1)}$ cannot be computed explicitly because the output feature vectors, $\psi(y_i)$, do not have an explicit form

and can potentially be infinitely long. However, $h^{(k+1)}$ is only used through the computation of $f(h^{(k+1)},\mathbf{w}^{(k)})$ (that evaluates $h^{(k+1)}$ at points of the form $\mathbf{z}=\mathbf{w}\cdot\mathbf{x}$), which can be explicitly computed using the kernel trick in the output feature space $\mathcal{F}_y$.

Proximal gradient descent step on $\mathbf{w}^{(k+1)}$. The minimization problem of (**6**) in $\mathbf{w}$ with a fixed h is a non-convex and non-smooth problem that is solved using a proximal gradient method (a single pass is used at each step of the method). Similarly to (41), $\tilde{f}_{h^{(k+1)}}=f(h^{(k+1)},.)$ is first approximated at the current point $\mathbf{w}^{(k)}$ by a linear function, thanks to its first-order Taylor expansion: $\tilde{f}_{h^{(k+1)}}(\mathbf{w})\approx \tilde{f}_{h^{(k+1)}}(\mathbf{w}^{(k)})+\bigtriangledown\tilde{f}_{h^{(k+1)}}(\mathbf{w}^{(k)})^T(\mathbf{w}-\mathbf{w}^{(k)})$, valid for $\mathbf{w}$ in the neighborhood of $\mathbf{w}^{(k)}$. Hence, to ensure that $\mathbf{w}$ stays close to $\mathbf{w}^{(k)}$, a ridge penalty, $\|\mathbf{w}-\mathbf{w}^{(k)}\|_2^2$, is added to the optimization problem that thus becomes:

$$\mathbf{w}^{(k+1)}= \underset{\mathbf{w}\in(\mathbb{R}^+)^p}{\operatorname{argmin}}\left[\bigtriangledown\tilde{f}_{h^{(k+1)}}(\mathbf{w}^{(k)})^\top(\mathbf{w}-\mathbf{w}^{(k)}) +\frac{\eta^{(k)}}{2}\|\mathbf{w}-\mathbf{w}^{(k)}\|_2^2+\lambda_2\|\mathbf{w}\|_1\right],$$

where $\frac{1}{\eta^{(k)}}$ is called the step size because it is involved in a gradient descent step similar to the one already described for the unsupervised case.

Similarly to the unsupervised case, this optimization problem is solved using a proximal approach and leads to obtain an explicit form for the update:

$$\mathbf{w}^{(k+1)}= \left(\mathbf{w}^{(k)}-\frac{1}{\eta^{(k)}}\bigtriangledown\tilde{f}_{h^{(k+1)}}(\mathbf{w}^{(k)})-\frac{\lambda_2}{\eta^{(k)}}\right)_+, \quad (\mathbf{9})$$

where $(\mathbf{u})_+$ is the $\mathbb{R}^p$ vector with entries $\max(u_j,0)$, $\forall\mathbf{u}\in\mathbb{R}^p$.

When further replacing $h^{(k+1)}$ by the solution given in Equation (**8**), the following expression is obtained for $\bigtriangledown\tilde{f}_{h^{(k+1)}}(\mathbf{w}^{(k)})$: $\forall j=1,\ldots,p$,

$$\left(\bigtriangledown\tilde{f}_{h^{(k+1)}}(\mathbf{w}^{(k)})\right)_j= 2\mathrm{Tr}\left((\mathbf{K}_x^{\mathbf{w}^{(k)}}\mathbf{A}-\mathbf{I}_n)\mathbf{K}_y\mathbf{A}\mathbf{E}_j\right), \quad (\mathbf{10})$$

where Tr is the trace operator, $\mathbf{A}=(\lambda_1\mathbf{I}_n+\mathbf{K}_x^{\mathbf{w}^{(k)}})^{-1}$ and $\mathbf{E}_j\in\mathbb{R}^{n\times n}$, for $j=1,\ldots,p$, is defined as: $\forall i=1,\ldots,n$,

$$[\mathbf{E}_j]_{\cdot,i}=\frac{\partial}{\partial w_j}\left(\kappa_{\mathbf{w}^{(k)}}(\mathbf{D}_{\mathbf{w}}\mathbf{x}_i)\right)\Big|_{\mathbf{w}=\mathbf{w}^{(k)}}, \quad (\mathbf{11})$$

where $[\mathbf{E}_j]_{\cdot,i}$ denotes the ith column of the matrix $\mathbf{E}_j$. The proof of these results are given in Supplementary material.

In conclusion, the implementation of Algorithm 1 thus reduces to the steps described in Algorithm 2.

Algorithm 2 Practical implementation of Algorithm 1.

Initialize $\mathbf{w}^{(0)}$ with $w_j^{(0)}\geq 0$ for $j=1,\ldots,p$
For $k=0,\ldots,T-1$ (or until convergence)
1. Compute $\mathbf{K}_x^{\mathbf{w}^{(k)}}:[\mathbf{K}_x^{\mathbf{w}^{(k)}}]_{ii'}=K_x(\mathbf{D}_{\mathbf{w}^{(k)}}\mathbf{x}_i,\mathbf{D}_{\mathbf{w}^{(k)}}\mathbf{x}_{i'})$ for $i,i'=1,\ldots,n$
2. Compute the p matrices $\mathbf{E}_1,\ldots,\mathbf{E}_p$ as in Equation (**11**)
3. for $j=1,\ldots,p$, $w_j^{(k+1)}=$ $\left(w_j^{(k)}-\frac{2}{\eta^{(k)}}\mathrm{Tr}((\mathbf{K}_x^{\mathbf{w}^{(k)}}\mathbf{A}-\mathbf{I}_n)\mathbf{K}_y\mathbf{A}\mathbf{E}_j)-\frac{\lambda_2}{\eta^{(k)}}\right)_+$
Output: $\mathbf{w}^{(T)}$

Extensions with prior knowledge regularization.

Note that the proposed framework is flexible enough to allow the incorporation of additional *a priori* knowledge. For instance, in the unsupervised feature selection framework, an interesting case is when some *a priori* relations between these features are given. This can happen when the features represent taxons for which an interaction network is known or simply using the observed correlations between features as a proxy of their relations. These relations can be represented by a graph, $\mathcal{G}$, whose p vertices correspond to the p original features and whose edges (that can be weighted with positive weights) are the relations between these features.

Similarly to (21, 54, 55), this information can be used to perform a regularization based on the Laplacian of $\mathcal{G}$, $\mathbf{L}_\mathcal{G}$ and leads to extend Equation (**2**) to the following optimization problem

$$\mathbf{w}^*:=\underset{\mathbf{w}\in(\mathbb{R}^+)^p}{\operatorname{argmin}}\|\mathbf{K}_x^{\mathbf{w}}-\mathbf{K}_x\|_F^2+\mu\mathbf{w}^\top\mathbf{L}_\mathcal{G}\mathbf{w}+\lambda\|\mathbf{w}\|_1, \quad (\mathbf{12})$$

with $\mu>0$ a second regularization parameter. The optimization problem related to this extension can be solved similarly to the original problem.

Note that other types of *a priori* information, like a cluster structure, could also be incorporated to the original optimization problem of Equation (**2**) or of Equation (**6**) by introducing alternative constrains in a similar fashion. Two such extensions are already implemented in the package **mixKernel**: the structure based regularization of Equation (**12**) and another extension that uses kernel PCA (56) and that is designed to reduce the distance distortion in dimension reduction tasks.

Experimental evaluation setup.

Benchmark datasets for the evaluation of the unsupervised feature selection. To evaluate the accuracy of the unsupervised version of the proposed approach, a benchmark of three biological datasets was analyzed: two microarray datasets, denoted in the sequel by "*Carcinom*" and "*Glioma*", provided in the Python package **scikit-feature**, and a DNA barcoding dataset, denoted by "*Koren*", available from the R package **mixOmics** (28).

"*Carcinom*" and "*Glioma*" datasets respectively contain the expression of 9,182 genes obtained from 174 samples and 4,434 genes from 50 samples. To perform the feature selection on these datasets, **UKFS** was used with the Gaussian kernel $K_x(\mathbf{x}_i,\mathbf{x}_{i'})=e^{-\sigma^*\|\mathbf{x}_i-\mathbf{x}_{i'}\|^2}$ with σ^* chosen so as to minimize the reproduced inertia in the projection on the first two axes of the KPCA with kernel K_x.

"*Koren*" includes the abundance of 973 operational taxonomic units (OTUs) collected from 43 samples. **UKFS** was used with the kernel induced by the Bray-Curtis dissimilarity between samples on raw abundances, $d_{BC}(\mathbf{x}_i,\mathbf{x}'_i)=\frac{\sum_{s=1}^{p}|\mathbf{x}_{is}-\mathbf{x}_{i's}|}{\sum_{s=1}^{p}(\mathbf{x}_{is}+\mathbf{x}_{i's})}$ (p is the number of observed OTUs). An interesting property of this dissimilarity is that it avoids the need to use one of the standard pre-processing steps to account for the compositional nature of this dataset (*e.g.*, total sum scaling normalization (TSS) and centered log ratio transformation (CLR)).

UKFS was then compared to several alternatives:

- two methods based on the computation of a score: the Laplacian score (denoted by **lapl**) (57) and **SPEC** (18);
- three methods based on a learning approach constrained to a sparse representation. These methods were mostly designed for clustering (or are based on the implicit assumption that samples are structured into subgroups) and require the *a priori* definition of a number of clusters: **MCFS** (19), **NDFS** (21) and **UDFS** (22);
- one neural network: the concrete autoencoder, denoted **Autoencoder** (58). This method can handle supervised and unsupervised frameworks and the unsupervised setting is used for the purpose of comparison.

Except for **Autoencoder** that requires large computational resources and CPU computing, simulations were all performed on the same 40-node computer without concurrent access. The Python implementation available from the **scikit-feature** package (13) https://github.com/jundongl/scikit-feature was used for **lapl**, **SPEC**, **MCFS**, **NDFS** and **UDFS** and a Python implementation based on *Keras* https://keras.io/ was used for **Autoencoder** https://github.com/mfbalin/Concrete-Autoencoders. To address the underlying compositional structure of "*Koren*" with these methods, standard pre-processing steps, *i.e.*, total sum scaling normalization (TSS) and centered log ratio transformation (CLR), were applied before selecting the relevant features. No pre-processing was performed for the other two datasets.

To evaluate the different methods, the following steps were used:

1. each method was run to select d features with increasing values of $d\in\{10,20,...,290,300\}$, except for **UKFS** for which d is given by the number of selected features when increasing the regularization parameter, λ. The solutions for different values of λ were obtained using a warm restart strategy: the first solution was computed for a small λ and subsequent solutions were obtained using the previously obtained solution as an initial value for the method. 30 values of λ were used, with an exponential increase. Whereas this strategy shrinks the obtained solution around the values found for the initial λ, it has the advantage of providing a consistent and approximately smooth evaluation of the evolution of the quality criterion;
2. based on the selected features, the kernel k-means algorithm was repeated 20 times, using the Gaussian kernel for "*Carcinom*" and "*Glioma*" and the Bray-Curtis dissimilarity for "*Koren*". Data were clustered into C classes, in which C was chosen as the true value of the underlying clustering ($C=11$ for "*Carcinom*", $C=4$ for "*Glioma*" and $C=3$ for "*Koren*");
3. the relevance of the obtained feature selection was then obtained as its ability to recover the true underlying classification structure of the dataset. This true partition is used as ground truth to compute standard clustering performance metrics, *i.e.*, the normalized mutual information (NMI, (59)) and the overall accuracy (ACC). More precisely, the average NMI and ACC are computed over the 20 runs of the k-means algorithm and the ability of every method to select non redundant information is also evaluated by computing the average Kendall correlation between selected features. Kendall correlation was chosen over Pearson correlation to account for the fact that input features can have distributions strongly departing from the Gaussian and a high skewness. For "*Koren*", correlations were computed using raw counts.
   Note that **UKFS** is not specifically optimized for this type of problem, contrary to **MCFS**, **NDFS** and **UDFS**, which explicitly have a cluster structure assumption and for which C, the *a priori* number of clusters of the method, was set to its true value (usually not known in advance).

Evaluation of the structure based extension of unsupervised feature selection. The structure based version of the proposed approach (Equation (**12**)), was also evaluated in a similar fashion against alternative methods. Two datasets were used: the "*Koren*" dataset described in the previous section and another dataset included in the R package **mixOmics**, called "*HMP*". This latter dataset is a Human Microbiome 16S dataset, including OTU counts on the three most diverse bodysites: Subgingival plaque (Oral), Antecubital fossa (Skin) and Stool, sampled from 54 unique individuals for a total of 162 samples. The 1,674 OTUs processed and included in **mixOmics** are a subsample of the full dataset where OTU counts below 0.01 percent compared to the total count were filtered out. The original dataset can be downloaded from the Human Microbiome Project (http://hmpdacc.org/HMQCP/all/). Relations between features (OTUs for both cases) were obtained by computing the Pearson correlation matrix, which was used as the adjacency matrix of the graph (the method is denoted by **UKFS-G** in the sequel).

Evaluation of the kernel output feature selection for multiple output regression problems. To evaluate the accuracy of **KOKFS**, it was first used to solve multiple output regression problems, *i.e.*, cases where $\mathcal{Y}=\mathbb{R}^q$. This setting indeed

provides an easy way to evaluate the method performance: more precisely, KOKFS was run with Gaussian input and output kernels and selected features were evaluated by assessment of their predictive power (this can be performed with any regression method). Note that contrary to standard multivariate linear regression incorporating feature selection (like multivariate lasso implemented in the R package **glmnet** (26)), the proposed approach can 1/ use a more adapted norm than the Euclidean norm for large dimensional input spaces and adapts the feature selection to this norm and 2/ does not directly aim at predicting Y but rather at selecting features that best explain the global resemblance between samples as described by the kernel K_y.

The considered datasets were:

1. The "*Nutrimouse*" dataset (60), available in the R package **mixOmics** (28). This dataset contains the expressions of $p=120$ genes (obtained by RT-qPCR) and the concentration of $q=21$ hepatic fatty acids for $n=40$ mice.

2. The "*Diogenes*" dataset, described in (61, 62) and available on GEO Gene Expression Omnibus (GEO repository, http://www.ncbi.nlm.nih.gov/geo/) under the accession number GSE95640. This dataset contains gene expression acquired by RNA-Seq on human adipose tissue at two different time steps (CID1 and CID2) of a dietary intervention (before and after a 8 week low calorie diet) for $n=167$ individuals. The expression of $p=q=269$ genes, respectively corresponding to expressions before and after the diet, were used in the experiments. These genes were genes of interest for the biologists and corresponded to genes listed in a parallel study on the same individuals (available on GEO under the accession number GPL19141). Prior analysis, gene expressions were log-transformed.

3. The "*TCGA*" dataset based upon data generated by The Cancer Genome Atlas (TGCA) Research Network (https://www.cancer.gov/tcga). The subset is restricted to primary tumor samples of breast cancer for which mRNA and miRNA expressions were both available. Expression data were log-transformed and corrected for a batch effect (plate) prior analysis. The dataset was composed of $n=1194$ individuals for which $p=655$ miRNA and $q=9884$ mRNA were available.

Then, experiments were conducted in two steps: a first step for feature selection and a second one for performance assessment with nonparametric regressions based on the selected features.

For the **feature selection step**, Gaussian kernels were used for the input and output kernels. The parameters of both Gaussian kernels were set to $\frac{n(n-1)}{\sum_{i\neq i'}\|z_i-z_{i'}\|_2^2}$, where z stands for either $\mathbf{x}$ or y. The hyper-parameters, λ_1 and λ_2, as described in Equation (**6**), were set as follows: for $\lambda_2=0$ (no feature selection), λ_1 (that controls the complexity of the function h) was first tuned by 5-fold cross-validation based on averaged mean squared error minimization. The selected value was then used for all values of λ_2. λ_2 (that controls the sparsity of the weights and thus performs feature selection) was varied using a warm restart strategy in order to obtain the solutions for increasing values of λ_2. The solution obtained for a small value of λ_2 was first computed and then the algorithm was run for the next λ_2 value using the solution previously obtained as the starting point for $\mathbf{w}$. A line search procedure was used for setting a value for the step size that ensured a decrease of the objective at each iteration. Features were ordered by their order of appearance in the selection along the λ_2 paths.

For the **regression step**, the features selected by **KOKFS** were then passed to a SVM regression to predict each of the q output features in $\mathcal{Y}$. More precisely, for a number of selected features between 1 and 40 (ordered by the regularization path), regressions were performed by ϵ-regression SVM with Gaussian kernel and hyperparameters were tuned by cross-validation as implemented in the R package **e1071**. For each output variable, pseudo-R^2 was computed as:

$$\text{pseudo}-R^2(\mathbf{y}_j)=1-\frac{\sum_{i=1}^{n}(y_{ij}-\hat{y}_{ij})^2}{\text{Var}(\mathbf{y}_j)} \qquad (\mathbf{13})$$

in which $\mathbf{y}_j$ stands for the jth variable and $\hat{y}_{ij}$ is the value predicted by the SVM from the selected features for the ith sample and the jth output variable.

Since $q=9{,}884$ for "*TCGA*", this regression step was restricted to $q'=50$ variables only. To do so, a hierarchical clustering was performed, with Ward's linkage based on a distance obtained from the correlation between genes. The dendrogram resulting from the hierarchical clustering was cut at 50 clusters and, in each cluster, a single output variable was selected, which was the most correlated in average with the other variables from its cluster.

The overall approach was compared to predictions made by several alternatives:

- the multivariate Gaussian lasso method implemented in the R package **glmnet** (26), in which a group lasso penalty is used to select features common to all linear models fitted during the learning. The multivariate Gaussian lasso method was used in two different ways: either the direct predictions of the multivariate Gaussian lasso were used to compute a pseudo-R^2 as in Equation (**13**) or the selected features were submitted to the same procedure than the features selected by **KOKFS** (*i.e.*, SVM regression for each output variable, followed by pseudo-R^2 computation);

- the multiple output regression **Relief**, as described in (32) and available through their Java program CLUS http://source.ijs.si/ktclus/clus-public/;

- the multiple output random forest (**RF**), also described in (32) and available through their Java program CLUS http://source.ijs.si/ktclus/clus-public/;

- the standard and block version of **HSIC** lasso (34, 39), available in the pyHSIClasso Python package https://github.com/riken-aip/pyHSIClasso.

The three last methods were used in replacement to **KOKFS** in the feature selection step described above and subjected to the same type of regression step as the features selected by **KOKFS**. The number of features included in the prediction

was varied from 1 to 40 by order of appearance along the regularization path (multivariate lasso, **HSIC** lasso) or by ranking of the features (**Relief**, **RF**).

In addition, for "*Nutrimouse*", the selected features were also compared to the set of features extracted by **CCA** in the seminal work (29), by submitting them to SVM regression.

Finally, running times for every method were also obtained using a single core of the same computer. The provided running times were the ones needed to select 40 variables or, as **Relief** and **RF** are ranking methods, the running time for ranking all the variables. For multivariate lasso, the running time was measured for computing the regularization path for 100 values of λ (with default regularization path provided in the R package **glmnet**. Finally, to account for the need to tune the hyperparameter λ_2 in the running time of **KOKFS**, the space of λ_2 was explored in order to select the value for which 40 variables are selected. **KOKFS** was first run for λ_2 in the grid val_$\lambda_2 = \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$. If the number of selected variables gets below 40 at the tth value of the grid, the algorithm was stopped and a new grid of 5 log spaced values was considered between val_$\lambda_2[t-1]$ and val_$\lambda_2[t]$. This process was iterated until a value of λ_2 was found for which 40 variables are selected. Running times were averaged over 10 repetitions of the methods.

Evaluation of kernel output feature selection with time series outputs. To evaluate the accuracy of **KOKFS** on outputs beyond the case of multiple numeric outputs, an example with time series outputs was used. For such outputs, the Euclidean distance is meaningless because it doesn't account for the natural order of the numeric values in the series, which is directly induced from the times of measurements. Solutions to overcome these limitations include the use of dedicated mixed models (as for the extension of random forest in (63)) or the computation of a relevant similarity measure between time series. For instance, Fréchet distances (64) can provide informative insights on the resemblance and are increasingly used for the analysis of various type of non Euclidean data (65).

To illustrate the usefulness of the proposed approach when the resemblance between outputs is well embedded into Fréchet distance, the data on the impact of various interventions on Covid-19 epidemic was used. These data have previously been collected in (66) and are available in their github repository https://github.com/complexity-science-hub/ranking_npis. More precisely, their L2 version of non-pharmaceutical interventions (NPI) data was used, which consisted in the encoding of governmental interventions to face the Covid-19 pandemic into 46 themes for 79 countries during 261 days ranging from March to July 2020. Input features X consisted of the activation of these various interventions (encoded as 0/1 to indicate if this intervention was activated or not) in every country at different days. The Gaussian kernel was used for computing a similarity between input features.

To avoid redundancy in the dataset, only one day for each month was selected (the last day of the month). The corresponding output data, Y, consisted of the evolution of the reproduction number at date t, R_t, and during the 20 days after the observation date, $R_{t+1}, \ldots, R_{t+20}$. The Fréchet distances between time series were obtained and log-transformed to improve their discriminating power. An associated similarity matrix was obtained by subtracting the Fréchet distance values to their maximum. This matrix was used as the output kernel. Removing all couples of (country, date) that had at least one missing value, the final dataset consisted of $n = 365$ different observations for $p = 46$ governmental interventions.

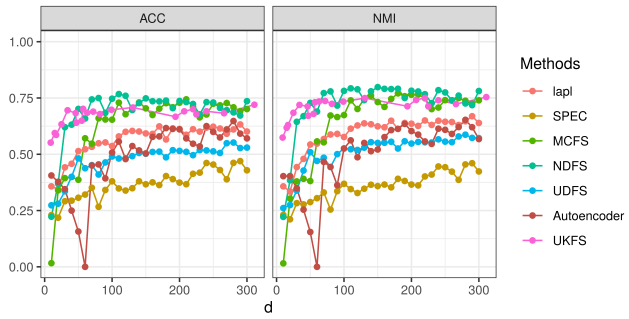

Figure 2. Comparison of the different approaches on "*Carcinom*" in terms of ACC (left) and NMI (right) versus the number of selected features, d, computed from kernel k-means results using only the selected features.

As in the previous section, λ_2 was varied using a warm restart strategy. Given the shape of the λ_2 regularization path (see Figure S1 of supplementary material), features were ranked by their maximum associated weight along the path, which was fairly similar to (but more discriminating than) their order of appearance in the selection. No ground truth is available for this problem so the ranking obtained by (66) was used and a Spearman correlation test was performed to assess if the two rankings were significantly related.

RESULTS

This section provides results obtained for the simulation setting described in the previous section. First, results obtained for the unsupervised framework are given with several benchmark datasets for the basic version and with two additional datasets for the structure based version. Then, results obtained for the kernel output framework are given, with several multiple output regression problems and then with time series output. Discussion on the obtained results is provided in the next section.

Benchmark datasets for the evaluation of unsupervised feature selection.

Table 2 presents the results obtained for **UKFS** and its competitors on the 3 benchmark datasets, in terms of ACC and NMI for $d \in \{10, 300\}$, of average CPU time for one run over the range of d and of the average area under the curve (AUC) of NMI, ACC and COR over the range of d. CPU time for **Autoencoder** for "*Carcinom*" is not given with the same exactness than for the other methods because this method was run in parallel contrary to the others. In addition, Figure 2 provides a comparison of the different approaches in terms of NMI and ACC evolution on "*Carcinom*" versus the number of selected features, d.

Results demonstrate a high efficiency of the proposed approach to select relevant features with no *a priori* on the number of clusters present in the data. For the three tested

Table 2. Comparison of the different approaches in terms of CPU times in seconds, ACC and NMI computed from kernel k-means results based on the first 10 or 300 selected features (average over 20 clustering results and standard deviations between parenthesis). COR and average AUC of ACC, NMI and COR over the full range of tested d are also reported.

	lapl	SPEC	MCFS	NDFS	UDFS	Autoencoder	UKFS
			"*Carcinom*" ($n=174$, $p=9{,}182$)				
ACC (10)	0.36 (0.03)	0.23 (0.17)	0.02 (0.07)	0.22 (0.28)	0.27 (0.07)	0.41 (0.21)	**0.55** (0.04)
NMI (10)	0.36 (0.02)	0.23 (0.17)	0.02 (0.07)	0.22 (0.28)	0.26 (0.06)	0.40 (0.21)	**0.57** (0.03)
ACC (300)	0.60 (0.05)	0.43 (0.11)	0.70 (0.05)	**0.74** (0.06)	0.53 (0.05)	0.57 (0.06)	0.72 (0.07)
NMI (300)	0.64 (0.04)	0.42 (0.10)	0.74 (0.04)	**0.78** (0.03)	0.57 (0.03)	0.57 (0.05)	0.75 (0.05)
ACC AUC	164.02 (3.14)	106.52 (6.99)	184.17 (7.23)	200.88 (7.78)	138.48 (4.13)	143.13 (4.30)	**206.55** (5.62)
NMI AUC	172.50 (3.00)	103.66 (6.75)	189.96 (6.96)	212.46 (7.78)	148.78 (3.72)	145.09 (4.72)	**218.96** (3.82)
COR AUC	28.14	30.75	29.56	27.49	30.30	33.18	**24.75**
CPU time	**0.25** (0.04)	2.47 (0.39)	11.69 (5.21)	6,162 (305)	99,138 (2,913)	>4 days	326 (52)
			"*Glioma*" ($n=50$, $p=4{,}434$)				
ACC (10)	**0.66** (0.04)	0.41 (0.01)	0.60 (0.01)	0.46 (0.04)	0.47 (0.03)	0.53 (0.04)	0.53 (0.06)
NMI (10)	**0.50** (0.03)	0.16 (0.01)	0.49 (0.01)	0.20 (0.04)	0.17 (0.02)	0.34 (0.04)	0.26 (0.05)
ACC (300)	0.58 (0.07)	0.49 (0.03)	**0.64** (0.04)	0.52 (0.04)	0.52 (0.06)	0.58 (0.06)	0.57 (0.07)
NMI (300)	0.47 (0.06)	0.24 (0.03)	**0.52** (0.02)	0.36 (0.07)	0.27 (0.06)	0.35 (0.05)	0.42 (0.05)
ACC AUC	166.31 (2.43)	140.72 (1.13)	172.78 (2.37)	147.77 (2.32)	147.50 (3.14)	132.76 (3.81)	**178.57** (9.43)
NMI AUC	134.79 (2.55)	68.32 (1.13)	**145.89** (1.39)	93.72 (3.83)	70.60 (2.45)	71.81 (2.63)	127.09 (9.68)
COR AUC	81.70	70.70	76.43	68.02	72.33	**45.96**	52.14
CPU time	**0.02** (0.00)	0.63 (0.02)	1.05 (0.01)	368 (21)	2,636 (93)	42 162.29 (8 721.86)	23.74 (4.03)
			"*Koren*" ($n=43$, $p=980$)				
ACC (10)	0.48 (0.09)	0.68 (0.05)	0.82 (0.10)	0.80 (0.08)	**0.94** (0.09)	0.58 (0.06)	0.84 (0.17)
NMI (10)	0.13 (0.12)	0.39 (0.06)	0.62 (0.12)	0.61 (0.11)	**0.90** (0.11)	0.33 (0.08)	0.71 (0.10)
ACC (300)	0.74 (0.18)	0.80 (0.13)	0.77 (0.16)	0.87 (0.18)	0.87 (0.15)	0.86 (0.17)	**0.89** (0.02)
NMI (300)	0.53 (0.27)	0.61 (0.19)	0.66 (0.17)	0.78 (0.21)	0.76 (0.22)	0.78 (0.21)	**0.80** (0.05)
ACC AUC	172.90 (5.65)	225.25 (6.64)	233.94 (6.71)	263.04 (4.40)	**263.48** (5.61)	239.76 (8.96)	242.39 (8.71)
NMI AUC	88.29 (8.40)	163.35 (9.46)	186.58 (7.32)	**236.38** (6.87)	234.37 (6.38)	207.48 (11.43)	216.29 (12.18)
COR AUC	48.18	52.34	49.94	48.48	48.69	**32.60**	47.77
CPU time	**0.01** (0.00)	0.07 (0.01)	1.11 (0.12)	5.88 (0.26)	9.70 (0.36)	1 650.46 (224.47)	10.69 (0.03)

datasets, **UKFS** is in the range of or surpasses results obtained with other methods. More precisely, Table 2 shows that **UKFS**, **MCFS** and **UDFS** respectively obtain the best results on "*Carcinom*", "*Glioma*" and "*Koren*" datasets. On the contrary, both score based methods exhibit poor performances, except on the "*Glioma*" dataset for which **lapl** ranks first when the clustering uses only 10 selected features. Figure 2 confirms these results and shows that **UKFS** selects features allowing to produce clustering with a quality fairly similar to those obtained by two methods designed for such purpose, *i.e.*, **NDFS** and **MCFS**. This is especially true for the situation in which a very small number of features are selected (with less than 50 selected features, **UKFS** obtains performances comparable to the best method with more than 100 selected features).

From the point of view of the redundancy of selected features, **UKFS** takes advantage of the joint selection of features to exhibit a correlation between selected features which is smaller, in average, than the one obtained by all the other methods.

Finally, Figure 3 shows the impact of an incorrect setting of the *a priori* number of clusters for methods that requires this information (the example is given for the "*Carcinom*" dataset with **MCFS**). The performances of this method are negatively impacted for an over-estimation of C (30 instead of 11) and strongly negatively impacted for an under-estimation of C (2 instead of 11).

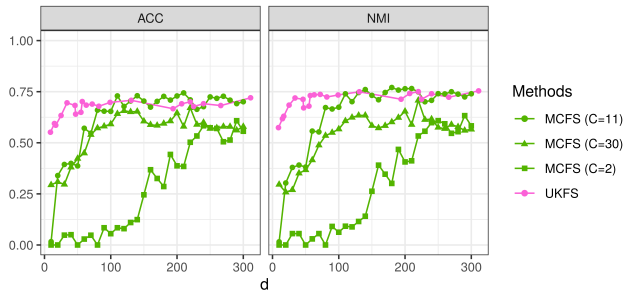

Figure 3. Influence of the **MCFS** settings on its performances for "*Carcinom*". Presented results correspond to different numbers of clusters $C \in \{2,11,30\}$ (the true number of clusters is $C=11$).

Structure based extension of unsupervised feature selection.

Figure 4 provides the results of the comparison of **UKFS** and **UKFS-G** (structure based version of **UKFS**) on "*Koren*" and Figure 5 and Table 3 the results of the comparison of **UKFS** and **UKFS-G** with all the other methods on "*HMP*".

On "*Koren*", using the graph only gives a very slight improvement compared to the original method (average ACC AUC equal to 267.96 (8.03) compared to 242 (8.71) without the constraint, which makes it the best method in terms of ACC AUC compared to results reported in Table 2 on "*Koren*"). For "*HMP*", the improvement is more important and allows **UKFS-G** to compete with the best approach on this

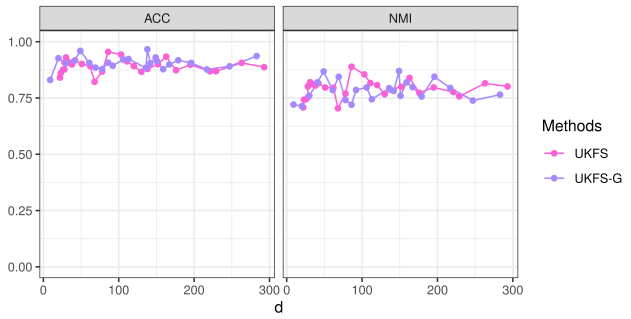

Figure 4. Comparison of **UKFS** and **UKFS-G** on the "*Koren*" dataset. ACC (left) and NMI (right) as obtained from kernel k-means using the selected features.

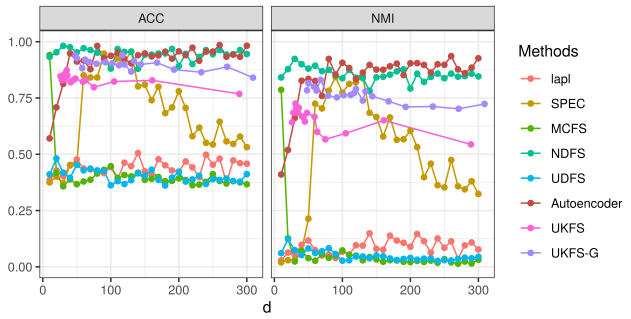

Figure 5. Comparison of **UKFS-G** with all alternative methods on the "*HMP*" dataset. ACC (left) and NMI (right) as obtained from kernel k-means using the selected features.

dataset (**NDFS** that uses the true number of clusters, $C=3$, which is a strong advantage for this dataset). Overall, not only do these results show that adding the structure constraint does not alter the performance of **UKFS** but, in some cases, it can improve it.

Evaluation of kernel output feature selection for multiple output regression.

Figure 6 provides the evolution of the average pseudo-R^2 (over the q outputs) for the different feature selection methods, when the number of selected features, d, used in the regression increases from 4 to 40. Note that it was not possible to obtain feature selection for multivariate lasso and **RF** for "*TCGA*" due to the large size of the dataset. On the contrary, block **HSIC** lasso selection was not computed for "*Nutrimouse*" since the dataset was very small (and the approach is thus not relevant in this case, compared to standard **HSIC**).

All figures show the same tendencies: **KOKFS** selects features that, combined with SVM prediction, outperform the prediction performance of features selected with multivariate lasso, with **HSIC** lasso, with **RF** or with regularized **CCA**. The comparison with **Relief** is less clear but **Relief** has strongly varying performances, from being the best method for "*Diogenes*" to being the worst for "*TCGA*".

In addition, Figure 7 provides information about redundancy between selected features by displaying their average absolute value of Pearson correlation versus the number of selected features, d. For the three benchmarks, **KOKFS** is the method that extract the less redundant features (smallest average correlation), only comparable with Relief for "*Diogenes*" and "*TCGA*" and with RF for "*Nutrimouse*". However, **Relief** exhibited poor performance on "*TCGA*" and **RF** also exhibited poor performance on "*Nutrimouse*".

Finally, Averaged running times of each method is provided in Table 4. The fastest methods are **Relief**, **HSIC** lasso and block **HSIC** lasso, with a clear advantage to block **HSIC** lasso when both the (input and output) dimension and the sample size are large. **KOKFS** is the slowest method on the smallest dataset ("*Nutrimouse*") but scales better for larger (input and output) dimensions ("*Diogenes*") and larger sample size ("*TCGA*") than the other two methods (**RF** and multivariate lasso). Note that, for these two methods, it was not possible to compute the solution for "*TCGA*" due to overloaded memory and that **RF** also overloaded the CPU for all simulations (so the reported running times are not really comparable with that of the other methods).

Evaluation of kernel output feature selection with time series outputs.

Table 5 provides the ranking of the government measures as provided by the feature selection approach **KOKFS** to explain the similarity between $(R_t)_t$ time series evolution during the 20 following days. Even though, contrary to (66), the purpose was not to directly find the government measures the most effective for reducing the reproduction number $(R_t)_t$ but merely to find the measures that most explain similarity in the reproduction number evolution, the found conclusions were very similar to the ones of (66): among important measures, border health check, individual movement restrictions, the

Table 3. "*HMP*". Comparison of the different approaches in terms of ACC and NMI computed from kernel k-means results based on the first 10 or 300 selected features (average over 20 clustering results and standard deviations between parenthesis). Average AUC of ACC and NMI over the full range of tested d are also reported.

"*HMP*" ($n=162$, $p=1{,}674$)								
	lapl	SPEC	MCFS	NDFS	UDFS	Autoencoder	UKFS	UKFS-G
ACC (10)	0.39 (0.03)	0.38 (0.03)	0.94 (0.01)	0.93 (0.09)	0.41 (0.04)	0.57 (0.1)	0.85 (0.07)	**0.94** (0.02)
NMI (10)	0.03 (0.02)	0.02 (0.01)	0.79 (0.01)	**0.84** (0.05)	0.06 (0.03)	0.41 (0.06)	0.54 (0.13)	0.78 (0.11)
ACC (300)	0.46 (0.05)	0.53 (0.16)	0.37 (0.03)	0.94 (0.08)	0.41 (0.05)	**0.98** (0.01)	0.77 (0.10)	0.84 (0.13)
NMI (300)	0.08 (0.04)	0.32 (0.20)	0.03 (0.03)	0.85 (0.05)	0.04 (0.03)	**0.93** (0.01)	0.64 (0.09)	0.72 (0.06)
ACC AUC	127.89 (2.68)	199.39 (8.18)	116.06 (2.39)	**273.95** (4.40)	116.58 (2.64)	267.12 (5.80)	213.37 (11.58)	231.72 (6.00)
NMI AUC	25.52 (2.40)	148.06 (8.61)	14.47 (2.15)	**247.92** (3.28)	13.84 (1.46)	245.76 (3.80)	160.62 (16.24)	192.78 (7.12)

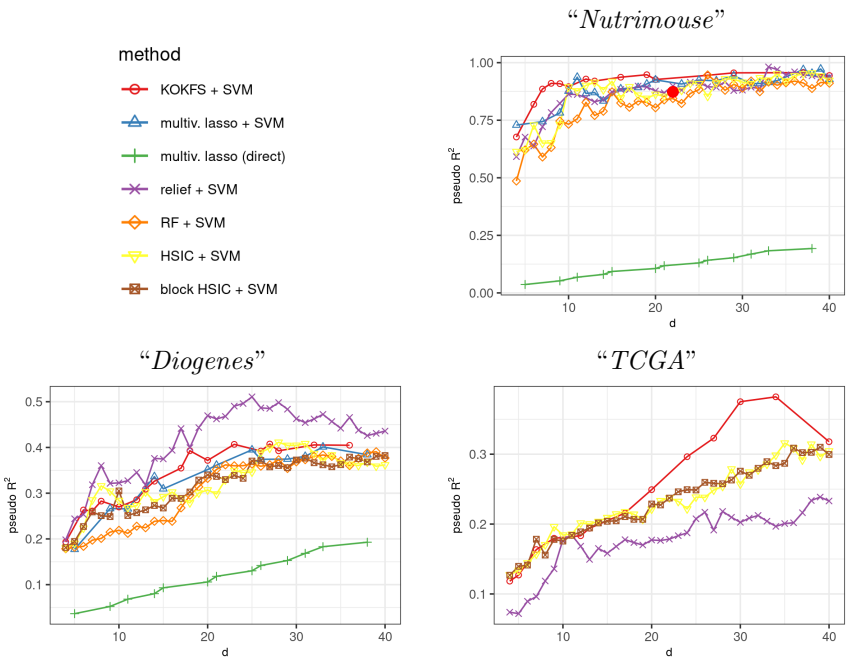

Figure 6. Prediction performances (in terms of average pseudo-R^2 obtained with SVM regression) over all q output variables for the three benchmark datasets versus the number of features included in the regression, d (along the regularization path for the sparse penalty or along the feature ranking, depending on the method). Direct prediction performances with the multivariate lasso are also given when possible. For "*Nutrimouse*", the red dot corresponds to the features selected by regularized **CCA**.

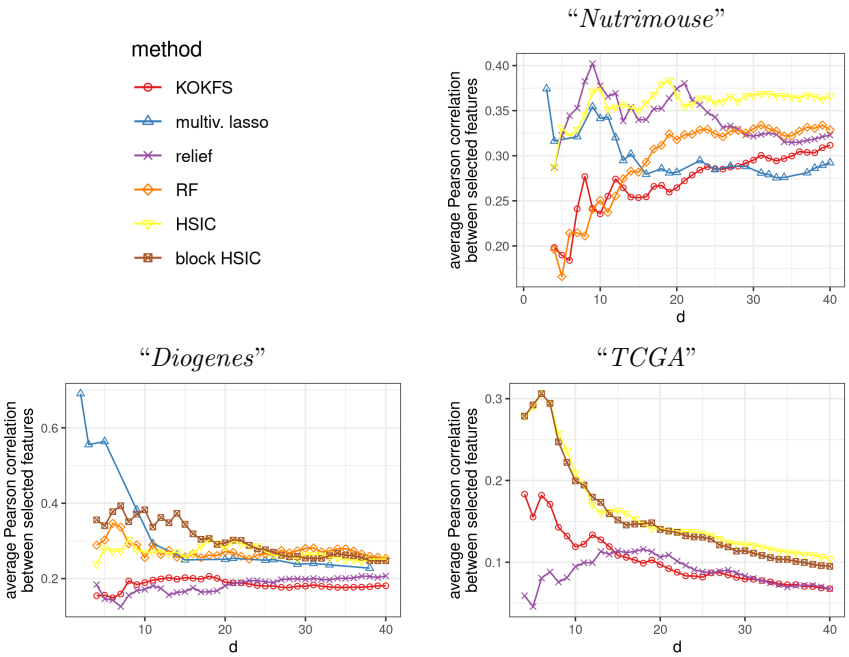

Figure 7. Average (absolute value of) Pearson correlation between selected features for the three benchmark datasets versus the number of features included in the regression, d (along the regularization path for the sparse penalty or along the feature ranking, depending on the method).

increase of availability of Ppe and the fact that government provided support to vulnerable persons were found by both approaches. Similarly, surveillance, the increase of isolation and quarantine facilities and providing international help were also found weakly influential measures by both approaches. The Spearman correlation test also confirmed the significant similarity between the two rankings ($\rho = 0.405$; p-value $= 0.0082$).

Table 4. Averaged running time (in seconds) for selecting 40 variables (10 repetitions). The $\star$ indicates that, unlike other methods that used a single CPU while running, **RF** overloaded the CPU with a 400% CPU usage so its running time is not fully comparable to the others.

Methods	"*Nutrimouse*" $n=40$ $p=120$ $q=21$	"*Diogene*" $n=167$ $p=269$ $q=269$	"*TCGA*" $n=1,194$ $p=655$ $q=9,884$
KOKFS	43.31 ±2.00	413.95 ±3.49	43,502.20 ±35.97
multiv. lasso	3.08 ±0.11	669.85 ±14.19	/ /
Relief	0.49 ±0.06	**0.80** ±0.03	31.99 ±1.01
RF*	3.61 ±0.30	222.43 ±1.73	/ /
HSIC	**0.13** ±0.03	1.34 ±0.03	234.72 ±0.03
block **HSIC**	0.19 ±0.01	0.97 ±0.02	**18.70** ±0.15

DISCUSSION

Overall, the quality of the selection performed with the new method presented in this article is very satisfactory: for the unsupervised case, **UKFS** obtains results that are far better than the simplest and fastest methods based on scoring and results comparable to methods using an *a priori* information on a number of cluster. Note that, in real-life applications, this information is usually not available and, as shown in the experiments, performances of such methods are strongly impeded by an incorrect assumption on this hyperparameter. Since **UKFS** does not require such an *a priori* knowledge, its use is advantageous is very general situations, where there is no strong *a priori* on the dataset structure.

For the kernel output case, the situation is very similar, with an exception: the very simple **Relief** method outperforms, on some datasets, the performance of **KOKFS** but with a performance quality that is very varying from one dataset to the other. A specific feature of **KOKFS** could explain both its improved performance compared to **HSIC** methods (that are also kernel methods) and the varying performance of **Relief**: in **Relief**, the features are selected independently, which leads to a strong redundancy when the input dataset has a large number of features (and this drawback impacts less **Relief** when the number of features is small to moderate). This is the same feature that explains the better performance of **UKFS** (and the other multivariate unsupervised methods) compared to the simplest score based methods (**lapl** and **SPEC**). This explanation is supported by the good tradeoff between performance and redundancy of features obtained for both methods and all datasets for the proposed approach and thus indicates that it is probably best suited when the original number of features is large and that the correlation structure between these features is strong.

Also note that comparison between performances might also be influenced by some decisions made on the method tuning or evaluation. First, as already discussed before, for **MCFS**, **NDFS** and **UDFS**, a proper tuning of the number of clusters might provide a fairer comparison of the performance of these three methods with others. However, in unsupervised

Table 5. Ranking of the government measures as provided by the feature selection approach **KOKFS** to explain the similarity between $(R_t)_t$ time series evolution during the 20 following days (from the most important to the least important). The number in the right column corresponds to the maximum weight for feature j during the learning: $\max_{k=1,\dots,K} w_j^{(T)}(k)$, where $w_j^{(T)}(k)$ is the weight for feature j at the last iteration when using the kth value of the λ_2 grid.

Increase In Medical Supplies And Equipment	1.57
Border Health Check	1.36
Public Transport Restriction	1.29
The Government Provides Assistance To Vulnerable Populations	1.29
Individual Movement Restrictions	1.23
Increase Availability Of Ppe	1.21
Activate Case Notification	1.18
Border Restriction	1.18
Measures To Ensure Security Of Supply	1.17
Port And Ship Restriction	1.13
Activate Or Establish Emergency Response	1.13
National Lockdown	1.12
Crisis Management Plans	1.12
Cordon Sanitaire	1.12
Airport Restriction	1.12
Airport Health Check	1.11
Mass Gathering Cancellation	1.10
Adapt Procedures For Patient Management	1.10
Enhance Laboratory Testing Capacity	1.10
Receive International Help	1.08
Measures For Special Populations	1.08
Closure Of Educational Institutions	1.07
Quarantine	1.07
Police And Army Interventions	1.06
Small Gathering Cancellation	1.06
Special Measures For Certain Establishments	1.06
Actively Communicate With Healthcare Professionals	1.05
Travel Alert And Warning	1.04
Enhance Detection System	1.04
Return Operation Of Nationals	1.04
Educate And Actively Communicate With The Public	1.04
Research	1.04
Work Safety Protocols	1.04
Tracing And Tracking	1.03
Actively Communicate With Managers	1.03
Repurpose Hospitals	1.03
Increase Patient Capacity	1.03
Restricted Testing	1.02
Environmental Cleaning And Disinfection	1.01
Measures For Public Transport	1.01
Isolation Of Cases	1.01
Personal Protective Measures	1.00
Increase Healthcare Workforce	1.00
Provide International Help	1.00
Surveillance	1.00
Increase Isolation And Quarantine Facilities	1.00

setting, it is very difficult to perform, having no ground truth available for that task. Overall, the tuning of all hyperparameters (especially, the regularization parameters of **UKFS**, **KOKFS**, **HSIC** lasso, block **HSIC** lasso and multivariate lasso that are indirectly linked to the number of selected features) is always a bit difficult and this is especially true for **KOKFS** that has two hyper-parameters. A simple heuristic was proposed for the joint tuning of these two hyper-parameters but this impacts the running time and is a direction of improvement for the method.

In addition, the fact that the proposed feature selection method is able to explicitly account for the type of kernel (Gaussian) that is further used in the non linear prediction approach (kernel k-means and SVM) might play a role in the good results of **KOKFS**. This shows that the method is especially well adapted to select features used in subsequent kernel methods, often more adapted to describe relations between samples than the standard Euclidean norm in $\mathbb{R}^p$ again when p is large.

Finally, an important note has to be made on the running times. Both **UKFS** and **KOKFS** have averaged running times, much larger than score based methods (unsupervised case, **lapl** and **SPEC**) or than **Relief** and **HSIC** approaches (kernel output case). This is expected because the fastest methods all deal with features independently. On the opposite, multivariate methods like ours are able to account for colinearities between features and to reduce redundancy in selected features, especially for datasets with very large dimensions, but they do not scale well. **UKFS** and **KOKFS** are particularly sensitive to large sample size (large n as in "*TCGA*"), as most kernel methods. To a lesser extent, they are also impacted by large dimensions of inputs (large p as in "*Carcinom*", "*Koren*", "*Diogene*" and "*TCGA*") because of the need to recompute the full kernel at each iteration step (contrary to **HSIC** methods for instance). As a consequence, the running time is not very good on small to moderate size datasets ("*Nutrimouse*" and "*Diogene*").

However, while accounting for the multivariate aspect of output features, the kernel output method is almost insensitive to large dimensions of outputs (large q), because the output kernel is computed only once. It is thus more efficient for large (input and output) dimensions than multivariate lasso ("*Diogene*") and is the only multivariate method for which the training was not possible for the large sample size and large dimension dataset ("*TCGA*"). Note that the running time of another multivariate feature selection method, the **Autoencoder**, is also very prohibitive (almost 1 hour for the smallest dataset, which is not realistic for computational biology data that are usually large). Note that **Autoencoder** would scale better if run on GPU (only CPU computations were performed for the sake of fair comparison) but the algorithm presented in this work is based on operations that have already been implemented in GPU for modern frameworks such as Pytorch. Therefore, it would benefit from a similar speedup.

Finally, to a lesser extent, the remark on running times can be said for **UDFS**: processing the "*Carcinom*" with this method required more than a day. In conclusion, both **UKFS** and **KOKFS** thus offer a good trade-off with acceptable running times, while accounting for colinearities between features (and for the multivariate aspect of the outputs) but they should not be used for fast screening purpose on large datasets.

CONCLUSION

We have proposed a generic approach for feature selection in kernel methods that proved efficient in various situations, is able to incorporate prior knowledge and to handle various non numeric outputs. This approach should be able to greatly contribute to a better interpretation of omics integration analysis for which kernel methods are already a gold standard.

However, even if the proposed method is computationally efficient compared to more demanding approaches, such as autoencoders, it tends to scale poorly when the number of samples, n, becomes very large. Future work should leverage this problem by investigating the use of Nyström approximation or random Fourier features to speed up the most demanding parts of the algorithm.

ACKNOWLEDGEMENTS

The authors are grateful to the GenoToul bioinformatics platform (INRAE Toulouse, http://bioinfo.genotoul.fr/) and its staff for providing computing facilities.

Conflict of interest statement. None declared.

REFERENCES

1. Schölkopf, B., Tsuda, K., and Vert, J.-P. (2004) Kernel Methods in Computational Biology, MIT Press, London, UK.
2. Duda, R. O., Hart, P. E., and Stork, D. G. (2000) Pattern Classification, Willey-Blackwell, USA 2nd edition.
3. Rapaport, F., Zinovyev, A., Dutreix, M., Barillot, E., and Vert, J.-P. (2007) Classification of microarray data using gene networks. *BMC Bioinformatics,* **8**, 35.
4. Noble, W. S. (2004) Support vector machine applications in computational biology. In Schölkopf, B., Tsuda, K., and Vert, J.-P., (eds.), *Kernel Methods in Computational Biology*, pp. 71–92 MIT Press.
5. Qiu, J., Hue, M., Ben-Hur, A., Vert, J.-P., and Stafford, N. W. (2007) A structural alignment kernel for protein structures. *Bioinformatics,* **23**(9), 1090–1098.
6. Mahé, P. and Vert, J.-P. (2009) Graph kernels based on tree patterns for molecules. *Machine Learning,* **75**, 3–35.
7. Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S., Smola, A. J., and Kriegel, H.-P. (2005) Protein function prediction via graph kernels. *Bioinformatics,* **2005**(21), i47–i56.
8. Speicher, N. K. and Pfeifer, N. (2015) Integrating different data types by regularized unsupervised multiple kernel learning with application to cancer subtype discovery. *Bioinformatics,* **31**(12), i268–i275.
9. Mariette, J. and Villa-Vialaneix, N. (2018) Unsupervised multiple kernel learning for heterogeneous data integration. *Bioinformatics,* **34**(6), 1009–1015.
10. Hofmann, D., Gisbrecht, A., and Hammer, B. (2015) Efficient approximations of robust soft learning vector quantization for non-vectorial data. *Neurocomputing,* **147**, 96–106.
11. Mariette, J., Olteanu, M., and Vialaneix, N. (2017) Efficient interpretable variants of online SOM for large dissimilarity data. *Neurocomputing,* **225**, 31–48.
12. Kwok, J. T. and Tsang, I. W. (2004) The pre-image problem in kernel methods. *IEEE Transactions on Neural Networks,* **15**(6), 1517–1525.
13. Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., and Liu, H. (2018) Feature selection: a data perspective. *ACM Computing Surveys,* **50**(6), 94:1–94:45.
14. Tibshirani, R. (1996) Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, series B,* **58**(1), 267–288.
15. Robnik-Šikonja, M. and Kononenko, I. (2003) Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning,* **53**(1-2), 23–69.
16. Lin, D. and Tang, X. (2006) Conditional infomax learning: an integrated framework for feature extraction and fusion. In Leonardis, A., Bischolf, H., and Pinz, A., (eds.), *Proceedings of European Conference on Computer Vision (ECCV 2006)*, Springer, Berlin, Heidelberg Vol. 3951 of Lecture Notes in Computer Science, pp. 68–82.
17. Yu, E., Sun, J., Li, J., Chang, X., Han, X.-H., and Hauptmann, A. G. (2019) Adaptive semi-supervised feature selection for cross-modal retrieval. *IEEE Transactions on Multimedia,* **21**(5), 1276–1288.
18. Zhao, Z. and Liu, H. (2007) Spectral feature selection for supervised and unsupervised learning. In Ghahramani, Z., (ed.), *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)*, Corvalis, OR, USA: ACM, New York, NY, USA pp. 1151–1157.
19. Cai, D., Zhang, C., and He, X. (2010) Unsupervised feature selection for multi-cluster data. In Rao, B., Krishnapuram, B., Tomkins, A., and Yang, Q., (eds.), *Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD 2010)*, Washington, DC, USA: ACM, New York, NY, USA pp. 333–342.
20. Masaeli, M., Yan, Y., Cui, Y., Fung, G., and Dy, J. G. (2010) Convex principal feature selection. In Parthasarathy, S., Liu, B., Goethals, B., Pei, J., and Kamath, C., (eds.), *Proceedings of the SIAM International Conference on Data Mining (SDM 2010)*, Columbus, Ohio, USA: SIAM pp. 619–628.
21. Li, Z., Yang, Y., Liu, J., Zhou, X., and Lu, H. (2012) Unsupervised feature selection using nonnegative spectral analysis. In *Proceedings of the 26th Conference on Artificial Intelligence (AAAI 2012)* Toronto, Ontario, Canada: pp. 1026–1032.
22. Yang, Y., Shen, H. T., Ma, Z., Huang, Z., and Zhou, X. (2011) $\ell_{2,1}$-norm regularized discriminative feature selection for unsupervised learning. In Walsh, T., (ed.), *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, Barcelona, Spain: AAAI Press pp. 1589–1594.
23. Luo, M., Nie, F., Chang, X., Yang, Y., Hauptmann, A. G., and Zheng, Q. (2018) Adaptive unsupervised feature selection with structure regularization. *IEEE Transactions on Neural Networks and Learning Systems,* **29**(4), 944–956.
24. Balın, M. F., Abid, A., and Zou, J. (2019) Concrete autoencoders: differentiable feature selection and reconstruction. In *Proceedings of the 36th International Conference on Machine Learning (PMLR)* Vol. 97, pp. 444–453.
25. Doquet, G. and Sebag, M. (2019) Agnostic feature selection. In *Proceedings of ECLM-PKDD 2019* Würzburg, Germany: pp. 343–358.
26. Friedman, J. H., Hastie, T., and Tibshirani, R. (2010) Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software,* **33**(1), 1–22.
27. Perrot-Dockès, M., Lévy-Leduc, C., Sansonnet, L., and Chiquet, J. (2018) Variable selection in multivariate linear models with high-dimensional covariance matrix estimation. *Journal of Multivariate Analysis,* **166**, 78–97.
28. Rohart, F., Gautier, B., Singh, A., and Le Cao, K.-A. (2017) mixOmics: an R package for omics feature selection and multiple data integration. *PLoS Computational Biology,* **13**(11), e1005752.
29. González, I., Déjean, S., Martin, P. G., Gonçalves, O., Besse, P., and Baccini, A. (2008) Highlighting relationships between heterogeneous biological data through graphical displays based on regularized canonical correlation analysis. *Journal of Biological Systems,* **17**(2), 173–199.
30. Li, Y., Nan, B., and Zhu, J. (2015) Multivariate sparse group lasso for the multivariate multiple linear regression with an arbitrary group structure. *Biometrics,* **71**(2), 354–363.
31. Perrot-Dockès, M., Lévy-Leduc, C., Chiquet, J., Sansonnet, L., Brégère, M., Étienne, M.-P., Robin, S., and Genta-Jouve, G. (2018) A variable selection approach in the multivariate linear model: an application to LC-MS metabolomics data. *Statistical Applications to Genetics and Molecular Biology,* **17**(5), 20170077.
32. Petković, M., Kocev, D., and Džeroski, S. (2020) Feature ranking for multi-target regression. *Machine Learning,* **109**, 1179–1204.
33. Sechidis, K., Spyromitros-Xioufis, E., and Vlahavas, I. (2019) Information theoretic multi-target feature selection via output space quantization. *Entropy,* **21**(9), 855.
34. Yamada, M., Jitkrittum, W., Sigal, L., Xing, E. P., and Sugiyama, M. (2014) High-dimensional feature selection by feature-wise kernelized Lasso. *Neural Computation,* **26**(1), 185–207.
35. Li, F., Yang, Y., and Xing, E. P. (2005) From lasso regression to feature vector machine. In Weiss, Y., Schölkopf, B., and Platt, J., (eds.), *Advances in Neural Information Processing Systems (Proceedings of NIPS 2005)*, MIT Press Vol. 18, pp. 779–786.
36. Ravikumar, P., Lafferty, J., Liu, H., and Wasserman, L. (2009) Sparse additive models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology),* **71**(5), 1009–1030.
37. Song, L., Smola, A., Gretton, A., Bedo, J., and Borgwardt, K. (2012) Feature selection via dependence maximization. *The Journal of Machine Learning Research,* **13**(47), 1393–1434.
38. Masaeli, M., Fung, G., and Dy, J. G. (2010) From transformation-based dimensionality reduction to feature selection. In Fürnkranz, J. and Joachims, T., (eds.), *Proceedings of International conference on Machine learning (ICML 2010)*, pp. 751–758.
39. Climente-González, H., Azencott, C.-A., Kaski, S., and Yamada, M. (2019) Block HSIC Lasso: model-free biomarker detection for ultra-high dimensional data. *Bioinformatics,* **35**(14), i427–i435.
40. Grandvalet, Y. and Canu, S. (2002) Adaptive scaling for feature selection in SVMs. In Becker, S., Thrun, S., and Obermayer, K., (eds.), *Proceedings of Advances in Neural Information Processing Systems (NIPS 2002)*, MIT Press pp. 569–576.
41. Allen, G. I. (2013) Automatic feature selection via weighted kernels and regularization. *Journal of Computational and Graphical Statistics,* **22**(2), 284–299.
42. Varma, M. and Babu, B. R. (2009) More generality in efficient multiple kernel learning. In Danyluk, A., Bottou, L., and Littman, M., (eds.), *Proceedings of the 26th Annual International Conference on Machine Learning (ICML 2009)*, New York, NY, USAACM: ACM pp. 1065–1072.
43. Bauschke, H. H. and Combettes, P. L. (2011) Convex Analysis and Monotone Operator Theory in Hilbert Spaces, CMS Books in MathematicsSpringer, New York, NY, USA.
44. Parikh, N. and Boyd, S. (2014) Proximal algorithms. *Foundations and Trends® in Optimization,* **1**(3), 127–239.
45. Candès, E. J., Wakin, M. B., and Boyd, S. P. (2008) Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications,* **14**(5-6), 877–905.

46. Barzilai, J. and Borwein, J. M. (1988) Two-point step size gradient methods. *IMA Journal of Numerical Analysis,* **8**(1), 141–148.
47. Gong, P., Zhang, C., Lu, Z., Huang, J., and Ye, J. (2013) A general iterative shrinkage and thresholding algorithm for non-convex regularized optimization problems. In Dasgupta, S. and McAllester, D., (eds.), *Proceedings of the International Conference on Machine Learning (ICML 2013)*, Atlanta, DA, USA: ACM Vol. 23, pp. 37–45.
48. Blumensath, T. and Davies, M. E. (2009) Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis,* **27**(3), 265–274.
49. Attouch, H., Bolte, J., Redont, P., and Soubeyran, A. (2010) Proximal alternating minimization and projection methods for nonconvex problems: an approach based on the Kurdyka-Łojasiewicz inequality. *Mathematics of Operations Research,* **35**(2), 438–457.
50. Brouard, C., Szafranski, M., and d'Alché-Buc, F. (2016) Input Output Kernel Regression: supervised and semi-supervised structured output prediction with operator-valued kernels. *Journal of Machine Learning Research,* **17**, 1–48.
51. Ciliberto, C., Rosasco, L., and Rudi, A. (2016) A consistent regularization approach for structured prediction. In Lee, D., Sugiyama, M., van Luxburg, U., Guyon, I., and Garnett, R., (eds.), *Advances in Neural Information Processing Systems (NIPS 2016)*, MIT Press Vol. 29, pp. 4412–4420.
52. Chen, D., Jacob, L., and Mairal, J. (2019) Biological sequence modeling with convolutional kernel networks. *Bioinformatics,* **35**(18), 3294–3302.
53. Rakotomamonjy, A., Bach, F. R., Canu, S., and Grandvalet, Y. (2008) SimpleMKL. *Journal of Machine Learning Research,* **9**, 2491–2521.
54. Feng, Y., Xiao, J., Zhuang, Y., and Liu, X. (2013) Adaptive unsupervised multi-view feature selection for visual concept recognition. In Lee, K., Matsushita, Y., Rehg, J., and Hu, Z., (eds.), *Computer Vision – ACCV 2012*, Springer Berlin Heidelberg Vol. 7724 of Lecture Notes in Computer Science, pp. 343–357.
55. Hou, C., Nie, F., Li, X., Yi, D., and Wu, Y. (2014) Joint embedding learning and sparse regression: a framework for unsupervised feature selection. *IEEE Transactions on Cybernetics,* **44**(6), 793–804.
56. Schölkopf, B., Smola, A., and Müller, K.-R. (1998) Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation,* **10**(5), 1299–1319.
57. He, X., Cai, D., and Niyogi, P. (2005) Laplacian score for feature selection. In Weiss, Y., Schölkopf, B., and Platt, J., (eds.), *Proceedings of the 18th International Conference on Neural Information Processing Systems (NIPS 2005)*, MIT Press Vol. 18, pp. 507–514.
58. Abid, A., Balin, M. F., and Zou, J. (2019) Concrete autoencoders for differentiable feature selection and reconstruction. In Chaudhuri, K. and Salakhutdinov, R., (eds.), *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, Long Beach, California, USA: PMLR Vol. 97, pp. 444–453.
59. Danon, L., Diaz-Guilera, A., Duch, J., and Arenas, A. (2005) Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment,* **2005**, P09008.
60. Martin, P. G., Guillou, H., Lasserre, F., Déjean, S., Lan, A., Pascussi, J.-M., SanCristobal, M., Legrand, P., Besse, P., and Pineau, T. (2007) Novel aspects of PPARα-mediated regulation of lipid and xenobiotic metabolism revealed through a multrigenomic study. *Hepatology,* **45**(3), 767–777.
61. Carayol, J., Chabert, C., Di Cara, A., Armenise, C., Lefebvre, G., Langin, D., Viguerie, N., Metairon, S., Saris, W. H., Astrup, A., et al. (2017) Protein quantitative trait locus study in obesity during weight-loss identifies a leptin regulator. *Nature Communications,* **8**, 2084.
62. Armenise, C., Lefebvre, G., Carayol, J., Bonnel, S., Bolton, J., Di Cara, A., Gheldof, N., Descombes, P., Langin, D., Saris, W. H., et al. (2017) Transcriptome profiling from adipose tissue during a low-calorie diet reveals predictors of weight and glycemic outcomes in obese, nondiabetic subjects. *The American Journal of Clinical Nutrition,* **106**(3), 736–746.
63. Capitaine, L., Genuer, R., and Tiébaut, R. (2021) Random forests for high-dimensional longitudinal data. *Statistical Methods in Medical Research,* **30**(1), 166–184.
64. Alt, H. and Godau, M. (1993) Computing the Frechet distance between two polygonal curves. *International Journal of Computational Geometry & Applications,* **5**(01n02), 75–91.
65. Nye, T. M., Tang, X., Weyenberg, G., and Yoshida, R. (2017) Principal component analysis and the locus of the Fréchet mean in the space of phylogenetic trees. *Biometrika,* **104**(4), 901–922.
66. Haug, N., Geyrhofer, L., Londei, A., Dervic, E., Desvars-Larrive, A., Loreto, V., Pinior, B., Thurner, S., and Klimek, P. (2020) Ranking the effectiveness of worldwide COVID-19 government interventions. *Nature Human Behaviour,* **4**(4), 1303–1312.