Overview

A survey on multi-output regression

Hanen Borchani,[1]* Gherardo Varando,[2] Concha Bielza[2] and Pedro Larrañaga[2]

In recent years, a plethora of approaches have been proposed to deal with the increasingly challenging task of multi-output regression. This study provides a survey on state-of-the-art multi-output regression methods, that are categorized as problem transformation and algorithm adaptation methods. In addition, we present the mostly used performance evaluation measures, publicly available data sets for multi-output regression real-world problems, as well as open-source software frameworks. © 2015 John Wiley & Sons, Ltd.

How to cite this article:
WIREs Data Mining Knowl Discov 2015, 5:216–233. doi: 10.1002/widm.1157

INTRODUCTION

Multi-output regression, also known in the literature as multi-target,[1–5] multi-variate,[6–8] or multi-response[9,10] regression, aims to simultaneously predict multiple real-valued output/target variables. When the output variables are binary, the learning problem is called multi-label classification.[11–13] However, when the output variables are discrete (not necessarily binary), the learning problem is referred to as multi-dimensional classification.[14]

Several applications for multi-output regression have been studied. They include ecological modeling to predict multiple target variables describing the condition or quality of the vegetation,[3] chemometrics to infer concentrations of several analytes from multivariate calibration using multivariate spectral data,[15] prediction of the audio spectrum of wind noise (represented by several sound pressure variables) of a given vehicle component,[16] real-time prediction of multiple gas tank levels of the Linz Donawitz converter gas system,[17] simultaneous estimation of different biophysical parameters from remote sensing images,[18] channel estimation through the prediction of several received signals,[19] and so on.

In spite of their different backgrounds, these real-world applications give rise to many challenges such as missing data (i.e., when some feature/target values are not observed), the presence of noise typically due to the complexity of the real domains, and most importantly, the multivariate nature and the compound dependencies between the multiple feature/target variables. In dealing with these challenges, it has been proven that multi-output regression methods yield to a better predictive performance, in general, when compared against the single-output methods.[15–17] Multi-output regression methods provide as well the means to effectively model the multi-output datasets by considering not only the underlying relationships between the features and the corresponding targets but also the relationships between the targets, guaranteeing thereby a better representation and interpretability of the real-world problems.[3,18] A further advantage of the multi-target approaches is that they may produce simpler models with a better computational efficiency.[3]

Existing methods for multi-output regression can be categorized as: (1) *problem transformation methods* (also known as local methods) that transform the multi-output problem into independent single-output problems each solved using a single-output regression algorithm, and (2) *algorithm adaptation methods* (also known as global or big-bang methods) that adapt a specific single-output method (such as decision trees and support vector

*Correspondence to: hanen@cs.aau.dk

[1]Machine Intelligence Group, Department of Computer Science, Aalborg University, Aalborg, Denmark

[2]Computational Intelligence Group, Departamento de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, Madrid, Spain

Conflict of interest: The authors have declared no conflicts of interest for this article.

machines) to directly handle multi-output data sets. Algorithm adaptation methods are deemed to be more challenging because they usually aim not only to predict the multiple targets but also to model and interpret the dependencies among these targets.

Note here that the *multi-task learning* (MTL) problem[20–24] is related to the multi-output regression problem: it also aims to learn multiple related tasks (i.e., outputs) at the same time. Commonly investigated issues in MTL include modeling task relatedness and the definition of similarity between jointly learned tasks, feature selection, and certainly, the development of efficient algorithms for learning and predicting several tasks simultaneously using different approaches, such as clustering, kernel regression, neural networks, tree and graph structures, Bayesian model, and so forth. The main difference between multi-output regression and multi-task problems is that tasks may have different training sets and/or different descriptive features, in contrast to the target variables that share always the same data and/or descriptive features.

The article is organized as follows. In *Problem Transformation Methods* section, the state-of-the-art multi-output regression approaches are presented according to the categorization as problem transformation and algorithm adaptation methods. In *Discussion* section, we provide a theoretical comparison of the different presented approaches. In *Performance Evaluation Measures* section, we discuss evaluation measures, and publicly available data sets for multi-output regression learning problems are given in *Data Sets* section. *Open-Source Software Frameworks* section describes the open-source software frameworks available for multi-output regression methods, and finally, *Conclusion* section sums up the article with some conclusions and possible lines for future research.

MULTI-OUTPUT REGRESSION

Let us consider the training data set D of N instances containing a value assignment for each variable $X_1, \ldots, X_m$, $Y_1, \ldots, Y_d$, i.e., $D = \{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \ldots, (\mathbf{x}^{(N)}, \mathbf{y}^{(N)})\}$. Each instance is characterized by an input vector of m descriptive or predictive variables $\mathbf{x}^{(l)} = \left(x_1^{(l)}, \ldots, x_j^{(l)}, \ldots, x_m^{(l)}\right)$ and an output vector of d target variables $\mathbf{y}^{(l)} = \left(y_1^{(l)}, \ldots, y_i^{(l)}, \ldots, y_d^{(l)}\right)$, with $i \in \{1, \ldots, d\}$, $j \in \{1, \ldots, m\}$, and $l \in \{1, \ldots, N\}$. The task is to learn a multi-target regression model from D consisting of finding a function h that assigns to each instance, given by the vector $\mathbf{x}$, a vector $\mathbf{y}$ of d target values:

$$h : \Omega_{X_1} \times \ldots \times \Omega_{X_m} \rightarrow \Omega_{Y_1} \times \ldots \times \Omega_{Y_d}$$
$$\mathbf{x} = (x_1, \ldots, x_m) \mapsto \mathbf{y} = (y_1, \ldots, y_d),$$

where Ω_{X_j} and Ω_{Y_i} denote the sample spaces of each predictive variable X_j, for all $j \in \{1, \ldots, m\}$, and each target variable Y_i, for all $i \in \{1, \ldots, d\}$, respectively. Note that, all target variables are considered to be continuous here. The learned multi-target model will be used afterward to simultaneously predict the values $\left\{\hat{\mathbf{y}}^{(N+1)}, \ldots, \hat{\mathbf{y}}^{(N')}\right\}$ of all target variables of the new incoming unlabeled instances $\left\{\mathbf{x}^{(N+1)}, \ldots, \mathbf{x}^{(N')}\right\}$.

Throughout this section, we provide a survey on state-of-the-art multi-output regression learning methods categorized as problem transformation methods (*Problem Transformation Methods* section) and algorithm adaptation methods (*Algorithm Adaptation Methods* section).

Problem Transformation Methods

These methods are mainly based on transforming the multi-output regression problem into single-target (ST) problems, then building a model for each target, and finally concatenating all the d predictions. The main drawback of these methods is that the relationships among the targets are ignored, and the targets are predicted independently, which may affect the overall quality of the predictions.

Recently, Spyromitros-Xioufis et al.[4] proposed to extend well-known multi-label classification transformation methods to deal with the multi-output regression problem and to also model the target dependencies. In particular, they introduced two novel approaches for multi-target regression, *multi-target regressor stacking* (MTRS) and *regressor chains* (RC), inspired by popular and successful multi-label classification approaches.

As discussed in Spyromitros-Xioufis et al.,[4] only approaches based on single labels (such as the typical binary relevance, stacked generalization-based methods, and classifier chains) can be straightforwardly adapted to multi-output regression by using a regression instead of a classification algorithm. Multi-label approaches, based on either pairs of labels or sets of labels paradigms, are generally not transferable to multi-target regression problems. However, the random k-labelsets (RAkEL) method has been the inspiration for a new problem transformation method recently proposed by Tsoumakas et al.[5] Their method creates new output variables as random linear combinations of k original output variables. Next, a user-specified multi-output algorithm is applied to

predict the new variables, and finally, the original targets are recovered by inverting the random linear transformation.

In this section, we present state-of-the-art multi-output regression methods based on problem transformation, namely, ST method, MTRS, RC, and multi-output support vector regression (MO-SVR).

Single-Target Method

In the baseline ST method,[4] a multi-target model is comprised of d single-target models, each trained on a transformed training set $D_i = \left\{ \left(\mathbf{x}_1^{(1)}, y_i^{(1)} \right), \ldots, \left(\mathbf{x}^{(N)}, y_i^{(N)} \right) \right\}, i \in \{1, \ldots, d\}$, to predict the value of a single-target variable Y_i. In this way, the target variables are predicted independently and potential relationships between them cannot be exploited. The ST method is also known as binary relevance in the literature.[13]

As the multi-target prediction problem is transformed into several single-target problems, any off-the-shelf ST regression algorithm can be used. For instance, Spyromitros-Xioufis et al.[4] used four well-known regression algorithms, namely, ridge regression,[25] SVR machines,[26] regression trees,[27] and stochastic gradient boosting.[28]

Moreover, Hoerl and Kennard[25] proposed the *separate ridge regression* method to deal with multi-variate regression problems. It consists of performing a separate ridge regression of each individual target Y_i on the predictor variables $\mathbf{X} = (X_1, \ldots, X_m)$. The regression coefficient estimates $\hat{a}_{ij}$, with $i \in \{1, \ldots, d\}$ and $j \in \{1, \ldots, m\}$, are the solution to a penalized least squares criterion:

$$\left\{ \hat{a}_{ij} \right\}_{j=1}^{m} = \arg \min_{\{a_j\}_{j=1}^{m}} \left\{ \sum_{l=1}^{N} \left[y_i^{(l)} - \sum_{j=1}^{m} a_j x_j^{(l)} \right]^2 \right\} + \lambda_i \sum_{j=1}^{m} a_j^2, \; i \in \{1, \ldots, d\},$$

where $\lambda_i > 0$ represents the ridge parameters.

Multi-Target Regressor Stacking

The MTRS method[4] is inspired by[29] where *stacked generalization*[30] was used to deal with multi-label classification. MTRS training is a two-stage process. First, d ST models are learned as in ST. However, instead of directly using these models for prediction, MTRS includes an additional training stage where a second set of d meta-models are learned, one for each target Y_i, $i \in \{1, \ldots, d\}$.

Each meta-model is learned on a transformed training set $D_i^* = \left\{ \left(\mathbf{x}^{*(1)}, y_i^{(1)} \right), \ldots, \left(\mathbf{x}^{*(N)}, y_i^{(N)} \right) \right\}$, where $\mathbf{x}^{*(l)} = \left(x_1^{(l)}, \ldots, x_N^{(l)}, \hat{y}_1^{(l)}, \ldots, \hat{y}_d^{(l)} \right)$ is a transformed input vector consisting of the original input vector of the training set augmented by predictions (or estimates) of their target variables yielded by the first-stage models. In fact, MTRS is based on the idea that a second-stage model is able to correct the prediction of a first-stage model by using information about the predictions of other first-stage models.

The predictions for a new instance $\mathbf{x}^{(N+1)}$ are obtained by generating first-stage models inducing the estimated output vector $\hat{\mathbf{y}}^{(N+1)} = \left(\hat{y}_1^{(N+1)}, \ldots, \hat{y}_d^{(N+1)} \right)$, then applying the second-stage models on the transformed input vector $\mathbf{x}^{*(N+1)} = \left(x_1^{(N+1)}, \ldots, x_m^{(N+1)}, \hat{y}_1^{(N+1)}, \ldots, \hat{y}_d^{(N+1)} \right)$ to produce the final estimated multi-output targets $\hat{\hat{\mathbf{y}}}^{(N+1)} = \left(\hat{\hat{y}}_1^{(N+1)}, \ldots, \hat{\hat{y}}_d^{(N+1)} \right)$.

Regressor Chains

The RC method[4] is inspired by the recent multi-label chain classifiers.[31] RC is another problem transformation method, based on the idea of chaining single-target models. The training of RC consists of selecting a random chain (i.e., permutation) of the set of target variables, then building a separate regression model for each target following the order of the selected chain.

Assuming that the ordered set or the full chain $C = (Y_1, Y_2, \ldots, Y_d)$ is selected, the first model is only concerned with the prediction of Y_1. Then, subsequent models for $Y_{i, s.t.\, i>1}$ are trained on the transformed data sets $D_i^* = \left\{ \left(\mathbf{x}_i^{*(1)}, y_i^{(1)} \right), \ldots, \left(\mathbf{x}_i^{*(N)}, y_i^{(N)} \right) \right\}$, where $\mathbf{x}_i^{*(l)} = \left(x_1^{(l)}, \ldots, x_m^{(l)}, y_1^{(l)}, \ldots, y_{i-1}^{(l)} \right)$ is a transformed input vector consisting of the original input vector of the training set augmented by the actual values of all previous targets in the chain. Spyromitros-Xioufis et al.[4] then introduced the regressor chain corrected (RCC) method that uses cross-validation estimates instead of the actual values in the transformation data step.

However, the main problem with the RC and RCC methods is that they are sensitive to the selected chain ordering. To avoid this problem, and alike,[31] Spyromitros-Xioufis et al.[4] proposed a set of regression chain models with differently ordered chains: if the number of distinct chains was less than 10, they created exactly as many models as the number of distinct label chains; otherwise, they selected 10 chains randomly. The resulting approaches are called *ensemble of regressor chains* (ERC) and *ensemble of regressor chains corrected* (ERCC).

Multi-Output Support Vector Regression

Zhang et al.[32] presented a MO-SVR approach based on problem transformation. It builds a multi-output model that takes into account the correlations between all the targets using the vector virtualization method. Basically, it extends the original feature space and expresses the multi-output problem as an equivalent single-output problem, so that it can then be solved using the single-output least squares SVR machines (LS-SVR) algorithm.

In particular, Zhang et al.[32] used a binary representation to express $\mathbf{y}^{(l)}$ with vectors $\mathbf{I}_i$ of length d such that only the ith element representing the ith output takes the value 1 and all the remaining elements are zero. In this way, for any instance $(\mathbf{x}^{(l)}, \mathbf{y}^{(l)})$, d virtual samples are built by feature vector virtualization as follows:

$$\left(\mathbf{x}^{(l)}, \mathbf{y}^{(l)}\right) \rightarrow \begin{matrix} \left(\mathbf{I}_1, \mathbf{x}^{(l)}, y_1^{(l)}\right) \\ \cdots \\ \left(\mathbf{I}_d, \mathbf{x}^{(l)}, y_d^{(l)}\right). \end{matrix}$$

This yields, a new data set $D_i = \left\{\left(\left(\mathbf{I}_i, \mathbf{x}^{(l)}\right), y_i^{(l)}\right)\right\}$, with $i \in \{1, \ldots, d\}$ and $l \in \{1, \ldots, N\}$, in the extended feature space. The solution follows directly from solving a set of linear equations using extended LS-SVR, where the objective function f to be minimized is defined as follows:

$$f = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{C}\sum_{l=1}^{N}\sum_{i=1}^{d} e_i^{(l)^2}$$

$$\text{s.t. } y_i^{(l)} = \mathbf{w}^T \phi\left(\mathbf{I}_i, \mathbf{x}^{(l)}\right) + \mathbf{I}_i\, \mathbf{b} + e_i^{(l)},$$

where $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_d)$ defines the weights, $\phi(\cdot)$ is a nonlinear transformation to the feature space, and $\mathbf{b} = (b_1, \ldots, b_d)^T$ is the bias vector. C is the trade-off factor used to balance the strengths of the Vapnik-Chervonenkis dimension and the loss, and $e_i^{(l)}$ is the fitting error for each instance in the data set D_i.

Algorithm Adaptation Methods

These methods are based on the idea of simultaneously predicting all the targets using a single model that is able to capture all dependencies and internal relationships between them. This actually has several advantages over problem transformation methods: it is easier to interpret a single multi-target model than many single-target models and it ensures better predictive performance especially when the targets are correlated.[3,6,10] In this section, we present state-of-the-art multi-output regression methods defined as extensions of several standard learning algorithms including statistical methods, support vector machines, kernel methods, regression trees, and classification rules.

Statistical Methods

The statistical approaches are considered as the first attempt to deal with simultaneously predicting multiple real-valued targets. They aim to take advantage of correlations between the target variables in order to improve predictive accuracy compared with the traditional procedure of doing individual regressions of each target variable on the common set of predictor variables.

Izenman[33] proposed reduced-rank regression which places a rank of constraint on the matrix of estimated regression coefficients. Considering the following regression model:

$$y_i = \sum_{j=1}^{m} a_{ij} x_j + \varepsilon_i,\ i \in \{1, \ldots, d\},$$

the aim is to determine the coefficient matrix $\tilde{\mathbf{A}}_r \in \mathbb{R}^{d\times m}$ of rank $r \leq \min\{m, d\}$ such that

$$\tilde{\mathbf{A}}_r = \arg \min_{\text{rank}(\mathbf{A})=r} E\left[(\mathbf{y} - \mathbf{A}\mathbf{x})^T \sum^{-1} (\mathbf{y} - \mathbf{A}\mathbf{x})\right]$$

with estimated error $\sum = E(\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T)$, where $\boldsymbol{e}^T = \{\varepsilon_1, \ldots, \varepsilon_d\}$. The above equation is then solved as $\tilde{\mathbf{A}}_r = \mathbf{B}_r\hat{\mathbf{A}}$, where $\hat{\mathbf{A}} \in \mathbb{R}^{d\times m}$ is the matrix of the ordinary least squares (OLS) estimates and the reduced-rank shrinking matrix $\mathbf{B}_r \in \mathbb{R}^{d\times d}$ is given by

$$\mathbf{B}_r = \mathbf{T}^{-1}\mathbf{I}_r\mathbf{T},$$

where $\mathbf{I}_r = \text{diag}\{1\,(i \leq r)\}_{i=1}^{d}$ and $\mathbf{T}$ is the canonical co-ordinate matrix that seeks to maximize the correlation between the d-vector $\mathbf{y}$ and the m-vector $\mathbf{x}$.

Later, Brown and Zidek[7] presented a multi-variate version of the Hoerl-Kennard ridge regression rule and proposed the estimator $\hat{\beta}^*(\mathbf{K})$:

$$\hat{\beta}^*(\mathbf{K}) = \left(\mathbf{x}^T\mathbf{x} \otimes \mathbf{I}_d + \mathbf{I}_m \otimes \mathbf{K}\right)^{-1}\left(\mathbf{x}^T\mathbf{x} \otimes \mathbf{I}_d\right), \hat{\beta}$$

where $\mathbf{K}(d\times d) > 0$ is the ridge matrix. $\otimes$ denotes the usual Kronecker product, and $\hat{\beta}, \hat{\beta}^*$ are $(md \times 1)$ vectors of estimators of $\beta = (\beta_1, \ldots, \beta_m)^T$, where $\beta_1, \ldots, \beta_m$ are each $(1 \times d)$ row vectors of β. $\hat{\beta}$ represents the maximum likelihood estimator of β corresponding to $\mathbf{K} = \mathbf{0}$.

Furthermore, van der Merwe and Zidek[34] introduced the filtered canonical y-variate regression (FICYREG) method defined as a generalization to the multi-variate regression problem of the James-Stein estimator. The estimated coefficient matrix $\tilde{\mathbf{A}} \in \mathbb{R}^{d\times m}$ takes the form

$$\tilde{\mathbf{A}} = \mathbf{B}_f\hat{\mathbf{A}},$$

where $\hat{\mathbf{A}} \in \mathbb{R}^{d\times m}$ is the matrix of OLS estimates. The shrinking matrix $\mathbf{B}_f \in \mathbb{R}^{d\times d}$ is given by $\mathbf{B}_f = \hat{\mathbf{T}}^{-1}\mathbf{F}\hat{\mathbf{T}}$, where $\hat{\mathbf{T}}$ is the sample canonical co-ordinate matrix and $\mathbf{F} = \text{diag}\{f_1, \ldots, f_d\}$ represents the canonical co-ordinate shrinkage factors $\{f_i\}_{i=1}^{d}$ that depend on the number of targets d, the number of predictor variables m, and the corresponding sample-squared canonical correlations $\{\hat{c}_i^2\}_{i=1}^{d}$:

$$f_i = \left(\hat{c}_i^2 - \frac{m-d-1}{N}\right) \Big/ \hat{c}_i^2 \left(1 - \frac{m-d-1}{N}\right)$$
$$\text{and} \quad f_i \leftarrow \max\{0, f_i\}.$$

In addition, one of the most prominent approaches for dealing with the multi-output regression problem is the curds and whey (C&W) method proposed by Breiman and Friedman in Ref 6. Basically, given d targets $\mathbf{y} = (y_1, \ldots, y_d)^T$ with separate least squares regressions $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_d)^T$, where $\overline{\mathbf{y}}$ and $\overline{\mathbf{x}}$ are the sample means of $\mathbf{y}$ and $\mathbf{x}$, respectively, a more accurate predictor $\tilde{y}_i$ of each y_i is obtained using a linear combination

$$\tilde{y}_i = \overline{y}_i + \sum_{k=1}^{d} b_{ik}\left(\hat{y}_k - \overline{y}_k\right),\ i \in \{1, \ldots, d\},$$

of the OLS predictors

$$\hat{y}_i = \overline{y}_i + \sum_{j=1}^{m} \hat{a}_{ij}\left(x_j - \overline{x}_j\right), \quad \text{s.t.}$$
$$\{\hat{a}_{ij}\}_{j=1}^{m}$$
$$= \arg \min_{\{a_j\}_{j=1}^{m}} \left[\sum_{l=1}^{N}\left(y_i^{(l)} - \overline{y}_i - \sum_{j=1}^{m} a_j\left(x_j^{(l)} - \overline{x}_j\right)\right)^2\right],$$

rather than with the least squares themselves. Here $\hat{a}_{ij}$ are the estimated regression coefficients, and b_{ik} can be regarded as shrinking parameters that transform the vector-valued OLS estimates $\hat{\mathbf{y}}$ to the biased estimates $\tilde{\mathbf{y}}$, and are determined by the C&W procedure, which is a form of multi-variate shrinking. In fact, the estimates of the matrix $\mathbf{B} = [b_{ik}] \in \mathbb{R}^{d\times d}$ take the form of $\mathbf{B} = \mathbf{T}^{-1}\mathbf{S}\mathbf{T}$, where $\mathbf{T}$ is the $d \times d$ matrix whose rows are the response canonical co-ordinates maximizing the correlations between $\mathbf{y}$ and $\mathbf{x}$, and $\mathbf{S} = \text{diag}(s_1, \ldots, s_d)$ is a diagonal shrinking matrix. To estimate $\mathbf{B}$, C&W starts by transforming ($\mathbf{T}$), shrinking (i.e., multiplying by $\mathbf{S}$), and then transforming back ($\mathbf{T}^{-1}$).

More recently, Similä and Tikka[10] investigated the problem of input selection and shrinkage in multi-response linear regression. They presented a simultaneous variable selection (SVS) method called L_2-SVS, where the importance of an input in the model is measured by the L_2-norm of the regression coefficients associated with the input. To solve the L_2-SVS, $\mathbf{W}$, the $m \times d$ matrix of regression coefficients, is estimated by minimizing the error sum of squares subject to a sparsity constraint as follows:

$$\min_{\mathbf{W}} f(\mathbf{W}) = \frac{1}{2}\|\mathbf{y} - \mathbf{x}\mathbf{W}\|_F^2 \ \text{ subject to } \ \sum_{j=1}^{m}\|\mathbf{w}_j\|_2 \le r,$$

where the subscript F denotes the Frobenius norm, i.e., $\|\mathbf{B}\|_F^2 = \sum_{ij} b_{ij}^2$. The factor $\|\mathbf{w}_j\|_2$ is a measure of the importance of the jth input in the model, and r is a free parameter that controls the amount of shrinkage that is applied to the estimate.

If the value of $r \ge 0$ is large enough, the optimal $\mathbf{W}$ is equal to the OLS solution, whereas small values of r impose a row-sparse structure on $\mathbf{W}$, which means that only some of the inputs are effective in the estimate.

Abraham et al.[35] coupled linear regressions and quantile mapping to both minimize the residual errors and capturing the joint (including nonlinear) relationships among variables. The method was tested on bivariate and trivariate output spaces showing that it is able to reduce residual errors while keeping the joint distribution of the output variables.

Multi-Output Support Vector Regression

Traditionally, SVR is used with a single-output variable. It aims to determine the mapping between the input vector $\mathbf{x}$ and the single output y_i from a given training data set D_i, by finding the regressor $\mathbf{w} \in \mathbb{R}^{m\times 1}$ and the bias term $b \in \mathbb{R}$ that minimize

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{l=1}^{N} L\left(y^{(l)} - \left(\phi\left(\mathbf{x}^{(l)}\right)^T\mathbf{w} + b\right)\right),$$

where $\phi(\cdot)$ is a nonlinear transformation to a higher dimensional Hilbert space H, and C is a parameter chosen by the user that determines the trade-off between the regularization and the error reduction term, first and second addend, respectively. L is a Vapnick ε-insensitive loss function, which is equal to 0 for $|y^{(l)} - (\phi(\mathbf{x}^{(l)})^T\mathbf{w} + b)| < \varepsilon$ and to $|y^{(l)} - (\phi(\mathbf{x}^{(l)})^T\mathbf{w} + b)| - \varepsilon$ for $|y^{(l)} - (\phi(\mathbf{x}^{(l)})^T\mathbf{w} + b)| \ge \varepsilon$. The solution ($\mathbf{w}$ and b) is induced by a linear combination of the training set in the transformed space with an absolute error equal to or greater than ε.

Hence, in order to deal with the multi-output case, single-output SVR can be easily applied independently to each output (see *Multi-Output Support Vector Regression* in Problem Transformation Methods section). Because it exhibits the serious drawback of not taking into account the possible correlations between outputs however, several approaches have

been proposed to extend traditional SVR in order to better manage the multi-output case. In general, this consists of minimizing

$$\frac{1}{2}\sum_{i=1}^{d}\|\mathbf{w}_i\|^2 + C\sum_{l=1}^{N} L\left(\mathbf{y}^{(l)} - \left(\phi\left(\mathbf{x}^{(l)}\right)^T \mathbf{W} + \mathbf{b}\right)\right),$$

where the $m \times d$ matrix $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_d)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_d)^T$.

For instance, Vazquez and Walter[36] extended SVR by considering the so-called Cokriging[37] method, which is a multi-output version of Kriging that exploits the correlations due to the proximity in the space of factors and outputs. In this way, with an appropriate choice of covariance and cross-covariances models, the authors showed that multi-output SVR yields better results than an independent prediction of the outputs.

Sánchez-Fernández et al.[19] introduced a generalization of SVR. The so-called multiregressor SVR (M-SVR) is based on an iterative reweighted least squares (IRWLS) procedure that iteratively estimates the weights $\mathbf{W}$ and the bias parameters $\mathbf{b}$ until convergence, i.e., until reaching a stationary point where there is no more improvement of the considered loss function.

Similarly, Brudnak[38] developed a vector-valued SVR by extending the notions of the estimator, loss function and regularization functional from the scalar-valued case; and Tuia et al.[18] proposed a multi-output SVR method by extending the single-output SVR to multiple outputs while maintaining the advantages of a sparse and compact solution using a cost function. Later, Deger et al.[39] adapted Tuia et al.'s[18] approach to tackle the problem of reflectance recovery from multispectral camera output, and proved through their empirical results that it has the advantages of being simpler and faster to compute than a scalar-valued based method.

In Ref 40, Cai and Cherkassky described a new methodology for regression problems, combining Vapnik's SVM+ regression method and the MTL setting. SVM+, also known as learning with structured data, extends the standard SVM regression by taking into account the group information available in the training data. The SVM+ approach learns a single regression model using information on all groups, whereas the proposed SVM+MTL approach learns several related regression models, specifically one model for each group.

In Ref 41, Liu et al. considered the output space as a Riemannian submanifold to incorporate its geometric structure into the regression process, and they proposed a locally linear transformation (LLT) mechanism to define the loss functions on the output manifold. Their proposed approach, called LLT-SVR, starts by identifying the k-nearest neighbors of each output using the Euclidean distance, then obtains local coordinate systems, and finally trains the regression model by solving a convex quadratic programming problem.

Moreover, Han et al.[17] dealt with the prediction of the gas tank level of the Linz Donawitz converter gas system using a multi-output least squares SVR. They considered both the single-output and the combined-output fitting errors. In model solving, a full-rank equation is given to determine the required parameters during training using an optimization based on particle swarm[42] (an evolutionary computation method).

Xu et al.[43] recently proposed another approach to extend least squares SVR to the multi-output case. The so-called multi-output LS-SVR (MLS-SVR) then solves the problem by finding the weights $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_d)$ and the bias parameters $\mathbf{b} = (b_1, \ldots, b_d)^T$ that minimize the following objective function:

$$\min_{\mathbf{W}\in\mathbb{R}^{n_h\times d}, \mathbf{b}\in\mathbb{R}^d} F(\mathbf{W}, \Xi) = \frac{1}{2}\operatorname{trace}\left(\mathbf{W}^T\mathbf{W}\right) + \gamma\,\frac{1}{2}\operatorname{trace}\left(\Xi^T\Xi\right),$$
$$\text{s.t. } \mathbf{Y} = \mathbf{Z}^T\mathbf{W} + \operatorname{repmat}\left(\mathbf{b}^T, N, 1\right) + \Xi,$$

where $\mathbf{Z} = \left(\varphi\left(\mathbf{x}^{(1)}\right), \varphi\left(\mathbf{x}^{(2)}\right), \ldots, \varphi\left(\mathbf{x}^{(N)}\right)\right) \in \mathbb{R}^{n_h\times N}$, $\varphi : \mathbb{R}^m \to \mathbb{R}^{n_h}$ is a mapping to some higher dimensional Hilbert space H with n_h dimensions. The function repmat defined over a $1 \times d$ matrix $\mathbf{b}$ repmat$(\mathbf{b}^T, N, 1)$ creates a large block matrix consisting of an $N \times 1$ tiling of copies of $\mathbf{b}$. $\Xi = \left(\xi_1, \xi_2, \ldots, \xi_d\right) \in \mathbb{R}_+^{N\times d}$ is a matrix consisting of slack variables, and $\gamma \in \mathbb{R}^+$ is a positive real regularized parameter.

Kernel Methods

A study of vector-valued learning with kernel methods was started by Micchelli and Pontil,[9] where they analyzed the regularized least squares from the computational point of view. They also analyzed the theoretical aspects of reproducing kernel Hilbert spaces (RKHS) in the range-space of the estimator, and they generalized the representer theorem for Tikhonov regularization to the vector-valued setting.

Baldassarre et al.[44] later studied a class of regularized kernel methods for multi-output learning which are based on filtering the spectrum of the kernel matrix. They considered methods also including Tikhonov regularization as a special case, and alternatives such as vector-valued extensions of squared loss function (L_2) boosting and other iterative

schemes. In particular, they claimed that Tikhonov regularization could be seen as a low-pass filtering applied to the kernel matrix. The idea is thus to use different kinds of spectral filtering, defining regularized matrices that in general do not have interpretation as penalized empirical risk minimization.

In addition, Evgeniou and Pontil[45] considered the learning of an average task simultaneously with small deviations for each task, and Evgeniou et al. extended their earlier results in Ref 46 by developing indexed kernels with coupled regularization functionals.

Álvarez et al.[47] reviewed at length kernel methods for vector-valued functions, focusing especially on regularization and Bayesian prospective, connecting the two points of view. They provided a large collection of kernel choices, focusing on separable kernels, sum of separable kernels and further extensions as kernels to learn divergence-free and curl-free vector fields.

Multi-Target Regression Trees

Multi-target regression trees, also known as multivariate regression trees (MRTs) or multi-objective regression trees, are trees able to predict multiple continuous targets at once. Multi-target regression trees have two main advantages over building a separate regression tree for each target.[48] First, a single multi-target regression tree is usually much smaller than the total size of the individual single-target trees for all variables, and, second, a multi-target regression tree better identifies the dependencies between the different target variables.

One of the first approaches proposed for dealing with multi-target regression trees was proposed by De'ath.[49] He presented an extension of the univariate recursive partitioning method (CART)[50] to the multi-output regression problem. Hence, the so-called MRTs are built following the same steps as CART, i.e., starting with all instances in the root node, then iteratively finding the optimal split and partitioning the leaves accordingly until a pre-defined stopping criterion is reached. The only difference from CART is the redefinition of the impurity measure of a node as the sum of squared error over the multi-variate response:

$$\sum_{l=1}^{N}\sum_{i=1}^{d}\left(y_i^{(l)}-\overline{y}_i\right)^2,$$

where $y_i^{(l)}$ denotes the value of the output variable Y_i for the instance l and $\overline{y}_i$ denotes the mean of Y_i in the node. Each split is selected to minimize the sum of squared error. Finally, each leaf of the tree can be characterized by the multi-variate mean of its instances, the number of instances at the leaf, and its defining feature values. De'ath[49] claimed that MRT also inherits characteristics of univariate regression trees: they are easy to construct and the resulting groups are often simple to interpret; they are robust to the addition of pure noise response and/or feature variables; they automatically detect the interactions between variables, and they handle missing values in feature variables with minimal loss of information.

Struyf and Džeroski[48] proposed a constraint-based system for building multi-objective regression trees (MORTs). It includes both size and accuracy constraints, so that the user can trade off size (and thus interpretability) for accuracy by either specifying maximum tree size or minimum accuracy. Their approach consists of first building a large tree using the training set, then pruning it in a second step to satisfy the user constraints. This has the advantage that the tree can be stored in the inductive database and used for answering inductive queries with different constraints.

Basically, MORTs are constructed with a standard top-down induction algorithm,[50] and the heuristic used for selecting the attribute tests in the internal nodes is the *intra-cluster variation* summed over the subsets (or clusters) induced by the test. Intra-cluster variation is defined as $N \cdot \sum_{i=1}^{d} \mathrm{Var}\left(Y_i\right)$ with N the number of instances in the cluster, d number of target variables, and $\mathrm{Var}(Y_i)$ the variance of the target variable Y_i in the cluster. Minimizing intra-cluster variation produces homogeneous leaves, which in turn results in accurate predictions.

In addition, Appice and Džeroski[2] presented an algorithm, named multi-target stepwise model tree induction (MTSMOTI), for inducing multi-target model trees in a stepwise fashion. Model trees are decision trees whose leaves contain linear regression models that predict the value of a single continuous target variable. Based on the stepwise model tree induction algorithm,[51] MTSMOTI induces the model tree top-down by choosing at each step to either partition the training space (split nodes) or introduce a regression variable in the set of linear models to be associated with leaves. In this way, each leaf of such a model tree contains several linear models, each predicting the value of a different target variable Y_i.

Kocev et al.[3] explored and compared two approaches for dealing with multi-output regression problem: first, learning a model for each output separately (i.e., multiple regression trees) and, second, learning one model for all outputs simultaneously (i.e., a single multi-target regression tree). In order to improve predictive performance, Kocev et al.[52] also considered two ensemble learning techniques, namely, *bagging*[27] and *random forests*[53] of regression trees and multi-target regression trees.

Ikonomovska et al.[54] proposed an incremental multi-target model tree algorithm, referred to as FIMT-MT, for simultaneous modeling of multiple continuous targets from time changing data streams. FIMT-MT extends an incremental single-target model tree by adopting the principles of the predictive clustering methodology in the split selection criterion. In the tree leaves, linear models are separately computed for each target using an incremental training of perceptrons.

Stojanova et al.[55] developed the NCLUS algorithm for modeling nonstationary autocorrelation in network data by using predictive clustering trees (i.e., decision trees with a hierarchy of clusters: the top-node corresponds to one cluster containing all data, which is recursively partitioned into smaller clusters while moving down the tree). NCLUS is a top–down induction algorithm that recursively partitions the set of nodes based on the average values of variance reduction and autocorrelation measure computed over the set of all target variables.

More recently, a similar work has been proposed by Appice et al.[56] They dealt with the problem of modeling nonstationary spatial autocorrelation of multi-variate geophysical data streams by using interpolative clustering trees (i.e., tree-structured models where a split node is associated with a cluster and a leaf node with a single predictive model for the multiple target variables). Their proposed time-evolving method is also based on a top–down induction algorithm that makes use of variance reduction and spatial autocorrelation measure computed over the target variables.

Levatić et al.[57] addressed the task of semi-supervised learning for multi-target regression and proposed a self-training approach using a random forest of predictive clustering trees. The main feature of self-training is that it iteratively uses its own most reliable predictions in the learning process. The most reliable predictions are selected in this case using a threshold on the reliability scores, which are computed as the average of the normalized per-target standard deviations.

Rule Methods

Aho et al.[58] presented a new method for learning rule ensembles for multi-target regression problems and simultaneously predicting multiple numeric target attributes. The so-called *FItted Rule Ensemble* (FIRE) algorithm transcribes an ensemble of regression trees into a large collection of rules, then an optimization procedure is used to select the best (and much smaller) subset of these rules and determine their respective weights.

More recently, Aho et al.[1] extended the FIRE algorithm by combining rules with simple linear functions in order to increase the predictive accuracy. Thus, FIRE optimizes the weights of rules and linear terms with a gradient-directed optimization algorithm. Given an unlabeled example $\mathbf{x}$, the resulting rule ensemble is a vector $\hat{\mathbf{y}}$ consisting of the values of all target variables:

$$\hat{\mathbf{y}} = f(\mathbf{x}) = w_0\,\mathbf{avg} + \sum_{k=1}^{R} \mathbf{w}_k \mathbf{r}_k(\mathbf{x}) + \sum_{i=1}^{d}\sum_{j=1}^{m} \mathbf{w}_{ij}\mathbf{x}_{ij},$$

where $w_0 \in \mathbb{R}$ is the baseline prediction, $\mathbf{avg}$ is the constant vector whose components are the average values for each of the targets, and R defines the number of considered rules. Hence, the first sum is the contribution of the R rules: each rule $\mathbf{r}_k$ is a vector function that gives a constant prediction for each of the targets if it covers the example $\mathbf{x}$, or returns a zero vector otherwise; and the weights $\mathbf{w}_k$ are optimized by a gradient-directed optimization algorithm. The double sum is the contribution of optional $m \times d$ linear terms. In fact, a linear term $\mathbf{x}_{ij}$ is a vector that corresponds to the influence of the jth numerical descriptive variable X_j on the ith target variable Y_i, i.e., its ith component is equal to X_j, whereas all other components are zero:

$$\mathbf{x}_{ij} = \left(0, \ldots, \underbrace{0}_{\text{i}-1}, \underbrace{x_j}_{\text{i}}, \underbrace{0}_{\text{i}+1}, \ldots, 0\right).$$

Finally, the values of all weights $\mathbf{w}_{ij}$ are also determined using a gradient-directed optimization algorithm that depends on a gradient threshold τ. Thus, the optimization procedure is repeated using different values of τ in order to find a set of weights with the smallest validation error.

Table 1 summarizes the reviewed multi-output regression algorithms.

DISCUSSION

Note that, even though the ST method is a simple approach, it does not imply simpler models. In fact, exploiting relationships among the output variables could be used to improve the precision or reduce computational costs as explained in what follows.

First, let us point out that some transformation algorithms fail to properly exploit the multi-output relationships, and therefore they may be considered as ST methods. For instance, this is the case of RC using linear regression as base models, namely, OLS or ridge estimators of the coefficients.

TABLE 1 | Summary of Multi-Output Regression Methods

	Method	References	Year
Problem transformation methods	Single target	Spyromitros-Xioufis et al.[4]	2012
	Random linear target combinations	Tsoumakas et al.[5]	2014
	Separate ridge regression	Hoerl and Kennard[25]	1970
	Multi-target regressor stacking	Spyromitros-Xioufis et al.[4]	2012
	Regressor chains	Spyromitros-Xioufis et al.[4]	2012
	Multi-output SVR	Zhang et al.[32]	2012
Algorithm adaptation methods	Statistical methods	Izenman[33]	1975
		van der Merwe and Zidek[34]	1980
		Brown and Zidek[7]	1980
		Breiman and Friedman[6]	1997
		Similä and Tikka[10]	2007
		Abraham et al.[35]	2013
	Multi-output SVR	Brudnak[38]	2006
		Cai et al.[40]	2009
		Deger et al.[39]	2012
		Han et al.[17]	2012
		Liu et al.[41]	2009
		Sanchez et al.[19]	2004
		Tuia et al.[18]	2011
		Vazquez and Walter[36]	2003
		Xu et al.[43]	2013
	Kernel methods	Baldassarre et al.[44]	2012
		Evgeniou and Pontil[45]	2004
		Evgeniou et al.[46]	2005
		Micchelli and Pontil[9]	2005
		Álvarez at al.[47]	2012
	Multi-target regression trees	De'ath[49]	2002
		Appice and Džeroski[2]	2007
		Kocev et al.[3]	2009
		Kocev et al.[52]	2012
		Struyf and Džeroski[48]	2006
		Ikonomovska et al.[54]	2011
		Stojanova et al.[55]	2012
		Appice et al.[56]	2014
		Levatić et al.[57]	2014
	Rule methods	Aho et al.[58]	2009
		Aho et al.[1]	2012

Lemma 1. *RC with linear regression is an ST if OLS or ridge regression is used as base models.*

Proof. See Appendix A.

To our knowledge, Lemma 1 is valid just for linear regression. However, it presents an example of the fact that, in some cases, intuitions behind a model could be misleading. In particular, when problem transformations methods are used in combination with ensemble methods (e.g., ERC and ERCC), the advantages of the multi-output approach could be hard to understand and interpret.

In addition, statistical methods and multi-output SVR (MO-SVR) are methods that mainly rely on the idea of embedding of the output-space. They assume that the space of the output variables could

be described using a sub-space of lower dimensions than $\mathbb{R}^d$ (e.g., Izenman,[33] Brudnak,[38] and Liu et al.[41]). There are several reasons to adopt this embedding:

- When $m < d$. In this case, an embedding is certain.[38]
- When we have a prior knowledge on the output-space structure, e.g., *spatial* relationship among the output variables.[36]
- When we assume a linear model with a nonfull rank matrix of coefficients.[33,34]
- When we assume a manifold structure for the output-space.[41]

Such an embedding implies a series of advantages. First of all, a more compact representation of the output space is achieved. Second, in the case of linear models, it assures correct estimations of ill-posed problems.[7,33,44] Third, it may improve the predictive performance of the considered methods.[41] Moreover, in the case of MO-SVR and kernel methods with a large number of input variables, computations could become very costly, so exploiting output dependencies permits to reduce them.[38]

Statistical methods could be considered as direct extensions of the ST linear regression, while MO-SVR and kernel methods present the merits of dealing with nonlinear regression functions, and therefore they are more general. We could then ascribe statistical methods in the *modeling* tradition of statistics, while MO-SVR, kernel methods, rule methods and multi-target regression trees rather belong to the *algorithmic* branch of statistics or to the machine learning community (see Breiman[59]).

Predictive Performance

Considering the model's predictive performance as a comparison criterion, the benefits of using MTRS and RC (or ERC and the corrected versions) instead of the baseline ST approach are not so clear. In fact, in Ref 4, an extensive empirical comparison of these methods is presented, and the results show that ST methods outperform several variants of MTRS and ERC. This fact is especially notable in the straightforward applications. In particular, the benefits of MTRS and RC methods appear to derive uniquely from the randomization process (e.g., due to the order of the chain) and from the ensemble model (e.g., ERC).

Statistical methods could improve notably the performance with respect to a baseline ST regression but only if specific assumptions are fulfilled, i.e., a relation among outputs truly exists, and a linear output–output relationship (in addition to a linear input–output relationship) is verified. Otherwise, using these statistical models could produce a detriment of the predictive performance. In particular, if we assume that the $d \times m$ matrix of regression coefficients has a reduced rank $r < \min(d, m)$, when in reality it possess a full-rank, then we are obviously wrongly estimating the relationship and we lose some information.

MO-SVR and kernel methods are, in general, designed to achieve a good predictive performance where linearity cannot be assumed. It is interesting to notice that some of the MO-SVR methods are basically designed with the following goals: (1) speeding up computations, (2) obtaining a sparser representation (avoiding the use of the same support vector for several times) compared to the ST approach,[38] and (3) keeping more or less the same error rates as the ST approach. On the contrary, Liu et al.[41] implementation is only based on improving the predictive performance. The authors also advocate that their method should be implemented in every regression algorithm because it guarantees to find an optimal local basis for computing distances in the output manifolds.

Finally, multi-target regression trees and rule methods are also based on finding simpler multi-output models, that usually achieve good predictive results (i.e., comparable with ST approach).

Computational Complexity

For a large number of output variables, all problem transformation methods face the challenging problems of either solving a large number of ST problems (e.g., ST, MTRS, and RC) or a single large problem (e.g., LS-SVR[32]). Nevertheless, note that ST and some implementations of RC could be speeded up in the training and/or prediction phases using a parallel computation (see *Open Source Software Frameworks* section and Appendix B).

Using ST with kernel methods as a base model may also lead to compute the same kernel over the same points more than once. In this case, it is computationally more efficient to consider multi-output kernels and thus avoid redundant computations.

Multi-target regression trees and rule methods are also designed to be more competitive from the point of view of computational and memory complexity, especially compared to their ST counterparts.

Representation and Interpretability

Algorithm adaptation methods, relying on ST models, do not provide a description of the relationships among the multiple output variables. Those methods

are interpretable as long as the number of outputs is not intractable, otherwise, it is extremely difficult to analyze each model and retrieve information about the relationships between the different variables.

Statistical methods provide a similar representation as ST regression models (each output is a linear combination of inputs), the main difference is that the subset of independent and sufficient outputs could be discovered. In some cases (e.g., LASSO penalty estimations), the estimated model could be represented as a graph since the matrix of the regression coefficients tends to be sparse.

Kernel and MO-SVR methods suffer from the same problem as in the single-output SVR. In fact, their model interpretation is not straightforward since the input space is transformed. The gain in predictive performance is usually paid in terms of readability.

Multi-target regression trees and rule methods build human-readable predictive models. They are hence considered as the most interpretable multi-output models (if not coupled with ensemble methods), clearly illustrating which input variables are relevant and important in the prediction of a given group of outputs.

PERFORMANCE EVALUATION MEASURES

In this section, we introduce the performance evaluation measures used to assess the behavior of learned models when applied to an unseen or test data set of size N_{test}, and thereby to assess the multi-output regression methods used for model induction. Let $\mathbf{y}^{(l)}$ and $\hat{\mathbf{y}}^{(l)}$ be the vectors of the actual and predicted outputs for $\mathbf{x}^{(l)}$, respectively, and $\overline{\mathbf{y}}$ and $\overline{\hat{\mathbf{y}}}$ be the vectors of averages of the actual and predicted outputs, respectively. Besides measuring the computing times,[1,3,17,52] the mostly used evaluation measures for assessing multi-output regression models are:

- The average correlation coefficient (aCC)[3,43,52]:

$$\begin{aligned}\text{aCC} &= \frac{1}{d}\sum_{i=1}^{d}\text{CC} \\ &= \frac{1}{d}\sum_{i=1}^{d}\frac{\sum_{l=1}^{N_{\text{test}}}\left(y_i^{(l)}-\overline{y}_i\right)\left(\hat{y}_i^{(l)}-\overline{\hat{y}}_i\right)}{\sqrt{\sum_{l=1}^{N_{\text{test}}}\left(y_i^{(l)}-\overline{y}_i\right)^2\sum_{l=1}^{N_{\text{test}}}\left(\hat{y}_i^{(l)}-\overline{\hat{y}}_i\right)^2}}\end{aligned} \tag{1}$$

- The average relative error[43]:

$$\text{a}\delta = \frac{1}{d}\sum_{i=1}^{d}\delta = \frac{1}{d}\sum_{i=1}^{d}\frac{1}{N_{\text{test}}}\sum_{l=1}^{N_{\text{test}}}\frac{\left|y_i^{(l)}-\hat{y}_i^{(l)}\right|}{y_i^{(l)}} \tag{2}$$

- The mean-squared error (MSE)[38,40,48]:

$$\text{MSE} = \sum_{i=1}^{d}\frac{1}{N_{\text{test}}}\sum_{l=1}^{N_{\text{test}}}\left(y_i^{(l)}-\hat{y}_i^{(l)}\right)^2 \tag{3}$$

- The average root-mean-squared error (aRMSE)[3,17,18,39,52]:

$$\begin{aligned}\text{aRMSE} &= \frac{1}{d}\sum_{i=1}^{d}\text{RMSE} \\ &= \frac{1}{d}\sum_{i=1}^{d}\sqrt{\frac{\sum_{l=1}^{N_{\text{test}}}\left(y_i^{(l)}-\hat{y}_i^{(l)}\right)^2}{N_{\text{test}}}}\end{aligned} \tag{4}$$

- The average relative root-mean-squared error (aRRMSE)[1,4,5,52]:

$$\begin{aligned}\text{aRRMSE} &= \frac{1}{d}\sum_{i=1}^{d}\text{RRMSE} \\ &= \frac{1}{d}\sum_{i=1}^{d}\sqrt{\frac{\sum_{l=1}^{N_{\text{test}}}\left(y_i^{(l)}-\hat{y}_i^{(l)}\right)^2}{\sum_{l=1}^{N_{\text{test}}}\left(y_i^{(l)}-\overline{y}_i\right)^2}}\end{aligned} \tag{5}$$

- The model size[1,3,52]: defined, e.g., as the total number of nodes in trees (or the total number of rules) in multi-target regression trees (or rule methods).

Note that, the different described estimated errors are computed as the sum/average over all the separately computed errors for each target variable. This allows to calculate the model performance across multiple targets, which may potentially have distinct ranges. In such cases, the use of a normalization operator could be useful in order to obtain a normalized error values for each target, prior to averaging. The normalization is usually done by dividing each target variable by its standard deviation or by re-scaling its range. The re-scaling factor could be either determined by the data/application at hand (i.e., some prior knowledge), or by the type of the used evaluation measures. For instance, when using MSE or RMSE, a reasonable choice would be to scale each target variable by its standard deviation. Relative measures, such

as RRMSE, automatically re-scale the error contributions of each target variable, and hence, there might be no need here to use an extra normalization operator.

DATA SETS

Despite the many interesting applications of multi-target regression, there are only a few publicly available data sets. There follows a brief description of those data sets, which are then summarized in Table 2 including details about the number of instances (represented as training/testing or total number of instances/CV where cross-validation (CV) is applied for the evaluation process), the number of targets, and the number of features.

- Solar Flare[60]: data set for predicting how often three potential types of solar flare—common, moderate, and severe (i.e., $d = 3$)—occur in a 24-h period. The prediction is performed from the input information of 10 feature variables describing active regions on the sun.
- Water Quality[61]: data set for inferring chemical from biological parameters of river water quality. The data are provided by the Hydrometeorological Institute of Slovenia and cover the six-year period from 1990 to 1995. It includes the measured values of 16 different chemical parameters and 14 bioindicator taxa.
- OES97 and OES10[4]: data gathered from the annual Occupation Employment Survey compiled by the US Bureau of Labor Statistics for the years 1997 (OES97) and 2010 (OES10). Each row provides the estimated number of full-time equivalent employees across many employment types for a specific metropolitan area. The input variables are a randomly sequenced subset of employment types, and the targets ($d = 16$) are randomly selected from the entire set of categories above the 50% threshold.
- ATP1d and ATP7d[4]: data sets of airline ticket prices where the rows are sequences of time-ordered observations over several days. The input variables include details about the flights (such as prices, stops, and departure date), and the six target variables are the minimum prices observed over the next 7 days for six flight preferences (namely, any airline with any number of stops, any airline nonstop only, Delta Airlines, Continental Airlines, AirTran Airlines, and United Airlines).
- RF1 and RF2[4]: the river flow domain is a temporal prediction task designed to test predictions on the flows in a river network for 48 h in the future at specific locations. The data sets were obtained from the US National Weather Service and include hourly flow observations for eight sites in the Mississippi River network in the United States from September 2011 to September 2012. The RF1 and RF2 data sets contain a total of 64 and 576 predictor variables respectively, describing lagged flow observations from 6, 12, 18, 24, 36, 48, and 60 h in the past.
- EDM[62]: data set for the electrical discharge machining (EDM) domain in which the workpiece surface is machined by electrical discharges occurring in the gap between two electrodes—the tool and the workpiece. The aim here is to predict the two target variables, gap control and flow control, using 16 input variables representing mean values and deviations of the observed quantities of the considered machining parameters.

Note here that all the above data sets can be downloaded from http://users.auth.gr/espyromi/datasets.html.

- Polymer[43]: the Polymer test plant data set includes 10 input variables, measurements of controlled variables in a polymer processing plant (temperatures, feed rates, etc.), and 4 target variables which are measures of the output of that plant.

It is available from ftp://ftp.cis.upenn.edu/pub/ungar/chemdata/.

TABLE 2 | Multi-Target Regression Data Sets

Data Set	Instances	Features	Targets
Solar Flare[60]	1389/CV	10	3
Water Quality[61]	1060/CV	16	14
OES97[4]	323/CV	263	16
OES10[4]	403/CV	298	16
ATP1d[4]	201/136	411	6
ATP7d[4]	188/108	411	6
RF1[4]	4108/5017	64	8
RF2[4]	4108/5017	576	8
EDM[62]	154/CV	16	2
Polymer[43]	41/20	10	4
Forestry-Kras[63]	60607/CV	160	2
Soil quality[64]	1945/CV	142	3

- Forestry-Kras[63]: data set for the prediction of forest stand height and canopy cover for the Kras region in Western Slovenia. This data set contains 2 target variables representing forest properties (i.e., vegetation height and canopy cover) and 160 explanatory input variables derived from Landsat satellite imagery data. The data are available upon request from the authors.
- Soil quality[64]: data set for predicting the soil quality from agricultural measures. It consists of 145 variables, of which 3 are target variables (the abundances of Acari and Collembolans, and Shannon-Wiener biodiversity), and 142 are input variables that describe the field where the microarthropod sample was taken and mainly include agricultural measures (such as crops planted, packing, tillage, fertilizer, and pesticide use). The data are available upon request from the authors.

OPEN-SOURCE SOFTWARE FRAMEWORKS

We present now a brief summary of available implementations of some multi-output regression algorithms.

Problem Transformation Methods

Single target models, RC and all the methods described in Spyromitros-Xioufis et al.[4] have been implemented as an extension of MULAN[65] (developed by Tsoumakas, Spyromitros-Xioufis, and Vilcek), which also consists of an extension of the widely used WEKA software.[66] MULAN is available as a library, thus there is no graphical user or command line interfaces.

Similar to MULAN, there is also the MEKA software[67] (developed by Read and Reutemann), which is defined as a multi-label extension to WEKA. It mainly focuses on multi-label algorithms, but incorporates as well some multi-output algorithms. MEKA presents a graphical user interface similar to WEKA and it is very easy to use for nonexperts.

The main advantage of both MEKA and MULAN is that problem transformation methods can be coupled with any ST algorithm implemented in the WEKA library. Moreover, MULAN could be coupled with MOA[68] framework for data stream mining or integrated in ADAMs[69] framework for scientific workflow management.

Furthermore, problem transformation methods, such as ST, MTRS, and RC could be easily implemented in R,[70] and it is possible to use as well the **parallel**[70] package (included in R, version 2.14.0) to speed up computations using parallel computing (see Appendix B for a simple example of an R source code of a parallel ST implementation).

Statistical Methods

The **glmnet**[71] R package offers the possibility of learning multi-target linear models with penalized maximum likelihood. In particular, using this package, it is possible to perform LASSO, ridge or mixed penalty estimation of the coefficients.

Multi-Target Regression Trees

MRTs[49] are available through the R package **mvpart**,[72] which is an extension of the **rpart**[73] package that implements the CART algorithm. The mvpart package is not currently available in CRAN but its older versions could be retrieved.

Additional implementation of multi-target regression trees algorithms could be also found in the CLUS system,[74] focused on decision trees and rules induction. Among others, the CLUS system implements the predictive clustering framework and includes multi-objective regression trees.[48] Moreover, MULAN includes a wrapper implementation of CLUS, supporting hence multi-objective random forest.

CONCLUSION

In this study, the state of the art of multi-output regression is thoroughly surveyed, presenting details of the main approaches that have been proposed in the literature, and including a theoretical comparison of these approaches in terms of predictive performance, computational complexity, and representation and interpretability. Moreover, we have presented the most often used performance evaluation measures, as well as the publicly available data sets for multi-output regression problems, and we have provided a summary of the related open-source software frameworks.

To the best of our knowledge, there is no other review paper addressing the challenging problem of multi-output regression. An interesting line of future work would be to perform a comparative experimental study of the different approaches presented here on the publicly available data sets to round out this review. Another interesting extension of this review is to consider different categorizations of the described multi-output regression approaches, such as grouping them based on how they model the relationships among the multiple target variables.

ACKNOWLEDGMENTS

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness through the Cajal Blue Brain project (C080020-09; the Spanish partner of the Blue Brain initiative from EPFL), the TIN2013-41592-P project, and the Regional Government of Madrid through the S2013/ICE-2845-CASI-CAM-CM project.

REFERENCES

1. Aho T, Ženko B, Džeroski S, Elomaa T. Multi-target regression with rule ensembles. *J Mach Learn Res* 2009, 373:2055–2066.
2. Appice A, Džeroski S. Stepwise induction of multi-target model trees. In: *Proceedings of the Eighteenth European Conference on Machine Learning*, Warsaw, Poland, 2007, 502–509. Springer Verlag.
3. Kocev D, Džeroski S, White MD, Newell GR, Griffioen P. Using single- and multi-target regression trees and ensembles to model a compound index of vegetation condition. *Ecol Model* 2009, 220:1159–1168.
4. Spyromitros-Xioufis E, Groves W, Tsoumakas G, Vlahavas I. Multi-label classification methods for multi-target regression, arXiv preprint arXiv:1211.6581, 2012, 1159–1168. Cornell University Library.
5. Tsoumakas G, Spyromitros-Xioufis E, Vrekou A, and Vlahavas I. Multi-target regression via random linear target combinations. In: *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, Nancy, France, 2014, 225–240. Springer Verlag.
6. Breiman L, Friedman JH. Predicting multivariate responses in multiple linear regression. *J R Stat Soc Series B Stat Methodol* 1997, 59:3–54.
7. Brown PJ, Zidek JV. Adaptive multivariate ridge regression. *Ann Stat* 1980, 8:64–74.
8. Haitovsky Y. On multivariate ridge regression. *Biometrika* 1987, 74:563–570.
9. Micchelli CA, Pontil M. On learning vector-valued functions. *Neural Comput* 2005, 17:177–204.
10. Similä T, Tikka J. Input selection and shrinkage in multiresponse linear regression. *Comput Stat Data Anal* 2007, 52:406–422.
11. Madjarov G, Kocevb D, Gjorgjevikja D, Džeroskib S. An extensive experimental comparison of methods for multi-label learning. *Pattern Recogn* 2012, 45:705–727.
12. Tsoumakas G, Katakis I. Multi-label classification: an overview. *Int J Data Warehouse Min* 2007, 3:1–13.
13. Zhang M, Zhou Z. A review on multi-label learning algorithms. *IEEE Trans Knowl Data Eng* 2014, 26:1819–1837.
14. Bielza C, Li G, Larrañaga P. Multi-dimensional classification with Bayesian networks. *Int J Approx Reason* 2011, 52:705–727.
15. Burnham AJ, MacGregor JF, Viveros R. Latent variable multivariate regression modeling. *Chemometr Intell Lab* 1999, 48:167–180.
16. Kuznar D, Mozina M, BratkoI I. Curve prediction with kernel regression. In: *Proceedings of the ECML/PKDD 2009 Workshop on Learning from Multi-Label Data*, Bled, Slovenia, 2009, 61–68.
17. Han Z, Liu Y, Zhao J, Wang W. Real time prediction for converter gas tank levels based on multi-output least square support vector regressor. *Control Eng Pract* 2012, 20:1400–1409.
18. Tuia D, Verrelst J, Alonso L, Pérez-Cruz F, Camps-Valls G. Multioutput support vector regression for remote sensing biophysical parameter estimation. *IEEE Geosci Remote Sens Lett* 2011, 8:804–808.
19. Sánchez-Fernández M, de-Prado-Cumplido M, Arenas-Garca J, Pérez-Cruz F. SVM multiregression for nonlinear channel estimation in multiple-input multiple-output systems. *IEEE Trans Signal Process* 2004, 52:2298–2307.
20. Baxter J. A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Mach Learn* 1997, 28:7–39.
21. Caruana R. Multitask learning. *Mach Learn* 1997, 28:41–75.
22. Ben-David S, Schuller R. Exploiting task relatedness for multiple task learning. In: *Proceedings of the Sixteenth Annual Conference on Learning Theory*, Washington, DC, 2003, 567–580. Springer Verlag.
23. Jalali A, Sanghavi S, Ruan C, Ravikumar PK. A dirty model for multi-task learning. In: *Proceedings of the Advances in Neural Information Processing Systems 23*, Vancouver, Canada, 2010, 964–972.
24. Marquand AF, Williams SCR, Doyle OM, Rosa MJ. Full Bayesian multi-task learning for multi-output brain decoding and accommodating missing data. In: *Proceedings of the 2014 International Workshop on Pattern Recognition in Neuroimaging*, Tübingen, Germany, 2014, 1–4. IEEE Press.

25. Hoerl AE, Kennard RW. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics* 1970, 12:55–67.
26. Drucker H, Burges CJC, Kaufman L, Smola A, Vapnik V. Support vector regression machines. In: *Proceedings of the Advances in Neural Information Processing Systems 9*, Denver, CO, 1997, 155–161.
27. Breiman L. Bagging predictors. *Mach Learn* 1997, 24:123–140.
28. Friedman JH. Stochastic gradient boosting. *Comput Stat Data Anal* 2002, 38:367–378.
29. Godbole S, Sarawagi S. Discriminative methods for multi-labeled classification. In: *Proceedings of the Eighth Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Sydney, Australia, 2004, 22–30. Springer Verlag.
30. Wolpert DH. Stacked generalization. *Neural Netw* 1992, 5:241–259.
31. Read J, Pfahringer B, Holmes G, Frank E. Classifier chains for multi-label classification. *Mach Learn* 2011, 85:333–359.
32. Zhang W, Liu X, Ding Y, Shi D. Multi-output LS-SVR machine in extended feature space. In: *Proceedings of the 2012 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, Tianjin, China, 2012, 130–134.
33. Izenman AJ. Reduced-rank regression for the multivariate linear model. *J Multivar Anal* 1975, 5:248–264.
34. van der Merwe A, Zidek JV. Multivariate regression analysis and canonical variates. *Can J Stat* 1980, 8:27–39.
35. Abraham Z, Tan P, Perdinan P, Winkler J, Zhong S, Liszewska M. Position preserving multi-output prediction. In: *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, Prague, Czech Republic, 2013, 320–335. Springer Verlag.
36. Vazquez E, Walter E. Multi-output support vector regression. In: *Proceedings of the Thirteen IFAC Symposium on System Identification*, Rotterdam, The Netherlands, 2003, 1820–1825.
37. Chilès JP, Delfiner P. *Geostatistics: Modeling Spatial Uncertainty*. Wiley Series in Probability and Statistics. Wiley; 1999.
38. Brudnak M. Vector-valued support vector regression. In: *Proceedings of the 2006 International Joint Conference on Neural Networks*, Vancouver, Canada, 2006, 1562–1569. IEEE Press.
39. Deger F, Mansouri A, Pedersen M, Hardeberg JY. Multi- and single-output support vector regression for spectral reflectance recovery. In: *Proceedings of the Eighth International Conference on Signal Image Technology and Internet Based Systems*, Sorrento, Italy, 2012, 139–148. IEEE Press.
40. Cai F, Cherkassky V. SVM+ regression and multi-task learning. In: *Proceedings of the 2009 International Joint Conference on Neural Networks*, Atlanta, GA, 2009, 418–424. IEEE Press.
41. Liu G, Lin Z, Yu Y. Multi-output regression on the output manifold. *Pattern Recogn* 2009, 42: 2737–2743.
42. Kennedy J, Eberhart R. Particle swarm optimization. In: *Proceedings of IEEE International Conference on Neural Networks*, Perth, Western Australia, 1995, 2–8. IEEE Press.
43. Xu S, An X, Qiao X, Zhu L, Li L. Multi-output least-squares support vector regression machines. *Pattern Recogn Lett* 2013, 34:1078–1084.
44. Baldassarre L, Rosasco L, Barla A, Verri A. Multi-output learning via spectral filtering. *Mach Learn* 2012, 87:259–301.
45. Evgeniou T, Pontil M. Regularized multi-task learning. In: *Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Seattle, WA, 2004, 109–117. ACM Press.
46. Evgeniou T, Micchelli CA, Pontil M. Learning multiple tasks with kernel methods. *J Mach Learn Res* 2005, 6:615–637.
47. Álvarez MA, Rosasco L, Lawrence ND. Kernels for vector-valued functions: a review. *Found Trends Mach Learn* 2012, 4:195–266.
48. Struyf J, Džeroski S. Constraint based induction of multi-objective regression trees. In: *Proceedings of the Fifth International Workshop on Knowledge Discovery in Inductive Databases*, Berlin, Germany, 2006, 222–233. Springer Verlag.
49. De'ath G. Multivariate regression trees: a new technique for modeling species-environment relationships. *Ecology* 2002, 83:1105–1117.
50. Breiman L, Friedman JH, Stone CJ, Olshen RA. *Classification and Regression Trees*. Chapman & Hall/CRC; 1984.
51. Malerba D, Esposito F, Ceci M, Appice A, Pontil M. Top-down induction of model trees with regression and splitting nodes. *IEEE Trans Pattern Anal Mach Intell* 2004, 26:612–625.
52. Kocev D, Vens C, Struyf J, Džeroski S. Tree ensembles for predicting structured outputs. *Pattern Recogn* 2012, 46:817–833.
53. Breiman L. Random forests. *Mach Learn* 2001, 45:5–32.
54. Ikonomovska E, Gama J, Džeroski S. Incremental multi-target model trees for data streams. In: *Proceedings of the 2011 ACM Symposium on Applied Computing*, Taichung, Taiwan, 2011, 988–993. ACM.
55. Stojanova D, Ceci M, Appice A, Džeroski S. Network regression with predictive clustering trees. *Data Min Knowl Disc* 2012, 25:378–413.
56. Appice A, Malerba D. Leveraging the power of local spatial autocorrelation in geophysical interpolative clustering. *Data Min Knowl Disc* 2014, 28:1266–1313.

57. Levatić J, Ceci M, Kocev D, Džeroski S. Semi-supervised learning for multi-target regression. In: *Proceedings of the Third International Workshop on New Frontiers In Mining Complex Patterns*, Nancy, France, 2014, 110–123. Springer Verlag.
58. Aho T, Ženko B, Džeroski S. Rule ensembles for multi-target regression. In: *Proceedings of the Ninth IEEE International Conference on Data Mining*, Miami, FL; 2009, 21–30. IEEE Press.
59. Breiman L. Statistical modeling: the two cultures. *Stat Sci* 2001, 16:199–231.
60. Bache K, Lichman M. *UCI Machine Learning Repository*. University of California, School of Information and Computer Sciences, Irvine, CA, 2013. Available at: http://archive.ics.uci.edu/ml. (Accessed May 10, 2015).
61. Džeroski S, Demšar D, Grbović J. Predicting chemical parameters of river water quality from bioindicator data. *Appl Intell* 2000, 13:7–17.
62. Karalic A, Bratko I. First order regression. *Mach Learn* 1997, 26:147–176.
63. Stojanova D, Panov P, Gjorgjioski V, Kobler A, Džeroski S. Estimating vegetation height and canopy cover from remotely sensed data with machine learning. *Ecol Inform* 2010, 5:256–266.
64. Demšar D, Džeroski S, Larsen T, Struyfc J, Axelsenb J, Pedersenb MB, Kroghb PH. Using multi-objective classification to model communities of soil microarthropods. *Ecol Model* 2006, 191:131–143.
65. Tsoumakas G, Spyromitros-Xioufis E, Vilcek J, Vlahavas I. Mulan: a Java library for multi-label learning. *J Mach Learn Res* 2011, 12:2411–2414.
66. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The WEKA data mining software: an update. *ACM SIGKDD Explor Newsl* 2009, 11:10–18.
67. Read J, Reutemann P. MEKA: a multi-label extension to WEKA. Available at: http://meka.sourceforge.net/. (Accessed May 10, 2015).
68. Bifet A, Holmes G, Kirkby R, Pfahringer B. MOA: massive online analysis. *J Mach Learn Res* 2010, 11:1601–1604.
69. Reutemann P, Vanschoren J. Scientific workflow management with ADAMS. In: *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, Bristol, UK, 2012, 833–837. Springer Verlag.
70. R Core Team. *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, 2012. Available at: http://www.R-project.org/. (Accessed May 10, 2015).
71. Friedman J, Hastie T, Tibshirani R. Regularization paths for generalized linear models via coordinate descent. *J Stat Softw* 2010, 33:1–22.
72. De'ath G. mvpart: multivariate partitioning. Available at: http://CRAN.R-project.org/package=mvpart. (Accessed May 10, 2015).
73. Therneau T, Atkinson B, Ripley B. rpart: recursive partitioning and regression trees, R package version 4.1-9. Available at: http://CRAN.R-project.org/package=rpart. (Accessed May 10, 2015).
74. Declarative Languages and Artificial Intelligence Group (Katholieke Universiteit Leuven) and Department of Knowledge Technologies (Jožef Stefan Institute). CLUS system. Available at: https://dtai.cs.kuleuven.be/clus/. (Accessed May 10, 2015).

APPENDIX A

PROOF OF LEMMA 1

We present here the proof of Lemma 1, when OLS estimations of the coefficients are used. The case of ridge regression is similar.

Let $\mathbf{X}$ be the $N \times m$ matrix of input observations and $\mathbf{Y}$ the $N \times d$ matrix of output observations. Let us assume that $\mathbf{X}^t\mathbf{X}$ is invertible, otherwise, OLS estimation cannot be applied. Let also consider that the ordering of the chain is exactly as follows: $Y_1, \ldots, Y_d$. Hence, the coefficients of the first target are estimated as the OLS ones:

$$\mathbb{R}^m \ni \beta_1 = \left(\mathbf{X}^t\mathbf{X}\right)^{-1}\mathbf{X}^t\mathbf{y}_1,$$

where $\mathbf{y}_1$ is the first column of $\mathbf{Y}$, corresponding to the observations of Y_1. Next, in the second training step of the chain, the OLS estimation of the coefficients β_2 are computed as the regression of Y_2 over $X_1, \ldots, X_m, Y_1$ as follows:

$$\mathbb{R}^{m+1} \ni \beta_2 = \left(\begin{array}{c|c} \mathbf{X}^t\mathbf{X} & \mathbf{X}^t\mathbf{y}_1 \\ \hline \mathbf{y}_1^t\mathbf{X} & \mathbf{y}_1^t\mathbf{y}_1 \end{array}\right)^{-1} \left(\begin{array}{c} \mathbf{X}^t \\ \mathbf{y}_1^t \end{array}\right) \mathbf{y}_2.$$

Using the formula for computing the inverse of a block-defined matrix we obtain:

$$\left(\begin{array}{c|c} \mathbf{X}^t\mathbf{X} & \mathbf{X}^t\mathbf{y}_1 \\ \hline \mathbf{y}_1^t\mathbf{X} & \mathbf{y}_1^t\mathbf{y}_1 \end{array}\right)^{-1} = \left(\begin{array}{c|c} \left(\mathbf{X}^t\mathbf{X}\right)^{-1} + \beta_1\mathbf{C}\mathbf{D} & -\beta_1\mathbf{C} \\ \hline -\mathbf{C}\mathbf{D} & \mathbf{C} \end{array}\right),$$

where

$$\beta_1 = \left(\mathbf{X}^t\mathbf{X}\right)^{-1}\mathbf{X}^t\mathbf{y}_1 \in \mathbb{R}^{m\times 1},$$

$$\mathbf{C} = \left(\mathbf{y}_1^t\mathbf{Y}_1 - \mathbf{y}_1^t\mathbf{X}\left(\mathbf{X}^t\mathbf{X}\right)^{-1}\mathbf{X}^t\mathbf{y}_1\right)^{-1} \in \mathbb{R}^{1\times 1},$$

$$\mathbf{D} = \beta_1^t = \mathbf{y}_1^t\mathbf{X}\left(\mathbf{X}^t\mathbf{X}\right)^{-1} \in \mathbb{R}^{1\times m}.$$

Assuming that $\mathbf{y}_1^t\mathbf{y}_1 - \mathbf{y}_1^t\mathbf{X}\left(\mathbf{X}^t\mathbf{X}\right)^{-1}\mathbf{X}^t\mathbf{y}_1$ is invertible, i.e., it is different from 0, we have

$$\beta_2 = \left(\begin{array}{c} \overline{\beta}_2 \\ \beta_{2,1} \end{array}\right) = \left(\begin{array}{c} \left(\mathbf{X}^t\mathbf{X}\right)^{-1}\mathbf{X}^t\mathbf{y}_2 + \beta_1\mathbf{C}\mathbf{D}\mathbf{X}^t\mathbf{y}_2 \\ -\beta_1\mathbf{C}\mathbf{Y}_1^t\mathbf{y}_2 - \mathbf{C}\mathbf{D}\mathbf{X}^t\mathbf{y}_2 + \mathbf{C}\mathbf{y}_1^t\mathbf{y}_2 \end{array}\right),$$

and the model of the first two steps of the chain can be expressed as:

$$\hat{y}_1 = \beta_1^t \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} \quad \text{and} \quad \hat{y}_2 = \beta_2^t \begin{pmatrix} x_1 \\ \vdots \\ x_m \\ \hat{y}_1 \end{pmatrix}.$$

Finally, substituting $\hat{y}_1$ into the equation with $\hat{y}_2$, we obtain:

$$\hat{y}_2 = \overline{\beta}_2^t \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} + \beta_{2,1}\beta_1^t \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} = \left(\overline{\beta}_2^t + \beta_{2,1}\beta_1^t\right) \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix}.$$

Therefore, it is easy to see now that:

$$\overline{\beta}_2^t + \beta_{2,1}\beta_1^t = \left(\mathbf{X}^t\mathbf{X}\right)^{-1}\mathbf{X}^t\mathbf{y}_2. \tag{A1}$$

The right-hand side of Eq. (A1) are the OLS estimations of the regression coefficients of Y_2 over $X_1, \ldots, X_m$. Hence, the second step of the chain is equivalent to the OLS estimation of a ST model. Iterating the argument we obtain that every step of the chain is equivalent to the ST model.

APPENDIX B

R CODE FOR A PARALLEL IMPLEMENTATION OF ST

We developed the following R source code as an example for a parallel implementation of the ST method. It consists of (1) a function for learning ST models with a user-defined base model (referred to as the input parameter **base**) and a given training data set (**data**), (2) a function for testing the learned ST models and predicting the output target values given new unseen instances (i.e., **newdata**), and (3) an example of use of both functions: we consider here learning and testing a ST method using a support vector machine as a base model.

Note that, any available R base model implementation could be also used. For instance, we have tested these R code fragments with linear models (e.g., **lm** and **glm**), local polynomial regression (**loess**), robust linear regression (**rlm** from the **MASS** package), ridge linear regression (**linearRidge** from the **ridge** package), SVR (**svm** from the **e1071** package), and nonparametric kernel regression (npreg from the **np** package).

Moreover, it is possible to use parallel computations using the parameter **mc.cores**. In this example, we make use of the **mclapply** function, that relies on forking and thus works in parallel if **mc.cores>1**, only when performed on UNIX-like systems (see **parallel** documentation for its use on Windows systems).


```
require("parallel")   #parallel package

#1)Definition of the learning function for ST models

single_target_mvr<-function(outputs, inputs, base = lm, data,
                            mc.cores = 1, ...){
model_list <- mclapply(mc.cores = mc.cores,
                       X = outputs, FUN = function(out){
form <- as.formula(paste(out, "~", paste(inputs,
                            collapse = "+"),sep = "␣"))
return(base(form, data = data[, c(out, inputs)],...))})
class(model_list) <- "single_target"
names(model_list) <- outputs
return(model_list)
}

#2)Definition of the prediction function

predict.single_target <-  function(object,newdata = NULL,
                                   mc.cores=1,...){
prediction<-mclapply(mc.cores = mc.cores, X = object,
                     FUN = predict , newdata = newdata)
return(as.data.frame(prediction))
}

#3)Example of use of both functions for learning and testing ST
# using support vector machine as a base model

require("MASS")   #MASS package

#Generate a random covariance matrix with ones in the diagonal

A <- matrix(nrow = 8, runif(64, min = -1, max=1))
A <- apply(A, MARGIN = 1,
           FUN = function(x){return(x/sqrt( sum(x^2)))})
Sigma <- t(A) %*% A

#Generate training and test set from a multivariate
#Gaussian distribution with Sigma as covariance matrix

Dtrain <- data.frame(mvrnorm(n=1000,mu=rep(0,8),Sigma=Sigma))
Dtest <- data.frame(mvrnorm(n=1000, mu=rep(0,8), Sigma=Sigma))

require("e1071")   #e1071 package for svm
mc.cores <- min( 4, (detectCores() - 1) ) #number of cores

#Learn the ST model

STsvm <- single_target_mvr(outputs=names(Dtrain)[5:8],
                           inputs=names(Dtrain)[1:4], base=svm,
                           data=Dtrain, mc.cores= mc.cores)

#Predict target values and compute the mean squared error

prediction <- predict(STsvm , newdata = Dtest)
mse <- mean((prediction - Dtest[,5:8])^2)
```