Multi-Target Support Vector Regression Via Correlation Regressor Chains

Gabriella Melki[a], Alberto Cano[a], Vojislav Kecman[a], Sebastián Ventura[b,c]

[a]*Department of Computer Science, Virginia Commonwealth University, USA*
[b]*Department of Computer Science and Numerical Analysis, University of Cordoba, Spain*
[c]*Department of Computer Science, King Abdulaziz University, Saudi Arabia Kingdom*

Abstract

Multi-target regression is a challenging task that consists of creating predictive models for problems with multiple continuous target outputs. Despite the increasing attention on multi-label classification, there are fewer studies concerning multi-target (MT) regression. The current leading MT models are based on ensembles of regressor chains, where random, differently ordered chains of the target variables are created and used to build separate regression models, using the previous target predictions in the chain. The challenges of building MT models stem from trying to capture and exploit possible correlations among the target variables during training, at the expense of increasing the computational complexity of model training. This paper presents three multi-target support vector regression models. The first involves building independent, single-target Support Vector Regression (SVR) models for each output variable. The second builds an ensemble of random chains using the first method as a base model. The third calculates the targets' correlations and forms a maximum correlation chain, which is used to build a single chained support vector regression model. The experimental study evaluates and compares the performance of the three approaches with six other state-of-the-art multi-target regressors. The experimental results are then analyzed using non-parametric statistical tests. The results show that the maximum correlation SVR approach improves the performance of using ensembles of random chains.

Keywords: Multi-target regression, multi-output regression, regressor chains, support vector regressor.

1. Introduction

In supervised learning, *single-target* (ST) models are trained to predict the value of a single, categorical or numeric, target attribute of a given example. In some cases, more than one target, or output, can be associated with a single sample input. These situations are handled by a generalization of ST learning, which involves predicting these multiple outputs concurrently, and is known as multi-target (MT) learning [1, 5]. Specifically, MT learning includes *multi-target regression* (MTR), which addresses the prediction of continuous targets, *multi-label classification* [45] which focuses on binary targets, and *multi-dimensional classification* which describes the prediction of discrete targets [5, 35].

Multi-target prediction has the capacity to generate models representing a wide variety of real-world applications, ranging from natural language processing [23] to bioinformatics [32]. Other application areas include ecology [1], gene function prediction [25], predicting the quality of vegetation [20, 26], stock price index forecasting [43], and operations research [5, 21]. Constructing models for these types of real-world problems presents many challenges, such as missing data, (due to targets not being observed or recorded), and noisy data (due to instrument, experimental or human error). However, a characteristic of the MT datasets used in these applications and elsewhere, is that they are generated by a single system, indicating that the nature of the outputs captured has some structure [21]. Even though modeling the multi-variate nature and possible complex relationships between the target variables is challenging [5], they are more accurately represented by an MT model.

Several methods have been proposed for solving such multi-target tasks, and can be categorized into two groups. The first being *problem transformation* methods, or *local* methods, in which the multi-target problem is transformed into multiple single-target problems, each solved separately using base, or standard, classification and regression algorithms. The second being *algorithm adaptation* methods, or *global*, or *big-bang* methods, that adapt existing single-target methods to predict all the target variables simultaneously [5, 25]. Using *problem transformation* algorithms for a domain of t target variables, t predictive models must be constructed, each predicting a single-target variable [25]. Prediction for an unseen sample would be obtained by running each of the t single-target models and concatenating their results. Conversely, when using *algorithm adaptation* algorithms for the same domain of t target variables, only one model would need to be constructed which would output all t predictions.

It has previously been shown that *algorithm adaptation* methods perform bet-

ter than *problem transformation* methods [25, 36]. The most valuable advantage of using multi-target techniques is that, not only are the relationships between the sample variables and the targets exploited, but the relationships between the targets amongst themselves are as well [3, 8]. Single-target techniques, on the other hand, eliminate any possibility of learning from the possible relationships between the target variables because a single, independent model is trained for each target separately [4]. Another advantage of MT techniques is model interpretability [1, 43]. A single multi-target model is highly more interpretable than a series of single-target models. Not only is a single MT model more interpretable, but it could also be considerably more computationally efficient to train, rather than training multiple single-target models individually [2].

This paper presents three novel approaches to solving multi-target regression problems. The objective of this research topic is to investigate whether building a regression model using chaining is better than building independent single-target models for each target variable, and whether the maximization of the correlation in a chain performs better than an ensemble of random chains, as conducted by current state-of-the-art algorithms. Explain why we used Support Vector Regression

The first approach consists of a *problem transformation* method, in which a single-target Support Vector Regression (SVR) model is built for each target variable. This would be the base case for our model comparisons, where each of the models would be independently predicting a single-target variable. The second approach is an *algorithm adaptation* method called Support Vector Regression with Random Chains (SVRRC), inspired by the classification MT method called Ensemble of Random Chains Corrected (ERCC) [43], where random chaining is used to build an ensemble of support vector regressors. In this case, a number of random chains is generated to represent how each of the targets will be chained to train the model. This exploits possible dependencies between the target variables and ensures that they get used during model training. The third *algorithm adaptation* approach eliminates the need to generate a number of random chains, and is called Support Vector Regression with Correlation Chaining (SVRCC). It involves a single maximum correlation chain, in which the targets are arranged in an order that represents their correlation to one another. This ensures that the targets that will be used as input are correlated with the following target prediction. Rather than randomly creating chains and having the final model be based on a combination of the generated models, the third approach builds a single model based on the target's maximum correlation.

The experimental study evaluates and compares the performance of the three approaches, together with six other state-of-the-art multi-target regressors, on a

set of 19 datasets with varied input size and output targets. The results of the experiments are analyzed using non-parametric statistical analysis, namely the Bonferroni-Dunn and Wilcoxon tests [17]. These post-hoc tests involve multiple comparisons among the algorithms, where they try and show significant differences in their performances across all datasets. The statistical analysis of the experiments presented in this paper shows the increase in performance of the support vector regressors, specifically, the maximum correlation chain, SVRCC.

The paper is structured as follows. Section 2 reviews related works on multi-target regression. Section 3 presents the three multi-target support vector regression approaches. Section 4 presents the experimental study. Section 5 discusses the results and the statistical analysis. Finally, Section 6 shows the main conclusions of this work.

2. Background

This section defines first the notation that will be used in the paper, then it formally describes the multi-target regression problem along with the relevant state-of-the-art algorithms.

Table 1: Notation

Definition	Notation
Number of Samples	$\mathcal{N}$
Number of Input Attributes	d
Input Space	$\boldsymbol{X} = \{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_i, \ldots, \boldsymbol{X}_d\} \in \mathbb{R}^d$, $\lvert\boldsymbol{X}_i\rvert = \mathcal{N}$, $1 \leq i \leq d$
Input Instance	$\boldsymbol{x}^{(l)} = (x_1^{(l)}, \ldots, x_d^{(l)}) \in \boldsymbol{X}$, $1 \leq l \leq \mathcal{N}$
Number of Dataset Targets/Outputs	m
Target Space	$\boldsymbol{Y} = \{\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_j, \ldots, \boldsymbol{Y}_m\} \in \mathbb{R}^m$, $\lvert\boldsymbol{Y}_j\rvert = \mathcal{N}, 1 \leq j \leq m$
Target Instance	$\boldsymbol{y}^{(l)} = (y_1^{(l)}, \ldots, y_m^{(l)}) \in \boldsymbol{Y}$, $1 \leq l \leq \mathcal{N}$
Full Multi-Target (MT) Training Dataset	$\mathcal{D} = \{(x_1^{(1)}, y_1^{(1)}), \ldots, (x_d^{(\mathcal{N})}, y_m^{(\mathcal{N})})\}$
Single-Target (ST) Dataset with j^{th} Target	$\mathcal{D}_j = \{(x_1^{(1)}, y_j^{(1)}), \ldots, (x_d^{(\mathcal{N})}, y_j^{(\mathcal{N})})\} \in \mathcal{D}$, $1 \leq j \leq m$
Number of Cross-Validation (CV) Sets	k
ST Test Dataset with j^{th} Target, i^{th} CV Fold	$\mathcal{D}_j^{(i)} = \{(x_1^{(i)}, y_j^{(i)}), \ldots, (x_d^{(i)}, y_j^{(i)}) \in \mathcal{D}_j$, $i \in \{1, \ldots, \mathcal{N}\}$
ST Training Dataset with j^{th} Target, Excluding the i^{th} CV Fold	$\mathcal{D}_j^{(k-i)} = \mathcal{D}_j \setminus \mathcal{D}_j^{(i)}$
MT Regression Model	$h : \boldsymbol{X} \times \boldsymbol{Y}$
ST Regression Model	$h_j : \boldsymbol{X} \times \boldsymbol{Y}_j$, $1 \leq j \leq m$
Unknown Sample	$\boldsymbol{x}^{(\mathcal{N}')} = \{\boldsymbol{x}^{(\mathcal{N}+1)}, \ldots, \boldsymbol{x}^{(N')}\}$
Predicted Values for Unknown Sample	$\boldsymbol{y}^{(\mathcal{N}')} = \{\boldsymbol{y}^{(\mathcal{N}+1)}, \ldots, \boldsymbol{y}^{(\mathcal{N}')}\}$

Table 2: Transformation to m Single-Target Datasets

Dataset	Values	Output
$\mathcal{D}_1$	$\{(x_1^{(1)}, y_1^{(1)}), \ldots, (x_d^{(\mathcal{N})}, y_1^{(\mathcal{N})})\}$	$h_1 : \mathcal{D}_1 \rightarrow \mathbb{R}$
$\mathcal{D}_2$	$\{(x_1^{(1)}, y_2^{(1)}), \ldots, (x_d^{(\mathcal{N})}, y_2^{(\mathcal{N})})\}$	$h_2 : \mathcal{D}_2 \rightarrow \mathbb{R}$
$\vdots$	$\vdots$	$\vdots$
$\mathcal{D}_j$	$\{(x_1^{(1)}, y_j^{(1)}), \ldots, (x_d^{(\mathcal{N})}, y_j^{(\mathcal{N})})\}$	$h_j : \mathcal{D}_j \rightarrow \mathbb{R}$
$\vdots$	$\vdots$	$\vdots$
$\mathcal{D}_m$	$\{(x_1^{(1)}, y_m^{(1)}), \ldots, (x_d^{(\mathcal{N})}, y_m^{(\mathcal{N})})\}$	$h_m : \mathcal{D}_m \rightarrow \mathbb{R}$

2.1. Notation

Let $\mathcal{D}$ be a training dataset of $\mathcal{N}$ instances. Let $\boldsymbol{X} \in \mathcal{D}$ be a matrix consisting of d input variables and $\mathcal{N}$ samples, sometimes called input space, having a domain of $\boldsymbol{X} \in \mathbb{R}^d$. Let $\boldsymbol{Y} \in \mathcal{D}$ be a matrix consisting of m target variables and $\mathcal{N}$ samples, sometimes called output space, having a domain of $\boldsymbol{Y} \in \mathbb{R}^m$. For each sample $(\boldsymbol{x}^{(l)}, \boldsymbol{y}^{(l)}) \in \mathcal{D}$, $\boldsymbol{x}^{(l)} = (x_1^{(l)}, \ldots, x_d^{(l)})$ and $\boldsymbol{y}^{(l)} = (y_1^{(l)}, \ldots, y_m^{(l)})$ are the input and output vectors respectively, where $l \in \{1, \ldots, \mathcal{N}\}$. Using the training dataset $\mathcal{D} = \{(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), \ldots, (\boldsymbol{x}^{(\mathcal{N})}, \boldsymbol{y}^{(\mathcal{N})})\}$, the goal is to learn a multi-target regression model $h : \boldsymbol{X} \times \boldsymbol{Y}$, that assigns a vector $\boldsymbol{y}$ with m target values, for each input instance $\boldsymbol{x}$. The model will then be used to predict the values of $\{\boldsymbol{y}^{(\mathcal{N}+1)}, \ldots, \boldsymbol{y}^{(\mathcal{N}')}\}$ for new unlabeled input vectors $\{\boldsymbol{x}^{(\mathcal{N}+1)}, \ldots, \boldsymbol{x}^{(\mathcal{N}')}\}$. Table 1 summarizes the notation used in this paper.

2.2. Multi-Target Regression Methods

In the context of *problem transformation* for multi-target models, m single-target models will be trained on the dataset $\mathcal{D}_j = \{(x_1^{(1)}, y_j^{(1)}), \ldots, (x_d^{(N)}, y_j^{(\mathcal{N})})\}$, where $j \in \{1, \ldots, m\}$. This way there are m independent, single-target models, one model for each target variable. This is described as the baseline Single-Target (ST) model in [36]. A generalized visualization of the *problem transformation* method is shown in Table 2. Many *problem transformation* methods have been proposed to solve multi-target problems, which are detailed next.

Multiple authors have proposed Multiple-Target Support Vector Regression methods [5, 43, 44]. Specifically, *Xiong et. al.* presented a support vector regression method with a firefly heuristic in [43], in the context of interval forecasting

of a stock price index. Originally proposed in [33], the algorithm was modified in [43] with a firefly heuristic, named FA-MSVR, in order to intelligently identify the appropriate SVR hyper-parameters. Their method for optimizing the FA-MSVR was using an iterative reweighted least squares (IRWLS) approach based on a quasi-Newton strategy. To produce attractive results using support vector machines, appropriate hyper-parameters must be selected. This is a crucial and computationally expensive step to ensure the SVR model is performing to the best of its capabilities [46], which is why a firefly heuristic was used in [43]. The results obtained by *Xiong et. al.* indicate that FA-MSVR proved to be a promising alternate method for time series forecasting, thus highlighting the importance of setting the SVR hyper-parameters.

Moreover, other approaches based on Linear Target Combinations for MT Regression [39], and Multi-Objective Decision Trees (MORF) [27] have been proposed. Most commonly investigated issues for MT learning problems include dimensionality reduction for high-dimensional multi-labeled data. The curse of dimensionality is problematic due to the possible correlations between the data and the targets [10, 13, 22]. This multi-collinearity may cause the learning task to be more difficult and complex [34]. To reduce the dimensionality of the data, without losing the possible relationships to the targets, careful feature selection could be performed, which is a difficult task as well [29, 30]. Another issue would be processing large datasets quickly and feasibly (because of memory constraints), while providing insightful information [5, 7].

The MTS, MTSC, ERC, and ERCC methods are introduced by *Spyromitros et. al.* in [36]. The idea behind these algorithms was to investigate whether advances in multi-label learning can be successfully used and implemented in a multi-target regression setting, as well as shedding light on modeling target dependencies. They first describe the MTS and ERC models, which are both inspired by multi-label classification algorithms, and then introduce their corrected versions. These two methods involve two stages of learning, the first being building ST models and the second uses the knowledge gained by the first step to predict the target variables while using possible relationships the target variables might have with one another.

The two stages of training in MTS involve firstly, training m independent single-target models, like in ST. In the second step, a second set of m meta models are learned for each target variable, $\boldsymbol{Y}_j$, $1 \leq j \leq m$. These meta models are learned on a transformed dataset, where the input attributes space is expanded by adding the approximated target variables obtained in the first stage, excluding the j^{th} target being predicted. Each m meta model, $h_j^* : \boldsymbol{X} \times \mathbb{R}^{m-1} \rightarrow \mathbb{R}$, is

learned by the modified dataset $\mathcal{D}_j^* = \{(x_1^{*(1)}, y_j^{*(1)}), \ldots, (x_d^{*(\mathcal{N})}, y_j^{*(\mathcal{N})})\}$, where $\boldsymbol{x}_j^{*(l)} = \{x_1^{(l)}, \ldots, x_d^{(l)}, \hat{y}_1^{(l)}, \ldots, \hat{y}_{j-1}^{(l)}, \hat{y}_{j+1}^{(l)}, \ldots, \hat{y}_m^{(l)}\}$, $l \in \{1, \ldots, \mathcal{N}\}$ are the input vectors along with the $m-1$ predicted target variables, represented by $\hat{\boldsymbol{y}}$, obtained by the first step. To predict the output for a new input vector $\boldsymbol{x}^{(q)}$, the models trained in the first stage are applied and an output vector, $\hat{\boldsymbol{y}}^{(q)} = \{\hat{\boldsymbol{y}}_1^{(q)}, \ldots, \hat{\boldsymbol{y}}_m^{(q)}\} = \{h_1(\boldsymbol{x}^{(q)}), \ldots, h_m(\boldsymbol{x}^{(q)})\}$, is obtained. The second stage models are then applied on the transformed input vector $\boldsymbol{x}_j^{*(q)}$, as shown above, to produce a final output vector.

The ERC method is somewhat similar to the MTS method. In the training of a Regression Chain (RC) model, a random chain, or sequence, of the set of target variables is selected and for each target in the chain, models are built sequentially by using the output of the previous model as input for the next [37]. If the default, ordered chain is $C = \{\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_m\}$, the first model $h_1 : \boldsymbol{X} \rightarrow \mathbb{R}$ is trained for $\boldsymbol{Y}_1$, as in ST. For the subsequent models $h_{j,j>1}$, the dataset is transformed into $\mathcal{D}_j^* = \{(\boldsymbol{x}_j^{*(1)}, y_j^{(1)}), \ldots, (\boldsymbol{x}_j^{*(\mathcal{N})}, y_j^{(\mathcal{N})})\}$, where $\boldsymbol{x}_j^{*(l)} = \{x_1^{(l)}, \ldots, x_d^{(l)}, y_1^{(l)}, \ldots, y_{j-1}^{(l)}\}$ are the input vectors transformed by sequentially appending the true values of each of the previous targets in the chain. For a new input vector $\boldsymbol{x}^{(q)}$, the target values are unknown. So once the models are trained, the unseen input vector $\boldsymbol{x}^{(q)}$ will be appended with the approximated target values, making the models dependent on the approximated values obtained in each step. One of the issues associated with this method is that, if a single random chain is used, the possible relationships between the targets at the head of the chain and the end of the chain are not exploited due to the algorithm's sequential nature. Also, prediction error in the earlier stages of the models will be propagated as the rest of the models are trained, which is why the Ensemble of Regressor Chains was proposed in [36]. Instead of a single chain, k chains are created at random, and the final prediction values are obtained by taking the mean values of the k predicted values for each target.

In the methods described above, the estimated target variables (meta-variables) are used as input in the second stage of training. In both methods, the models are trained using these meta-variables that become noisy at prediction time, and thus the relationship between the meta-variables and target variable is muddied. Dividing the training set into sets, one for each stage, would not help this situation because both methods would be trained on training sets of decreasing size. Due to these issues, *Spyromitros et. al.* proposed modifications, in [36], to both methods that resembles k-fold cross-validation (CV) to be able to obtain unbiased estimates of the meta-variables. These methods are called Regression Chains Cor-

rected (RCC) and Multi-Target Stacking Corrected (MTSC).

The ERCC and MTSC procedures involve repeating the RCC and MTS procedures k times, respectively, with k randomly ordered chains for ERCC, and k different modified training sets for MTSC. The algorithms were tested and compared using Bagging of 100 regression trees as their base regression algorithm with ERC and ERCC ensemble size of 10, and 10-fold cross-validation. The corrected methods exhibited better performance than their original variants, as well as ST models. The ERCC algorithm had the best overall performance, as well as being statistically significantly more accurate of all the methods tested. These methods can be found and used through the open-source Java library, Mulan [38]; to replicate the results found in [36].

The following section presents our approaches to solving the multi-target regression problem inspired by the techniques presented in [36]. It will show that exploiting and making use of the possible correlations in the target variables produces better results than not.

3. Multi-target SVR proposal

Three novel models have been implemented for the purposes of multi-target regression. The base model is the SVR model, where m single-target soft-margin non-linear support vector regressors (NL-SVR) are built for each target variable $\boldsymbol{Y}_j$. For NL-SVR, the regularized soft-margin loss function given in equation (1) is minimized [24, 14],

$$\underset{\boldsymbol{w},\xi,\xi^*}{\text{minimize}} \quad \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{l=1}^{\mathcal{N}}\left(\xi^{(l)} + \xi^{*(l)}\right) \tag{1}$$

$$\text{subject to} \begin{cases} y_l - \langle \boldsymbol{w}, \phi\left(\boldsymbol{x}^{(l)}\right)\rangle \leq \epsilon + \xi^{(l)} & \text{(1a)} \\ \langle \boldsymbol{w}, \phi\left(\boldsymbol{x}^{(l)}\right)\rangle - y^{(l)} \leq \epsilon + \xi^{*(l)} & \text{(1b)} \\ \xi^{(l)}, \xi^{*(l)} \geq 0 & \text{(1c)} \end{cases}$$

where $\boldsymbol{w}$ represents the SVR weight vector, ϵ represents how precise the approximations are, $C > 0$ determines how to penalize deviations from ϵ, $\phi(\cdot)$ represents a feature mapping function, $\xi^{(l)}$ and $\xi^{*(l)}$ are slack variables, and $y^{(l)}$ is the label corresponding to the input vector $\boldsymbol{x}^{(l)}$. For simplicity, the bias SVR term has been excluded. In our algorithm implementation, the dual of this formulation [11, 14] given by (2) is solved,

$$\underset{\boldsymbol{\alpha},\boldsymbol{\alpha}^*}{\text{maximize}} \quad -\frac{1}{2}\sum_{l,k=1}^{\mathcal{N}}\left(\boldsymbol{\alpha}^{(l)}-\boldsymbol{\alpha}^{*(l)}\right)\left(\boldsymbol{\alpha}^{(k)}-\boldsymbol{\alpha}^{*(k)}\right)\mathcal{K}\left(\boldsymbol{x}^{(l)},\boldsymbol{x}^{(k)}\right) - \epsilon\sum_{l=1}^{\mathcal{N}}\left(\boldsymbol{\alpha}^{(l)}+\boldsymbol{\alpha}^{*(l)}\right)+\sum_{l=1}^{\mathcal{N}} y^{(l)}\left(\boldsymbol{\alpha}^{(l)}-\boldsymbol{\alpha}^{*(l)}\right) \tag{2}$$

$$\text{subject to} \quad \textstyle\sum_{l=1}^{\mathcal{N}}\left(\boldsymbol{\alpha}^{(l)}-\boldsymbol{\alpha}^{*(l)}\right)=0, \quad \boldsymbol{\alpha}^{(l)},\boldsymbol{\alpha}^{*(l)}\in[0,C] \tag{2a}$$

where the $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}^*$ vectors correspond to the SVR dual variables and $\mathcal{K}\left(\boldsymbol{x}^{(l)},\boldsymbol{x}^{(k)}\right)=\phi(\boldsymbol{x}^{(l)})'*\phi(\boldsymbol{x}^{(k)})$ is an $(\mathcal{N}\times\mathcal{N})$ Gaussian kernel matrix which is dependent on the γ parameter for its broadness. Using the SVR optimization problem described, the multi-target problem is solved by transforming it into m single-target problems, as shown in Algorithm 1. This algorithm will output m single-target models, h_j, $\forall j=1,\ldots,m$, for a given dataset $\mathcal{D}$. It first splits the dataset into m separate ones, each with a single-target variable $\boldsymbol{Y}_j$, and then builds a distinct SVR model for each of the datasets.

Building m ST models was a good base-line model, but as mentioned previously, it does not use any of the possible correlations between the target attributes during training. If these correlations are not exploited, it could retract from the model's potential performance. Therefore, we also proposed to construct a series of random chains and create an ensemble model, as done in [36], but using our base-line SVR method, named SVR Random Chains (SVRRC). For SVRRC, ensembles of at most 10 random chains are built, with length m, of different and distinct permutations of the target variable indices. For each chain, we train m chained models using the targets true values. Due to the computational complexity of building $m!$ distinct chains and training $(m!)\times m$ models, the number of ensembles and chains are limited to a maximum of 10, as proposed by *Spyromitros et. al.* in [36]. However, if the number of target variables is less than 3, i.e. $m!\leq 10$, we construct all $m!$ random chains.

For each of the random chains, the model is trained by predicting the first target variable in the chain. Next, the first target's true value, $\boldsymbol{Y}_j$, is appended to the end of the training set as such, $\mathcal{D}^*_{j+1}=\{\boldsymbol{X}^*_{j+1},\boldsymbol{Y}_j\}$, where $\boldsymbol{X}^*_{j+1}$ is the appended training set, $\boldsymbol{X}^*_{j+1}=\{x_1^{(l)},\ldots,x_d^{(l)},y_1^{(l)},\ldots,y_j^{(l)}\}, l=\{1,\ldots,\mathcal{N}\}$.

The chaining process is repeated for all the target indices in the chains. When chaining target values, there are two main options: using the predicted value as

Maybe we can give a clearer view on how the models work in the algorithms section

Why is this method good and why is it bad? It seems that the link between the two was lost on the reviewers, so we need to make it more clear

Clarify this, maybe with a diagram?

Algorithm 1 MT Support Vector Regression (SVR)

Input: Training dataset $\mathcal{D}$, number of cross-validation folds k
Output: ST models $h_j, j = 1, \ldots, m$

Build m ST SVR Models
for $j = 1$ to m **do**
  $\mathcal{D}_j = \{(x_1^{(1)}, y_j^{(1)}), \ldots, (x_d^{(\mathcal{N})}, y_j^{(\mathcal{N})})\}$
  Create the CV training $\mathcal{D}^{(k-i)}$ and test $\mathcal{D}^{(i)}$ sets
  for $i = 1$ to k **do**
    $\mathcal{D}_j^{(k-i)} = \{(x_1^{(k-i)}, y_j^{(k-i)}), \ldots, (x_d^{(k-i)}, y_j^{(k-i)})\}$
    $\mathcal{D}_j^{(i)} = \{(x_1^{(i)}, y_j^{(i)}), \ldots, (x_d^{(i)}, y_j^{(i)})\}$
    Train the j^{th} model on the training set $\mathcal{D}^{(k-i)}$
    $h_j^{(k-i)} : \mathcal{D}_j^{(k-i)} \rightarrow \mathbb{R}$
    Test the j^{th} model on the test set $\mathcal{D}^{(i)}$
    $\hat{\boldsymbol{Y}}_j^{(i)} = h_j^{(k-i)}(\boldsymbol{X}^{(i)})$
  end for
  Calculate and store aRRMSE error for the j^{th} model
end for
return $h_j, j = 1, \ldots, m$

input for the following target, or using the true value of the target variable as input of the subsequent targets. The main problem with the former approach is that errors are propagated, while the latter minimizes error propagation and results in better accuracy results. Our approach and the other methods compared, employ chaining of the true values. Given the ensemble of SVRs, the predicted values for a given instance are calculated by taking the mean of the multiple models generated using different random chains. For predicting unseen inputs that have no target values, the predicted value at each step of the chain $\hat{y_j}^{(l)}$ is appended to the input as shown in Algorithm 3. For SVRRC described in Algorithm 2, at most 10 models will be built (one for each chain RC, $RC_c = \{RC_c^{(1)}, \ldots, RC_c^{(m)}\}$, $c \leq 10$), each iterating and training m chained models. For each of the models in the chain, we use 10-fold cross-validation to ensure the hyper-parameters chosen best describe the data.

First expand on why this method is good, then talk about its weaknesses and how we can make it better

When the number of output variables increases, the number of possible chains increases factorially. Therefore, there is no guarantee that the random chains generated will truly reflect the relationships among the target variables. Moreover, building an ensemble of regressors is computationally expensive. Finding a

Algorithm 2 MT SVR with Random-Chains (SVRRC)

Input: Training dataset $\mathcal{D}$, number of cross-validation folds k
Output: c chained models $h_j, j = \{1, \ldots, m\}, c \leq 10$

Create at most 10 *distinct random chains of size* m
$RC_c^{(m)}$
Create ST transformation of $\mathcal{D}$ *with the* 1^{st} *index in chain* c
$\mathcal{D}_1^* = \{(x_1^{(1)}, y_1^{(1)}), \ldots, (x_d^{(\mathcal{N})}, y_1^{(\mathcal{N})})\}$
Build m *chained models for each chain* $c \in \{1, \ldots, 10\}$
for $j = 1$ to m **do**
  $h_j : \mathcal{D}_j^* \rightarrow \mathbb{R}$
  Initialize appended dataset, $\mathcal{D}_{j+1}^*$
  if $j < m$ **then**
    $\mathcal{D}_{j+1}^* \leftarrow \emptyset$
    Split $\mathcal{D}_j^*$ *into* k *disjoint parts* $\mathcal{D}_j^{*(i)}$, $i = 1, \ldots, k$ *for training and testing*
    for $i = 1$ to k **do**
      $h_j^{(i)} : \mathcal{D}_j^{(k-i)} \rightarrow \mathbb{R}$
      Test the model on the test set $D_j^{*(i)}$
      for $\boldsymbol{x}^{*(i)} \in \mathcal{D}_j^{*(i)}$ **do**
        $\hat{y}_j^{(i)} = h_j^{(i)}(\boldsymbol{x}^{*(i)})$
        Append $\boldsymbol{x}^{*(i)}$ *with the true value* $y_j^{(i)}$*, and add it to set* $\mathcal{D}_{j+1}^*$
        $\mathcal{D}_{j+1}^* = \mathcal{D}_{j+1}^* \cup [\boldsymbol{x}^{*(i)}, y_j^{(i)}]$
      end for
    end for
  end if
end for
return $h_j, j = 1, \ldots, m$

heuristic that allows the identification of a single, most appropriate chain, which fully reflects the output variable interrelations would improve the computational complexity of training the ensemble. Our third proposal builds a single chain based on the maximization of the correlations among the target variables. By calculating the correlation of the target variables and imposing that on the order of the chain, we ensure that each appended target provides some additional knowledge on the training of the next. With SVRRC, there is no proof or reasoning behind the generation of these chains, and since the number of random chains generated

Algorithm 3 SVR Chained Prediction

Input: Unknown sample $\boldsymbol{x}^{(q)}$, chained model $h_j, j = \{1...m\}$
Output: Output vector $\hat{\boldsymbol{y}}^{(q)}$ of size m

Initialize the output vector

$\hat{\boldsymbol{y}}^{(q)} = \mathbf{0}$

Initialize the input space for the first target

$\mathcal{D}_1^* = \boldsymbol{x}^{(q)}$

for $j = 1$ to m **do**

  $\boldsymbol{x}^* \in \mathcal{D}_j^*$

  $\hat{y}_j^{(q)} = h_j(\boldsymbol{x}^*)$

  $\mathcal{D}_{j+1}^* = \mathcal{D}_j^* \cup [\boldsymbol{x}^*, \hat{y}_j^{(q)}]$

end for

Create ST transformation of $\mathcal{D}$ with 1^{st} index in chain c

return $\hat{\boldsymbol{y}}^{(q)}$

is limited to 10, there is no way of ensuring that the 10 chains fully represent the targets' dependencies. Therefore, calculating and using the correlations of the targets would break this uncertainty, as done in Algorithm 4, the SVR Correlation Chain (SVRCC) method. The computational complexity and hardware constraints (memory size) are negligible during the construction of the targets' correlation matrix, since the correlation matrix would be an $(m \times m)$ matrix, and the likelihood that the number of targets is large enough to cause a memory issue is minimal. To calculate the correlation coefficients of the targets, we first construct the targets' co-variance matrix, $\sum_{i,j} = cov(Y_i, Y_j) = \mathbf{E}\left[(Y_i - \mu_i)(Y_j - \mu_j)\right]$, where $\mu_i = \mathbf{E}(Y_i)$, and $\mathbf{E}(Y_i)$ is the expected value of Y_i, $\forall i, j \in \{1, \ldots, m\}$. This matrix will show how the targets change together. We then calculate the correlation coefficients matrix $\rho_Y = corrcoef(Y) = \frac{cov(Y_i, Y_j)}{\sqrt{cov(Y_i, Y_i) cov(Y_j, Y_j)}}$, $\forall i, j \in \{1, \ldots, m\}$. The correlation coefficients matrix will describe the linear relationship among the target variables and we can then sort them in decreasing order, creating our maximum correlation chain. Having a single chain reduces the computational complexity, while providing a competitive model. For an unknown sample, the prediction method is the same as Algorithm 3.

Maybe a the end we can so a summary of the methods and how they are linked with the advantages of each and maybe some expectations on how they might perform?

Algorithm 4 MT SVR with Max-Correlation Chain (SVRCC)

Input: Training dataset $\mathcal{D}$, number of cross-validation folds k
Output: Chained model $h_j, j = \{1, \ldots, m\}$

Find maximum correlation chain for the target variables

$correlation_{(m \times m)} = corrcoef(\boldsymbol{Y})$
$sums_m = sum\left(correlation\left(l, j\right)\right), l = 1, \ldots, m$
Create the maximum correlation chain c *of size* m
$c_m = sort\left(sums_m, \textbf{decreasing}\right)$
$\mathcal{D}^*_{c_1} = \{(x_1^{(1)}, y_{c_1}^{(1)}), \ldots, (x_d^{(\mathcal{N})}, y_{c_1}^{(\mathcal{N})})\}$
Build a chained model, for max-chain c_m
for $j = 1$ to m **do**
  $h_{c_j} : \mathcal{D}^*_{c_j} \rightarrow \mathbb{R}$
  Initialize appended dataset, $\mathcal{D}^*_{c_{(j+1)}}$
  if $j < m$ **then**
    $\mathcal{D}^*_{c_{j+1}} \leftarrow \emptyset$
    Split $\mathcal{D}^*_{c_j}$ *into* k *disjoint parts* $\mathcal{D}^{*(i)}_{c_j}$, $i = 1, \ldots, k$ *for training and testing*
    for $i = 1$ to k **do**
      $h^{(i)}_{c_j} : \mathcal{D}^{*(k-i)}_{c_j} \rightarrow \mathbb{R}$
      Test the model on the test set $\mathcal{D}^{*(i)}_{c_j}$
      for $\boldsymbol{x}^{*(i)} \in \mathcal{D}^{*(i)}_{c_j}$ **do**
        $\hat{y}^i_{c_j} = h^{(i)}_{c_j}(\boldsymbol{x}^{*(i)})$
        Append $\boldsymbol{x}^{*(i)}$ *with the true value* $y^{(i)}_{c_j}$*, and add it to set* $\mathcal{D}^*_{c_{(j+1)}}$
        $\mathcal{D}^*_{c_{(j+1)}} = \mathcal{D}^*_{c_{(j+1)}} \cup [\boldsymbol{x}^{*(i)}, y^{(i)}_{c_j}]$
      end for
    end for
  end if
end for
return $h_j, j = \{1, \ldots, m\}$

4. Experiments

This section presents the experimental comparison of the three models proposed, along with six others from state-of-the-art. It introduces the datasets, algorithms, and performance measures.

4.1. Datasets

This section presents a description of the datasets used in the experiments. Although there are many interesting applications of multi-target regression, there are not many publicly available datasets to use. The datasets used in the experimental study were collected from the Mulan website [38], as well as the UCI Machine Learning Repository [31]. Information on the 19 datasets used is summarized in Table 3, where the number of samples, attributes (dimensionality), and targets are shown. Note that the datasets used in this experiment are the same datasets used in [36] to properly perform our model comparisons, as well as some additional datasets.

- **Electrical Discharge Machining** The EDM dataset's purpose is to optimize the machining time by approximating the behavior of a human operator who controls the values of two variables, gap control and flow control.
- **Solar Flare 1 & Solar Flare 2** The Solar Flare dataset [38] is used for predicting how often 3 types of solar flare are observed in 24 hours. The three types of solar flare - common, moderate, and severe.
- **Water Quality** The Water Quality dataset [38] is used for approximating 14 target attributes that represent chemicals found in Slovenian rivers. 16 input attributes describing the biology of the river, the chemistry and physical water quality, are used to predict the target variables.
- **OES97 & OES10** The Occupational Employment Survey datasets [38] were compiled during the years 1997 (OES97) and 2010 (OES10). There are 16 targets that are randomly selected from a set of employment categories.
- **ATP1d & ATP7d** The Airline Ticket Price datasets [38] are used to predict airline ticket prices. The 411 input variables include details about the different flights and the 6 targets are the minimum prices observed over the next 7 days for 6 airline preferences.
- **SCM1d** The Supply Chain Management dataset [38] is used to predict how likely a product is to succeed. The input variables are the observed prices for a specific day, and 4 projected prices. Each of the 16 target variables correspond to the next day mean price for each of the products.
- **Wisconsin Cancer** Each sample represents follow-up data for one breast cancer patient, who is exhibiting invasive breast cancer, and no evidence

of metastases at the time of diagnosis. This dataset can be used to predict whether the cancer is recurrent, and the time in which it might recur.

- **Stock** This dataset includes the daily stock prices from January 1988 through October 1991 for 10 aerospace companies.

- **California Housing** This dataset includes information on block groups of individuals in California, from the 1990 Census. It contains $20,640$ samples with 7 continuous attributes, which are used to predict the 2 target variables, median house value and the median income.

- **Puma8NH & Puma32H** The Puma8H and Puma32H datasets are synthetically generated from a simulation of the dynamics of a Unimation Puma 560 robot arm [31], used to predict the angular acceleration of one robot arm link. The inputs, 8 for Puma8H and 32 for Puma32H, include angular positions, velocities, and torques of the robot arm.

- **Friedman** This is an artificial dataset where 25 attributes of each of the samples are generated independently from a uniform distribution over $[0, 1]$. To measure the effects of non-related attributes, additional attributes are added to the datasets, which are independent from the output.

- **Polymer** The Polymer Test Plant dataset contains 10 input variables representing measurements of controlled variables, such as temperatures, feed rates, etc, in a polymer processing plant. The 4 target variables are the measurements of the output of that plant.

- **M5SPEC & MP5SPEC & MP6SPEC** These datasets consist of 80 samples of corn measured on 3 different NIR spectrometers, M5, MP5, and MP6.

4.2. Algorithms

This section presents the algorithms that we compare our proposals' performances to; namely ST, MTS, MTSC, ERC, ERCC, and MORF which has been used in the experimental study conducted in [36]. Our contributions are compared to these algorithms because they have shown considerable performance in training multi-target models. The have also made their framework readily available for reproducing their results, allowing proper comparisons to be made with their methods. Moreover, note that all three SVR algorithms are implemented within

Table 3: Multi-Target Regression datasets

Dataset	Samples ($\mathcal{N}$)	Attributes (d)	Targets (m)
EDM	145	16	2
Solar Flare 1	323	10	3
Solar Flare 2	1,066	10	3
Water Quality	1,060	16	14
OES97	323	263	16
OES10	403	298	16
ATP1d	201	411	6
ATP7d	188	411	6
SCM1d	9,125	280	16
Wisconsin Cancer	198	34	2
Stock	950	10	3
California Housing	20,640	7	2
Puma8NH	8,192	8	3
Puma32H	8,192	32	6
Friedman	500	25	6
Polymer	41	10	4
M5SPEC	80	700	3
MP5SPEC	80	700	3
MP6SPEC	80	700	3

the general framework of Mulan's MTRegressor[1] [38], which was built on top of Weka[2] [41], and we also used LIBSVM's Epsilon-SVR [9] implementation of Support Vector Regressors as the base SVR model.

In order to train a model that accurately describes a dataset and is not over-fit, one must experiment with different model hyper-parameters. In the case of SVMs and the SVM regression task, these parameters are the penalty parameter C, the Gaussian kernel parameter γ, and the error or tube parameter ϵ. We experimented

[1]http://mulan.sourceforge.net

[2]http://www.cs.waikato.ac.nz/ml/weka

with a range of parameters given in equations (3a) to (3c), referred to as (3), to ensure the final model is best representative of the dataset the model is being trained on.

$$C \in \{1, 10, 100\} \tag{3a}$$
$$\gamma \in \{1^{-9}, 1^{-7}, 1^{-5}, 1^{-3}, 1^{-1}, 1, 5, 10\} \tag{3b}$$
$$\epsilon \in \{0.01, 0.1, 0.2\} \tag{3c}$$

We have compared the models trained on the different hyper-parameters using the cross-validation procedure which ensures that the models' performances are accurately assessed and the model built is not biased towards the full dataset.

To ensure having a controlled environment when conducting our performance comparisons, the experimental environment for running the competing algorithms was the same as what was done in [36]. This includes the following. The ST baseline model used was Bagging [6] of 100 regression trees [42]. The MTSC and ERCC methods are run using 10-fold cross-validation, and the ensemble size for the ERC and ERCC methods was set to 10. The ensemble size of 100 trees was used for MORF, and the rest of its parameters were set as recommended by [28].

The main preprocessing step used for the experiments was normalization. The input variables were scaled to have a mean value of 0 and standard deviation of 1. To ensure that each sample is definitely used when building the final model, we used sampling without replacement when creating our training and testing sets. The preprocessed and final datasets used in the experiments, along with the code for the algorithms presented in this paper, can be publicly found here:
`http://people.vcu.edu/~acano/MTR-SVRCC`

4.3. Performance Evaluation

The performance metric used to analyze our contributions' performances is shown in Equation 4. For unseen or test datasets of size $\mathcal{N}_{test}$, the performances are evaluated by taking the average, relative root mean square error (aRRMSE) [5, 16] given by,

$$\textit{caRRMSE} = \frac{1}{m}\sum_{j=1}^{m}\sqrt{\frac{\sum_{l=1}^{\mathcal{N}_{test}}(y_j^{(l)} - \hat{y}_j^{(l)})^2}{\sum_{l=1}^{\mathcal{N}_{test}}(y_j^{(l)} - \bar{y}_j)^2}} \tag{4}$$

where $\hat{\mathbf{y}}$ is the predicted output and $\bar{\mathbf{y}}$ is the average of the true output target variable. The test dataset is the hold-out set during cross validation. This ensures our model is evaluated on data that it has not been trained on, and thus unbiased towards the training datasets.

5. Results

This section presents the results from the experimental study. Table 4 shows the aRRMSE results achieved by our algorithm implementations, along with the aRRMSE results obtained by *MORF, ST, MTS, MTSC, ERC,* and *ERCC*. The value with the lowest aRRMSE error is typeset in bold, and the last two rows of the table correspond to the average aRRMSE achieved across all datasets, as well as the average rank of each algorithm calculated over the dataset aRRMSE averages, according to Friedman [17]. The Friedman statistic, with alpha $= 0.05$, (distributed according to chi-square with 8 degrees of freedom) is 31.2560, with a p-value of $1.2600E^{-4}$. The results are sorted in order of each algorithm's decreasing average aRRMSE error.

As Table 4 shows, the SVR method's results are better than the competing algorithms. Even though a single-target model is being built for each of the target variables, it still out-ranked the competing *problem transformation* algorithms, such as the ERCC and MTSC algorithms. Having the single-target method, SVR, perform this well indicates that it would perform even better in an ensemble environment, where the correlations of the target variables are taken into account. This is shown by the increase in average performance and decrease in rank of SVRRC. The indicator that target variable correlation does play a crucial role in multi-target problems is made apparent when SVRRC performed better than building m single-target SVR models. Creating an ensemble model from the random chains of the target variables proved to capture additional invaluable information about the dataset that single-target models could not. For datasets Puma8NH and Mp5spec, the single-target SVR models performed better than SVRRC, which indicates that the targets possibly have minimal correlation with one another. Table 4 also shows that building an ensemble with random chains using the base SVR model performed better than the ERCC model, which uses regression trees as it's base model.

expand on this some more, maybe use some examples from the datasets where the targets could be correlated. Also note the correlation may be non-existent so our methods might not provide additional improvement. For the ones it failed on, check correlation coefficient?

The overall improvement in performance of SVRRC over SVR shows that chaining the target variables to the input positively contributes to the training of the model. This implies that capturing the correlation between the target variables is valuable during the model's training process. Therefore, having a single chain representing the direction of maximum correlation between the targets should also boost the algorithm's performance. This is shown by the results in Table 4, where the SVRCC algorithm outperforms the ensemble SVRRC, the single-target SVR, as well as all the competing methods.

Some of the datasets used in our experiments are considered to be sparse,

Table 4: aRRMSE Results & Algorithm Rank

Dataset	MORF	ST	MTS	MTSC	ERC	ERCC	SVR	SVRRC	SVRCC
EDM	0.7338	0.7421	0.7430	0.7396	0.7435	0.7407	0.7058	0.7069	**0.6978**
Solar-flare1	1.2825	1.1354	1.1270	1.0680	1.0501	1.0887	0.9916	0.9455	**0.9319**
Solar-flare2	1.4248	1.1494	**0.9448**	1.0553	1.0532	1.0879	1.0385	1.0353	1.0297
Water-quality	**0.8994**	0.9083	0.9110	0.9095	0.9097	0.9059	0.9542	0.9557	0.9451
Oes97	0.5490	0.5248	0.5259	0.5243	0.5254	0.5239	0.4641	0.4725	**0.4634**
Oes10	0.4518	0.4200	0.4201	0.4205	0.4202	0.4199	0.3570	0.3551	**0.3538**
Atp1d	0.4222	0.3735	0.3716	0.3717	**0.3710**	0.3724	0.3772	0.3739	0.3783
Atp7d	0.5508	0.5248	0.5143	**0.5074**	0.5343	0.5124	0.5455	0.5427	0.5341
Scm1d	0.5663	0.4775	0.4741	0.4701	0.4709	0.4663	0.4674	0.4726	**0.4620**
Wisconsin Cancer	0.9413	0.9313	**0.9287**	0.9333	0.9309	0.9321	0.9555	0.9483	0.9427
Stock	0.1652	0.1839	0.1771	0.1784	0.1778	0.1743	0.1464	0.1436	**0.1421**
California Housing	0.6611	0.6441	0.6939	0.6592	0.6682	0.6133	0.6130	0.5945	**0.5851**
Puma8NH	0.7863	0.8139	0.8111	0.8325	0.8203	0.8209	**0.7655**	0.7744	0.7675
Puma32H	0.9405	**0.8715**	0.8718	0.8775	0.8729	0.8738	0.9364	0.9366	0.9355
Friedman	0.9394	0.9214	0.9249	0.9202	0.9205	0.9199	0.9218	0.9208	**0.9195**
Polymer	0.6159	0.6185	**0.5701**	0.6604	0.6568	0.6347	0.6072	0.6052	0.6011
M5spec	0.5910	0.5503	0.5930	0.5638	0.5521	0.5526	0.3225	**0.2935**	0.3025
Mp5spec	0.5521	0.5112	0.5645	0.5165	0.5145	0.5156	**0.2510**	0.2622	0.2558
Mp6spec	0.5553	0.5166	0.5686	0.5128	0.5203	0.5100	0.2850	**0.2670**	0.2778
Average	0.7173	0.6747	0.6703	0.6695	0.6691	0.6666	0.6161	0.6109	**0.6066**
Rank	7.1579	5.4737	5.6316	5.5263	5.5263	4.5789	4.2632	4.0526	**2.7895**

such as the M5SPEC, MP5SPEC, MP6SPEC datasets, which only have 80 samples with 700 attributes, or the OES datasets where the number of samples and attributes is almost equal. These types of datasets are particularly difficult to learn from due to the lack of samples, as opposed to attributes describing them, as well as the risk of training a model that is over-fit. However, this type of machine learning problem does not affect the performance of support vector machines because the problem is scaled to the training set size rather than the attribute space dimension. Also, overfitting can be controlled with the SVM penalty parameter, C, which handles the trade-off of having a maximal margin and minimal error. This trait is shown by the results in Table 4, where our SVR algorithms perform the best when learning from the sparse datasets. For both OES datasets, the SVRCC algorithm performs the best. For the M5SPEC, MP5SPEC, and MP6SPEC datasets, SVR and SVRRC performed the best, with SVRCC's results being second best.

talk about this in the introduction too, also include that dimensionality is no issue for SVMs

Also note, for these datasets, the performance difference between our proposed algorithms against the others. The best performances achieved by the competing algorithms are at about 0.51, while our approaches reduce the error significantly. In regards to the sparse ATP1d and ATP7d datasets which also contain categorical attributes; the performance of algorithms that cannot process categorical data, depends highly on the preprocessing techniques used to transform these categorical variables into numeric ones. Regression trees are able to use categorical data, while support vector machines cannot. This is the most likely reason why our contributions did not perform as well as the competing algorithms on the Airline Ticket Price datasets. However, even with this disadvantage, our proposed methods surpass the others performances on 12 of the 19 datasets, and specifically, SVRCC performs the first and second best on 8 datasets.

The results presented in this section show that the SVRCC method outperforms the rest of the models. It has a consistent lower aRRMSE average value compared to the competing methods. This is a strong indicator that if the targets are correlated, using a single maximum correlation chain to build a single chained model will benefit prediction accuracy. Another benefit of using the SVRCC method over the ensemble of random chains, is the time taken to train the model. Training is single chained model, with a known and calculated maximum correlation chain will outperform building an ensemble of random chains in the context of speed and accuracy. Moreover, the time taken to predict the output of an unknown instance would be greatly improved due to the fact that one SVR model is trained and tested, rather than an ensemble of 10 random chains.

5.1. Statistical analysis

In order to compare the performances of the multiple models, non-parametric statistical tests are used to validate the experiments results obtained [12, 17].

To determine whether significant differences exist among the performance and results of the algorithms, the Iman-Davenport non-parametric test is run [18]. It is applied to rank the algorithms over the datasets used, according to the chi-squared distribution. The average ranks obtained by each method in the Friedman test are shown in Table 4. The Iman-Davenport test, which follows the F-distribution, is conducted to find significant differences among algorithms. The Iman-Davenport statistic, distributed according to the F-distribution with 8 and 144 degrees of freedom is 4.66, with a p-value of $4.3730E^{-5}$. The p-value obtained by the Iman-Davenport test is significantly less than 0.01, implying statistical confidence larger than 99%. Therefore, the test rejected the null-hypothesis and it can be said that

Table 5: Bonferroni-Dunn for aRRMSE

	$z = (R_0 - R_i)/SE$	$p - value$
MORF	4.9165	$1.0000E^{-6}$
MTS	3.1987	$1.3810E^{-3}$
MTSC	3.0802	$2.0690E^{-3}$
ERC	3.0802	$2.0690E^{-3}$
ST	3.0210	$2.5200E^{-3}$
ERCC	2.0140	$4.4011E^{-2}$
SVR	1.6586	$9.7201E^{-2}$
SVRRC	1.4216	$1.5513E^{-1}$

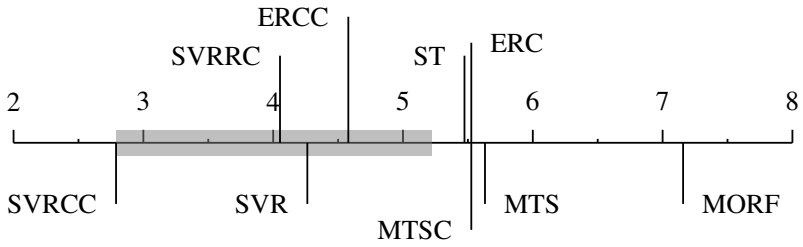

Figure 1: Bonferroni-Dunn test for aRRMSE

there exist statistically significant differences between the aRRMSE results of the algorithms.

The Iman-Davenport test indicates statistically significant differences, then the Bonferroni-Dunn post-hoc test [15] is used to find these differences that occur between the algorithms. The test assumes that the two classifiers performances are significantly different if their average ranks differ by at least some critical value [19]. Table 5 shows the results of the Bonferroni-Dunn test on the aRRMSE rate with $\alpha = 0.05$, with the control algorithm being SVRCC. Figure 1 represents the values that are proportional to the mean rank obtained by each algorithm. The critical difference value for our control, 2.43, is represented in Figure 1 as the gray rectangle. The algorithms that are to the right of the critical difference rectangle and our control algorithm SVRCC, are the ones with significantly different results. Therefore, the algorithms beyond the critical difference are significantly worse. Table 5 shows the p-values obtained by applying the Bonferroni-Dunn pro-

Table 6: Wilcoxon Test for aRRMSE

SVRCC vs.	R^+	R^-	p-value
SVR	175	15	$5.2260E^{-4}$
SVRRC	154	36	$1.5972E^{-2}$
MORF	179	11	$2.0980E^{-4}$
ST	161	29	$6.1800E^{-3}$
MTS	140	50	$7.2840E^{-2}$
MTSC	161	29	$6.1800E^{-3}$
ERC	162	28	$5.3300E^{-3}$
ERCC	155	35	$1.4070E^{-2}$

cedure over the results obtained by the Friedman procedure, and it rejects those hypotheses that have an unadjusted p-value $\leq 6.2500E^{-3}$. From Table 5 and from Figure 1, it can be observed that algorithms MORF, MTS, MTSC, ERC, and ST perform significantly worse. The insignificance of the results for algorithms SVR and SVRRC is due to the fact that they are algorithms with a similar base, which uses support vector regression. The same can be said for ERCC, where correlation chains have also been used, except that in SVRCC, a maximum chain is used. The results in Table 5, obtained by the post-hoc Bonferroni-Dunn test, show that our control algorithm, SVRCC, surpasses 5 out of 8 algorithms.

Table 6 shows the results of the Wilcoxon rank-sum test [40] for aRRMSE to compute multiple pairwise comparisons among the proposed algorithms and the other competing methods. MTS performs the best among the competing methods, but SVRCC outperforms it with a confidence of approximately 92.7160%. Also note that the results of the Wilcoxon test show that there are significant differences between SVRCC and ERCC's performance, where SVRCC succeeds a larger number of times than ERCC.

This section showed the results of the experimental study with respect to the performance of our SVRCC algorithm's performance against competing multi-target methods, using the aRRMSE measure. The proposed SVRCC is generally better than the other methods, where it has the lowest average aRRMSE rate, which is statistically significant against the competing methods. We could not say that the ERCC performed worse on all counts, but looking at the aRRMSE results, we can see that it never performed the best on any of the datasets, and as the Wilcoxon test showed, it performed significantly worse. Also, considering the

randomness of the chains created, along with the limited number that can be tested on, there is no guarantee that it will perform consistently with different numbers of multi-target output variables.

6. Conclusion

The experiments performed in this study investigate the multi-target regression problem, where a model is trained using a multi-dimensional input space in order to predict a multi-dimensional output space. Despite the potential use of this type of learning, multi-target regression has been investigated less than its classification counterpart, multi-label classification.

It has been shown that *algorithm adaptation* methods perform better than *problem transformation* methods due to their ability to capture relationships between target variables, which contributes positively to model training. An example of an *algorithm adaptation* method would be building a regression model using ensembles of random chains of the target variables. This method however, is computationally complex, and might not capture the true correlation of the target variables.

To address this problem, this investigation introduces three novel approaches, which use a base model of Support Vector Regression and explore various techniques for solving multi-target regression problems, ensuring the correlation of the target variables is captured. Their performances are then compared to 6 competitive *algorithm adaptation* and *problem transformation* methods.

The first method takes a *problem transformation* approach, which generates m single-target models, each trained independently of each other. This approach works well, but does not take the possible correlations between the target variables into account while training the m models. This lead to our second contribution, SVRRC, which creates ensemble chained model based on 10 random generated orders of the target variables. By using these differently ordered chains, a single ensemble model can be built based on the possible correlations of the target variables. Due to this methods computational complexity, along with the fact that the, at most 10, randomly generated chains may not represent the targets' correlations, we proposed our final method, SVRCC. This method eliminates the computational complexity for creating chained models, and it also ensures that the relationship between the target variables is fully exploited. This relationship is captured by creating a *maximum correlation chain*, where the targets are ordered in the direction of decreasing correlation. By doing this, each chained target attribute will positively contribute to the next one's prediction.

The experiments in this paper were all conducted using the Mulan framework, MTRegressor, as well as LIBSVM's Epsilon-SVR implementation of Support Vector Regressors. Firstly, they show the superior performance of using the SVR method as a baseline model, rather than regression trees, for Multi-Target regression. They also show an increase in performance when chaining is used to develop an ensemble model, SVRRC, using random chains. This portrays the importance of the relationship among the target variables, when training a regression model. Finally, the results show the superiority of using the SVRCC method, which has the lowest average aRRMSE value for performing 10-fold cross-validation on 19 datasets. It performed better than the single-target SVR model and the randomly chained ensemble model SVRRC, indicating that the targets' maximum correlation contributed to SVRCC's performance. In some cases, SVR performed better, which indicates that the targets have minimal correlation to one another. The statistical analysis of these results show the statistical significance of the error rates obtained by our experiments. They showed that statistically significant differences existed between 5 out of 9 algorithms, where the other 4 were our proposed methods along with the ERCC proposed by *Spyromitros et. al.*. The differences were minimal for these 4 due to their similar nature; our proposed methods all used support vector regression and the ERCC method used and ensemble of random chains. This, however, does not dismiss the findings for SVRCC. It's competitive accuracy, as well as speed (due to using a single chained model), show that it is a powerful learning algorithm for multi-target problems.

Acknowledgment

This research was supported by the Spanish Ministry of Economy and Competitiveness, project TIN2014-55252-P, and by FEDER funds.

References

[1] T. Aho, B. Zenko, S. Dzeroski, T. Elomaa, Multi-Target Regression with Rule Ensembles, Journal of Machine Learning Research 13 (2012) 2267–2407.

[2] A. Appice, S. Dzeroski, Stepwise induction of multi-target model trees, European Conference of Machine Learning Lecture Notes on Artificial Intelligence 4701 (2007) 502–509.

[3] J. Baxter, A bayesian/information theoretic model of learning to learn via multiple task sampling, Machine Learning 28 (1997) 7–39.

[4] S. Ben-David, R. Schuller, Exploiting task relatedness for multiple task learning, In: Proceedings of the Sixteenth Annual Conference on Learning Theory (2003) 567–580.

[5] H. Borchani, G. Varando, C. Bielza, P. Larrañaga, A survey on multi-output regression, WIREs Data Mining Knowledge Discovery 5 (2015) 216–233.

[6] L. Breiman, Bagging predictors, Machine Learning 24 (1996) 123–140.

[7] A. Cano, J. Luna, E. Gibaja, S. Ventura, LAIM discretization for multi-label data, Information Sciences 330 (2016) 370–384.

[8] R. Caruana, Multitask learning, Machine Learning 28 (1997) 41–75.

[9] C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology 2 (2011) 27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[10] F. Charte, A. Rivera, M. Del Jesus, F. Herrera, LI-MLC: A label inference methodology for addressing high dimensionality in the label space for multilabel classification, IEEE Transactions on Neural Networks and Learning Systems 25 (2014) 1842–1854.

[11] C. Cortes, V. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.

[12] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation 1 (2011) 3–18.

[13] Y. Ding, L. Cheng, W. Pedrycz, K. Hao, Global nonlinear kernel prediction for large data set with a particle swarm-optimized interval support vector regression, IEEE Transactions on Neural Networks and Learning Systems 26 (2015) 2521–2534.

[14] H. Drucker, C. Burges, L. Kaufman, A. Smola, V. Vapnik, Support vector regression machines, In: Proceedings of the Advances in Neural Information Processing Systems (1997) 155–161.

[15] O. Dunn, Multiple comparisons among means, J. Amer. Stat. Assoc. 56 (Mar. 1961) 52–64.

[16] J. Friedman, T. Hastie, R. Tibshirani, Regularization paths for generalized linear models via coordinate descent, Journal of Statistical Software 33 (2010) 1–22.

[17] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power, Information Sciences 180 (2010) 2044–2064.

[18] S. García, F. Herrera, An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons, Journal of Machine Learning Research 9 (2008) 2677–2694.

[19] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour - a case study on the CEC2005 Special Session on Real Parameter Optimization, Heuristics 15 (2008) 617–644.

[20] E. Hadavandi, J. Shahrabi, S. Shamishirband, A novel Boosted-neural network ensemble for modeling multi-target regression problems, Engineering Applications of Artificial Intelligence 45 (2015) 204–219.

[21] E. Hadavandi, K. Shahrabi, Y. Hayashi, SPMoE: a novel subspace-projected mixture of experts model for multi-target regression problems, Soft Computing 20 (2016) 2047–2065.

[22] J. He, H. Gu, Z. Wang, Multi-instance multi-label learning based on gaussian process with application to visual mobile robot navigation, Information Sciences 190 (2011) 162–177.

[23] M. Jeong, G. Lee, Multi-domain spoken language understanding with transfer learning, Speech Communication 51 (2009) 412–424.

[24] V. Kecman, Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models, MIT Press, 2001.

[25] D. Kocev, M. Ceci, Ensembles of Extremely Randomized Trees for Multi-target Regression, Lecture Notes in Computer Science 9356 (2015) 86–100.

[26] D. Kocev, S. Dzeroski, M. White, G. Newell, P. Griffioen, Using single- and multi-target regression trees and ensembles to model a compound index of vegetation condition, Ecological Modelling 20 (2009) 1159–1168.

[27] D. Kocev, C. Vens, J. Struyf, S. Dzeroski, Ensembles of Multi-Objective Decision Trees, Springer, Heidelberg (2007) 86–100.

[28] D. Kocev, C. Vens, J. Struyf, S. Dzeroski, Tree ensembles for predicting structured outputs, Pattern Recognition 43 (2013) 817–833.

[29] J. Lee, D.-W. Kim, Memetic feature selection algorithm for multi-label classification, Information Sciences 293 (2015) 80–96.

[30] H. Li, D. Li, Y. Zhai, S. Wang, J. Zhang, A novel attribute reduction approach for multi-label data based on rough set theory, Information Sciences 367-368 (2016) 827–847.

[31] M. Lichman, UCI machine learning repository, 2013. URL: `http://archive.ics.uci.edu/ml`.

[32] Q. Liu, Q. Xu, V. Zheng, H. Xue, Z. Cao, Q. Yang, Multi-task learning for cross-platform sirna efficacy prediction: an in-silico study, BMC Bioinformatics 11 (2010) 181–196.

[33] F. Pérez, G. Camps, E. Soria, J. Pérez, A. Figueiras, A. Artés, Multidimensional function approximation and regression estimation, Artificial Neural Networks (2002) 757–762.

[34] B. Qian, X. Wang, J. Ye, I. Davidson, A reconstruction error based framework for multi-label and multi-view learning, IEEE Transactions on Knowledge and Data Engineering 27 (2015) 594–607.

[35] J. Read, C. Bielza, P. Larranaga, Multi-dimensional classification with super-classes, IEEE Transactions on Knowledge and Data Engineering 26 (2014) 1720–1733.

[36] E. Spyromitros-Xioufis, G. Tsoumakas, W. Groves, I. Vlahavas, Multi-Label Classification Methods for Multi-Target Regression, Cornell University Library (2014).

[37] E. Spyromitros-Xioufis, G. Tsoumakas, W. Groves, I. Vlahavas, Multi-target regression via input space expansion: Treating targets as inputs, Machine Learning 104 (2016) 55–98.

[38] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, I. Vlahavas, Mulan: A java library for multi-label learning, Journal of Machine Learning Research 12 (2011) 2411–2414.

[39] G. Tsoumakas, E. Spyromitros-Xioufis, A. Vrekou, I. Vlahavas, Multi-target regression via random linear target combinations, Machine Learning and Knowledge Discovery in Databases 8726 (2014) 225–240.

[40] F. Wilcoxon, Individual comparisons by ranking methods, Biometr. Bull. 1 (Dec. 1945) 80–83.

[41] I. Witten, E. Frank, M. Hall, Data Mining: Practical Machine Learning Tools and Techniques, 3rd edition ed., Morgan Kaufmann, 2011.

[42] Q. Wu, Y. Ye, H. Zhang, T. Chow, S.-S. Ho, ML-TREE: A tree-structure-based approach to multilabel learning, IEEE Transactions on Neural Networks and Learning Systems 26 (2015) 430–443.

[43] T. Xiong, Y. Bao, Z. Hu, Multiple-output support vector regression with a firefly algorithm for interval-valued stock price index forecasting, Knowledge-Based Systems 55 (2014) 87–100.

[44] S. Xu, X. An, X. Qiao, L. Zhu, L. Li, Multi-output least-squares support vector regression machines, Pattern Recognition 34 (2013) 1078–1084.

[45] M.-L. Zhang, Z.-H. Zhou, A review on multi-label learning algorithms, IEEE Transactions on Knowledge and Data Engineering 26 (2014) 1819–1837.

[46] Y.-P. Zhao, K.-K. Wang, F. Li, A pruning method of refining recursive reduced least squares support vector regression, Information Sciences 296 (2015) 160–174.