UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Gabriel Ribeiro Lencione

Online Multi-task Learning for Regression Problems: New Methods and Applications

Aprendizado Multitarefa Recursivo para Problemas de Regressão: Novos Métodos e Aplicações

Campinas
2024

Gabriel Ribeiro Lencione

Online Multi-task Learning for Regression Problems: New Methods and Applications

Aprendizado Multitarefa Recursivo para Problemas de Regressão: Novos Métodos e Aplicações

Dissertation presented to the School of Electrical and Computer Engineering of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Electrical Engineering, in the area of Computer Engineering.

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na Área de Engenharia de Computação.

Supervisor: Prof. Dr. Fernando José Von Zuben

Este trabalho corresponde à versão final da dissertação defendida pelo aluno Gabriel Ribeiro Lencione, e orientada pelo Prof. Dr. Fernando José Von Zuben.

Campinas
2024

Ficha catalográfica
Universidade Estadual de Campinas (UNICAMP)
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

Lencione, Gabriel Ribeiro, 1996-
L545o Online multi-task learning for regression problems : new methods and applications / Gabriel Ribeiro Lencione. – Campinas, SP : [s.n.], 2024.

Orientador: Fernando José Von Zuben.
Dissertação (mestrado) – Universidade Estadual de Campinas (UNICAMP), Faculdade de Engenharia Elétrica e de Computação.

1. Inteligência artificial. 2. Multitarefa (Computação). 3. Funções de Kernel. 4. Aprendizado de máquina. I. Von Zuben, Fernando José, 1968-. II. Universidade Estadual de Campinas (UNICAMP). Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações Complementares

Título em outro idioma: Aprendizado multitarefa recursivo para problemas de regressão : Novos métodos e aplicações
Palavras-chave em inglês:
Artificial Intelligence
Multi-task computing
Kernel functions
Time series analysis
Machine learning
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Fernando José Von Zuben [Orientador]
Marcos Medeiros Raimundo
Emely Pujólli da Silva
Data de defesa: 08-08-2024
Programa de Pós-Graduação: Engenharia Elétrica

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-2307-7380
- Currículo Lattes do autor: http://lattes.cnpq.br/5558893492692375

Comissão Julgadora - Dissertação de Mestrado

Candidato: Gabriel Ribeiro Lencione **RA:** 168116
Data de defesa: 08 de agosto de 2024
Título da Tese em inglês: Online Multi-task Learning for Regression Problems: New Methods and Applications
Título da Tese: Aprendizado Multitarefa Recursivo para Problemas de Regressão: Novos Métodos e Aplicações

Prof. Dr. Fernando José Von Zuben (FEEC/Unicamp)
Prof. Dr. Marcos Medeiros Raimundo (IC/Unicamp)
Prof. Dra. Emely Pujólli da Silva (IC/Unicamp)

A Ata de Defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no SIGA (Sistema de Fluxo de Dissertação/Tese) e na Secretaria de Pós-Graduação da Faculdade de Engenharia Elétrica e de Computação.

Dedication

This dissertation is dedicated to my lovely wife Maria, whose support, affection and encouragement helped me to find strength in my weakest moments. I thank my parents for always asking me when I was going to graduate and I thank my sister Melissa for pointing out that I was not doing anything to finish it - she was right sometimes. Following this program while working full-time at Itaú-Unibanco was a bit of a challenge and that is why I also dedicate this dissertation to Professor Fernando, for his patience and support, and to all my friends and colleagues. Thank you.

Acknowledgments

The Itaú-Unibanco conceded up to eight hours per week to the dedication to this master's program and contributed financially to the student's participation in the IEEE-WCCI 2022 in Italy.

Resumo

Esta dissertação apresenta duas novas abordagens para problemas de regressão de Aprendizado Multitarefa (MTL, em inglês) recursivo. Utilizamos uma formulação MTL baseada em grafos de alto desempenho e desenvolvemos duas versões recursivas alternativas baseadas nas estratégias *Weighted Recursive Least Squares* (WRLS) e *Online Sparse Least Squares Support Vetor Regression* (OSLSSVR). Adotando transformações de empilhamento de tarefas, demonstramos a existência de uma única matriz que incorpora a relação de múltiplas tarefas e fornece informação estrutural a ser incorporada pelo método MT-WRLS no seu procedimento de inicialização e pelo MT-OSLSSVR na sua função de kernel multitarefa. Contrastando com a literatura existente, que se baseia principalmente na Descida de Gradiente Online (OGD, em inglês) ou em abordagens cúbicas inexatas, obtemos recursões exatas e aproximadas com um custo quadrático por instância na dimensão do espaço de entrada (MT-WRLS) ou na dimensão do dicionário de instâncias (MT-OSLSSVR). Comparamos os nossos métodos MTL online com outros concorrentes num estudo de caso real de previsão da velocidade do vento, evidenciando o ganho significativo no desempenho de ambas as abordagens propostas.

No entanto, demonstra-se aqui que o OSLSSVR recentemente proposto contém passos inconsistentes na sua formulação recursiva, o que poderia ter impedido a obtenção de resultados adequados com a nossa proposta MT-OSLSSVR. Em seguida, reformulamos a proposta OSLSSVR e consideramos um estudo de caso de regressão online como referência, capaz de revelar os efeitos prejudiciais dos erros detectados na formulação. Para além disso, a versão reformulada do OSLSSVR mostra-se operacional, exibindo um elevado desempenho e uma estimação online numericamente estável.

Com o objetivo de empregar a formulação MTL online em uma aplicação multitarefa adequada e dependente do tempo, este trabalho também estende o mecanismo de aprendizagem recentemente concebido chamado EVeP (*Extreme Value evolving Predictor*), um preditor evolutivo baseado em regras *fuzzy* e fundamentado em procedimentos inovadores para definir as partes antecedentes e consequentes das regras *fuzzy* existentes. No EVeP, os grânulos de informação são recursivamente atualizados e associados a distribuições Weibull, generalização das distribuições gaussianas que incorpora estatísticas mais robustas para estabelecer a região de influência de cada regra fuzzy. A informação partilhada de todas as regras, numa formulação multitarefa, é adotada para definir os parâmetros consequentes no EVeP. Dado que a formulação multitarefa é resolvida utilizando a aprendizagem em batelada e a descida do gradiente, o custo computacional por iteração tende a ser elevado, o que constitui uma preocupação nas aplicações práticas. Por isso, aqui a estrutura multi-tarefa na parte consequente das regras foi revista para incorporar a otimização convexa online, dando origem ao EVeP_OCO. Agora, as partes antecedentes e consequentes das regras são atualizadas de forma totalmente recursiva, com uma clara redução da carga computacional por iteração, particularmente quando são considerados os piores cenários: o custo por iteração depende do número atual de regras a atualizar. Os estudos de caso são compostos por uma variedade de problemas de previsão de séries temporais de referência e demonstram o ganho significativo em termos de custo computacional por iteração, com uma redução admissível do desempenho, ao substituir um procedimento de aprendizado multitarefa em batelada por um recursivo equivalente.

Abstract

This dissertation introduces two novel approaches for online Multi-Task Learning (MTL) Regression Problems. We employ a high performance graph-based MTL formulation and develop two alternative recursive versions based on the Weighted Recursive Least Squares (WRLS) and the Online Sparse Least Squares Support Vector Regression (OSLSSVR) strategies. Adopting task-stacking transformations, we demonstrate the existence of a single matrix incorporating the relationship of multiple tasks and providing structural information to be embodied by the MT-WRLS method in its initialization procedure and by the MT-OSLSSVR in its multi-task kernel function. Contrasting the existing literature, which is mostly based on Online Gradient Descent (OGD) or cubic inexact approaches, we achieve exact and approximate recursions with quadratic per-instance cost on the dimension of the input space (MT-WRLS) or on the size of the dictionary of instances (MT-OSLSSVR). We compare our online MTL methods to other contenders in a real-world wind speed forecasting case study, evidencing the significant gain in performance of both proposed approaches.

However, the recently proposed OSLSSVR is demonstrated here to contain inconsistent steps in its recursive formulation, which could have prevented us from achieving appropriate results with our MT-OSLSSVR proposal. We then reformulate the OSLSSVR proposal and consider an online regression case study as a benchmark, capable of revealing the harmful effects of the detected mistakes in the formulation. Besides, the reformulated version of OSLSSVR is shown to operate accordingly, exhibiting a high performance and numerically stable online estimation.

Aiming at employing online MTL in a suitable time-dependent multi-task application, this work also extends the recently conceived learning mechanism called EVeP (Extreme Value evolving Predictor), an evolving fuzzy-rule-based predictor characterized by innovative procedures to define the antecedent and consequent parts of the existing fuzzy rules. In EVeP, information granules are recursively updated and associated with Weibull distributions, a generalization of Gaussian distributions which incorporates more robust statistics to establish the region of influence of each fuzzy rule. Shared information from all the rules, in a multi-task formulation, is adopted to set the consequent parameters in EVeP. Given that the multi-task formulation is solved using batch learning and gradient descent, the computational cost per iteration tends to be high, being a concern in practical applications. Therefore, here the multi-task framework at the consequent part of the rules was revised to incorporate online convex optimization, given rise to EVeP_OCO. Now, antecedent and consequent parts of the rules are updated in a fully recursive way, with a clear reduction in the computational burden per iteration, particularly when the worst case scenarios are considered: the cost per iteration depends on the current number of rules to be updated. The case studies are composed of a variety of benchmark time series prediction problems. They demonstrate the significant gain in terms of computational cost per iteration, with an admissible reduction in performance by replacing a batch multi-task learning procedure by an online counterpart.

List of Figures

List of Tables

List of Acronyms

ML	Machine Learning
MTL	Multi-task Learning
STL	Single-task Learning
OL	Online Learning
OCO	Online Convex Optimization
OGD	Online Gradient Descent
ADMM	Alternating Direction Method of Multipliers
WRLS	Weighted Recursive Least Squares
MT-WRLS	Multi-task Weighted Recursive Least Squares
SVM	Support Vectors Machine
SVR	Support Vectors Regression
LSSVR	Least Squares Support Vectors Regression
KRLS	Kernel Recursive Least Squares
OSLSSVR	Online and Sparse Least Squares Support Vectors Regression
MT-OSLSSVR	Multi-task Online and Sparse Least Squares Support Vectors Regression
ELM	Extreme Learning Machines
EVeP	Extreme Value evolving Predictor
EVeP_OCO	Extreme Value evolving Predictor with Online Convex Optimization
CD	Concept Drift
ALD	Approximate Linear Dependency
RMSE	Root Mean Squared Error
RELRMSE	Relative Root Mean Squared Error
NDEI	Nondimensional Error Index
FNorm	Frobenius Norm

Contents

1 Notes on the Text

This dissertation is deeply inspired and contains fragments of the papers [1, 2, 3] throughout its entire text. These papers resulted directly from the research developed in this master's program and have been published [1] or submitted to journals [2, 3].

2 Introduction and Objectives

The success of computer-based inductive learning methods, well represented by the so-called Machine Learning (ML) field, has enabled the progress of its applications and its own development. With new and more promising techniques comes the desire to employ them in real-world problems, from which more challenges emerge, forcing the techniques to advance even more. One of the famous ML branches, Multi-task Learning (MTL) consists in exploiting the relationship among different tasks, sharing knowledge during the learning process, to improve the capability of generalization of the individual tasks. There is a heterogeneous group of techniques under the MTL designation with considerable evidence pointing to the benefits of its adoption in a great variety of applications.

However, when it comes to the applications of ML methods in real-world problems, we face time-dependent conditions of data that require the models to dynamically adapt in order to avoid drastic performance reduction. These conditions may be, for example, the incremental availability of data to feed the learning algorithms in data streams, often with few or no examples prior to their deployment, or the changes in data distribution that may occur in different manners as well (the so-called Concept Drift and Concept Shift). However, many of the ML (or MTL) pioneer methods were proposed in a batch-fashion learning scheme, in which all the training data are available to be used at once. This type of learning procedure is usually too expensive to be executed at each arrival of a new instance, as for the data stream example, opening the way for the development of recursive methods.

There are, in the literature of Multi-task Learning, few papers that propose recursive MTL methods, specially for regression problems. As shall be better detailed in the next sections, the existing techniques suffer either from slow convergence, resulting in bad approximation to the optimal solution, such as linear algorithms, or from high per-instance cost, sometimes almost as expensive as the batch methods. It is even more difficult to find online MTL proposals that implement a nonlinear input-output mapping. The adoption of nonlinearities in ML methods is intended to offer to models more flexibility to fit data that may not be well represented by linear models. The potential benefits of nonlinear Multi-Task Learning have already been outlined in the literature [4].

This research comes to fulfill these gaps, with the proposal of two alternative methods for recursive MTL that present better compromise between convergence and computational cost (specially per-instance). We explore the adaptation of two well-known recursive single-task methods to work within a well-succeeded MTL framework. The famous Weighted Recursive Least Squares (WRLS) and the prominent Online Sparse Least Squares Support Vector Regression (OSLSSVR) are recursive methods that deal with the same regression problem in the primal and dual spaces, respectively. We demonstrate that a simple modification to the initial conditions of the WRLS, and the appropriate choice for the kernel of the OSLSSVR enable us to achieve two alternative recursive MTL methods with quadratic per-instance complexity and immediate or parameterized convergence. We also combine one of our proposals with Extreme Learning Machines (ELM) [5], thus conceiving an online MTL method with nonlinear input-output mapping.

However, crucial inconsistencies are present in the original formulation of the OSLSSVR [6], specifically in the recursive formulas of the approximation to the optimal solution, which was found to produce results far from the expected and, ultimately, harmful instability of the predictors. They do not reduce the conceptual relevance of the OSLSSVR proposal, but motivate the necessary corrections that will ultimately lead to conformable results. In this research, we point out the issues in the OSLSSVR original proposal, fix its formulation, and demonstrate that proper results can only be obtained with the corrected version, which is not only necessary to produce stable and high performance results (the original work, even with some inconsistencies has already been demonstrated to outperform competitors in the literature), but also to enable its correct application in other regularization-dependent fields, such as our kernel-based online MTL proposal.

Furthermore, we propose the application of Online MTL to a state-of-the-art Evolving fuzzy system. Evolving fuzzy and neuro-fuzzy systems have been proposed and studied in the past decades aiming to provide a solid framework capable of dealing with the major particularities of data streams. The basic idea behind most of them [7] is the ability to evolve both the structure and the parameters continuously, adapting themselves in response to gradual or abrupt changes in data distribution. One of the most prominent exponents among the evolving fuzzy rule-based systems is the Extreme Value evolving Predictor (EVeP) [8], capable of achieving high performance on synthetic and real-world data flows, mainly due to the benefit of shared information when determining the consequent part of the existing rules, by means of a multi-task learning (MTL) [9] approach. At each incoming data instance, the adopted formulation in [8] is recursive when adjusting the information granules of the rules (structure). However, for every update of the information granules, the consequent parameters of all the rules are fully retrained from scratch, in an MTL configuration solved by gradient descent. This batch strategy for the consequent parameters foments the high performance of the evolving system, but at the cost of a greater computational burden per iteration.

This dissertation, therefore, aggregates three separate main objectives that are connected under the motivation to develop and apply online MTL methods for regression problems: (1) identify and correct the issues present in the original formulation of the OSLSSVR method, as it is an elementary part of our research, crucial to the development of a recursive kernel-based MTL method; (2) present two alternative online MTL proposals with better compromise between convergence and computational cost; (3) and apply an appropriate online MTL method to reduce the burden of updating the consequent parameters of the EVeP. For each of these three parts, we elaborate an experimental setup intended to either validate or assess our proposals in synthetic and real-world benchmarks, comparing to other contendenrs in literature.

We outline our main contributions as follows:

1. Our work demonstrates the existence of misconceptions in the original formulation of the OSLSSVR, a promising kernel-based online learning method that enables online optimization of regularized regression problems in the dual space.

2. We propose the corrected recursive formulation of the OSLSSVR while preserving its original properties, such as the complexity in the number of data examples $O(m_t^2)$, $m_t <<$

N.

3. We compare the original and the corrected methods on a regression time-dependent benchmark and demonstrate that the theoretical expected results are only achieved by our proposal, abolishing the harmful instability presented in the original formulation and therefore profiting from better performance.

4. Our research introduces the Multi-Task Weighted Recursive Least Squares (MT-WRLS), which recursively solves the adopted Graph-based MTL formulation in the primal space, producing the exact solution at each step, thus with immediate convergence, at the per-instance cost of $O(d^2 \times T^2)$.

5. We are the first, to the best of our knowledge, to propose a recursive kernel-based MTL method, the Multi-Task Online and Sparse Least Square Support Vector Regression (MT-OSLSSVR). The approximation to the exact solution can be controlled by its sparsity parameter, presenting a per-instance complexity of $O(d \times m_n^2)$ ($m_n << n$, n being the number of examples).

6. Both methods in contributions 4 and 5 are demonstrated to have better theoretical results than most of the existing techniques in literature. With them, we achieve a reasonable per-instance complexity with much stronger guarantees of convergence.

7. The combination of our MT-WRLS proposal with Extreme Learning Machines enabled the development of a convex online MTL method with nonlinear input-output mapping.

8. We test our proposals on a real-world time-dependent benchmark of wind speed forecasting and compare them to other contenders in the literature, empirically demonstrating their superior performance.

9. We successfully convert EVeP to a fully recursive formulation, employing online MTL when learning EVeP consequent parameters of a fuzzy rule-based evolving system. The proposal is successfully validated considering real and synthetic benchmark time series, including comparison with the literature.

This dissertation is divided as follows: In section 3, we present a literature review of the online MTL field, as well as a brief review of the adoption of MTL in Evolving Fuzzy System and of kernel-based recursive methods. Section 4 is devoted to identifying the issues in the original OSLSSVR and proposing a new formulation, with experimental validation. In section 5, we elaborate a formal description of our proposals, followed by their experimental methodology and the presentation and discussion of our results, assessing the achievement of our objectives through comparative analysis. Section 6 provides our online EVeP formulation with the description of its experimental setup, followed by the assessment of our results, including a detailed comparison to other contenders in literature. Section 7 outlines concluding remarks and indicates the future steps of our research.

3 Literature Review

3.1 Multi-task Learning

The combined treatment of multiple tasks is known in the literature as Multi-task Learning [10, 11]. If the task relatedness is properly incorporated into the multiple learning models, generally as a regularization step, information sharing is then promoted, possibly with a positive impact on the average generalization capability of the multiple tasks. Particularly when the size of the training dataset is reduced, the MTL approach tends to exhibit a distinguished performance in comparison to learning the tasks independently (also known as Single-task Learning or STL).

Although different methods have been proposed for modeling the task relatedness, many of them can be assigned under the so-called Structural Regularization [12], in which we take a usual loss function to be minimized over all the tasks, and add a regularization term. The regularization term generally incorporates the structural relationship among the tasks, thus promoting information sharing. One of the greatest advantages of this approach is that, since it works basically with linear models, most of the final optimization structures are convex, simplifying the training process [9, 12, 13].

Then, let $\mathbf{X}_t \in \mathbb{R}^{n_t \times d}$ be the input matrix for task $t \in \{1, 2, \ldots, T\}$, containing n_t training examples of dimension d and let $\mathbf{y}_t \in \mathbb{R}^{n_t}$ be the respective target vector. The linear multi-task regression problem is obtained when we attempt to estimate the matrix $\mathbf{\Theta} \in \mathbb{R}^{d \times T}$, in which its columns $\mathbf{w}_t \in \mathbb{R}^d$ represent the individual parameter vectors, in such a manner that the task predictions are given by $\hat{\mathbf{y}}_t = \mathbf{X}_t \mathbf{w}_t$. We can therefore state the multi-task learning problem as:

$$\mathbf{\Theta} = \arg\min_{\mathbf{\Theta}} \mathcal{J}(\mathbf{\Theta}), \mathcal{J}(\mathbf{\Theta}) = \mathcal{L}(\mathbf{\Theta}) + \mathcal{R}(\mathbf{\Theta}) \tag{1}$$

where $\mathcal{L}(\mathbf{\Theta})$ is the common loss function, being most frequently adopted in regression problems as the mean squared error (MSE), which is taken on average over all the T tasks:

$$\mathcal{L}(\mathbf{\Theta}) = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{N_t} \|\mathbf{y}_t - \mathbf{X}_t \mathbf{w}_t\|_2^2, \tag{2}$$

while $\mathcal{R}(\mathbf{\Theta})$ is the regularization term that seeks to incorporate task relatedness during training.

Although there are various proposed regularizers in the literature, we present three of the most consolidated, motivated by their flexibility and ease of handling:

- Argryou, Evgeniou & Pontil [12] proposed to employ the $\ell_{1,2}$ norm of $\mathbf{\Theta}$ as the regularization term, which induces sparsity to the learning model by vanishing entire rows of the parameters matrix. This is sometimes also referred to as group LASSO regularization because of its row-based analogous behaviour. The associated optimization problem becomes:

$$\mathbf{\Theta} = \arg\min_{\mathbf{\Theta}} \mathcal{L}(\mathbf{\Theta}) + \lambda \|\mathbf{\Theta}\|_{1,2}, \tag{3}$$

where λ is the hyperparameter controlling the penalty imposed upon the regularization term.

- Ji & Ye [14] supposed that all the tasks share a parameter subspace of reduced dimension. However, since the adoption of the rank of $\mathbf{\Theta}$ as the regularization term leads, in general, to an NP-hard problem [15], the authors employed the trace norm of the parameter matrix $\|\mathbf{\Theta}\|_*$, that has the property of being an envelope of $rank(\mathbf{\Theta})$. This formulation results in the following optimization problem:

$$\mathbf{\Theta} = \arg\min_{\mathbf{\Theta}} \mathcal{L}(\mathbf{\Theta}) + \lambda \|\mathbf{\Theta}\|_*, \tag{4}$$

  where λ is the respective hyperparameter, in the same manner as previously stated.

- Authors like Ayres and Von Zuben [8] used a graph-based structure to represent the pairwise task relatedness. We define the set $\mathcal{E}$ of edges $\mathbf{e}^{(j)} \in \mathbb{R}^T$, where two tasks (nodes) x and y are related in some degree if $\mathbf{e}_x^{(j)} = \alpha$ and $\mathbf{e}_y^{(j)} = -\beta$, with $\alpha, \beta > 0$. The whole graph is built upon the matrix $\mathbf{G} = [\mathbf{e}^{(1)},\ \mathbf{e}^{(2)},\ \ldots\ , \mathbf{e}^{(\|\mathcal{E}\|)}]$. The following regularization term penalizes, therefore, the weighted Euclidean distance between every connected pair, opening the possibility of asymmetric connections.

$$\mathcal{R}(\mathbf{\Theta}) = \|\mathbf{\Theta}\mathbf{G}\|_F^2 = \sum_{j=1}^{\|\mathcal{E}\|} \|\mathbf{\Theta}\mathbf{e}^{(j)}\|_2^2 \tag{5}$$

  The optimization problem becomes:

$$\mathbf{\Theta} = \arg\min_{\mathbf{\Theta}} \mathcal{L}(\mathbf{\Theta}) + \lambda_1 \|\mathbf{\Theta}\mathbf{G}\|_F^2 + \lambda_2 \|\mathbf{\Theta}\|_2, \tag{6}$$

  where the last term corresponds to the norm ℓ_2 of $\mathbf{\Theta}$ and is employed to induce sparsity to the learning model. The hyperparameters λ_1 and λ_2 control the imposed penalty to the learning process as well.

  In this method, it is important to find efficient ways of determining their degree of relationship. As it has already been done by [8], a similarity approach can be pursued, in which the weights α and β from a specific edge connecting the tasks x and y are assigned with the similarity measures $sim(x, y)$ and $sim(y, x)$.

Several papers demonstrate how the adoption of MTL can significantly improve the performance of the tasks. In the domain of climate variables prediction, we highlight the research of Gonçalves, Von Zuben and Banerjee [16], who have worked with variables such as temperature, pressure, and humidity. In order to facilitate and disseminate the use of multi-task learning via structural regularization, the MATLAB package MALSAR was proposed [9]. It implements the three methods previously described (batch formulation) under the aliases of *Least L21*, *Least Trace* and *Least SRMTL*.

3.2 Online Multi-task Learning

We introduce this subsection with a brief discussion about general Online Learning (OL) methods. Many strategies have been developed to efficiently deal with detecting concept drifts and retraining base models in data streams, but the main bottleneck persists [17]. Online Learning literature has evolved on a diverse set of branches aiming to face this challenge [18]. A non exhaustive formalism can be stated upon the regret function: let $\mathcal{J}^{[i]}$ be the loss function and let $\mathbf{\Theta}^{[i]}$ be the learnt parameters of the model at step i. The regret at time n expresses the difference between the cumulative losses of the predictions produced by $\mathbf{\Theta}^{[i]}$ and the minimal losses represented by an optimal $\mathbf{\Theta}^*$ considering the same ensemble of losses (all the points until n):

$$Regret(n) = \sum_{i=1}^{n} \mathcal{J}^{[i]}(\mathbf{\Theta}^{[i]}) - \min_{\mathbf{\Theta}^*} \sum_{i=1}^{n} \mathcal{J}^{[i]}(\mathbf{\Theta}^*) \tag{7}$$

The fundamental idea behind OL is to reduce the average regret ($Regret(n)/n$) over time. When the OL method does not provide an exact solution ($Regret(n) = 0$), which could be achieved applying batch learning at each step, the average regret is expected to start at a certain level and to asymptotically converge to zero as the online model learns the general distribution from incoming data. The final challenge of Online Learning is to discover the best (if possible) compromise between convergence and computational complexity.

One class of OL algorithms that has gained attention in the last years is the so-called Online Convex Optimization (OCO) [19]. Most of the methods belonging to that class are based on first or second order methods (linear and quadratic cost). The Online Gradient Descent (OGD) is its best exponent since it summarizes the main philosophy of this field: simplicity, online nature, low cost and good convergence rate [19]. It consists of applying a Gradient Descent iteration based on the incoming data instance as online parameters update step. Under the condition of convexity of $\mathcal{J}^{[i]}$, a convergence rate of $O(1/\sqrt{n})$ is guaranteed with a per-instance linear cost.

In [20], the authors developed an Online Multi-Task Perceptron framework for classification problems. Its optimization engine provides, for each task, a combination of the losses of all tasks, resulting in the weights of the parameters update (similar to a learning rate). The structural relationship of the tasks is not totally detailed and, although the update step possesses a linear cost in the dimension of the parameters vector, the cost of finding the update weights depends on the type of the adopted norm. The convergence to an optimal solution is bounded by $1/\sqrt{n}$.

Another linear online method was proposed by [21]. It also consists of an online Multi-task Perceptron for classification problems. The authors propose a linear update stage in which a matrix $\boldsymbol{A}$ is employed to incorporate the multi-task relatedness into the recursive optimization procedure. It is shown that, according to the choice of $\boldsymbol{A}$, the number of prediction errors produced by their classifier is bounded by a well-defined structural regularization component. The authors do not specify a convergence rate to the optimal solution. Our proposals are strongly inspired on their approach of dealing with multiple tasks by stacking the parameters

vectors into a single vector and on their multi-task kernels.

The work of [22] is the first, to the knowledge of these authors, to start from a batch graph-based multi-task optimization definition and propose a recursive procedure to approximate the online solution to the optimal one. They propose two different online methods: one profits from a constraint formulation of the optimization problem and apply one iteration of the ADMM (Alternating Direction Method of Multipliers) algorithm for each online update with a cubic cost on the size of the parameters vectors. The other is an implementation of the Online Gradient Descent, presenting a lower cost at the expense of diminished convergence rate. The computational cost of the first proposal is similar to exactly solving the batch problem at each online iteration as will be shown in the next sections. Our research addresses this per-instance burden with recursive exact solutions.

The authors of [23] propose a multi-task framework based on the epsilon tube of the Support Vector Regression (SVR) for modeling ensemble forecasts. Although they describe their method as an online learning procedure, their parameters update step is done in a batch manner with cubic complexity, penalizing the distance between new and current parameters.

More recent works based on OGD and first order algorithms were proposed by [24, 25]. They explore new structural regularizations and prove theoretical convergence improvements. One important thing to be noticed in all the cited works is either their lack of time-dependent regression case studies, or the absence of reproducible well-described methodologies, making it more difficult to compare the existing contenders in literature.

3.3 Multi-task Learning in Evolving Fuzzy Systems

Being one of the pioneers of the field of evolving Fuzzy-Rule-Based (eFRB) systems, Angelov and Filev [26] introduced the eTS (Evolving Takagi-Sugeno) system, which is the evolving extension of the traditional TS fuzzy system. By using linear polynomials at the consequent part of the rules, locally defined hyperplanes are created for each rule. Leite et al. [27] created an eFRB algorithm capable of simultaneously providing both singular and granular functions employing a linguistic description of the behavior of the system. They resorted to trapezoidal structures in the definition of the information granules.

Ayres and Von Zuben [28] [29] were the pioneers to consider, in the evolving scenario, the benefits of a Multi-task Learning (MTL) approach in the process of obtaining the TS parameters of the consequent part of the rules. They took the degree of superposition of the corresponding fuzzy granules to directly encode the structural relationship among the rules, which is used to define a regularization term that is added to the empirical loss. Ayres and Von Zuben [30] presented the Extreme Value evolving Predictor (EVeP), which resorts to a statistically consistent approach for the definition of the fuzzy rules forming the antecedent and the consequent parts of the rules. They employed fuzzy granules which capture the limiting distributions of the relative proximity among the data points supporting the rules. In this way, the information granules of the rules are indeed synthesized by a truly recursive approach. However, the TS consequent part of the rules are still obtained by batch learning, with a gradient descent procedure to solve the MTL formulation, at every updating of the information gran-

ules of the existing rules. It confers optimality to the consequent parameters, but increases the computational demand per iteration.

3.4 Online Support Vector Regression

Based on the concept of Vapnik's Support Vectors Machines (SVM) [31], the authors of [32] proposed the Support Vectors Regression (SVR) method. Instead of searching for the hyperplanes that optimizes the separation between two classes (classification SVM), the SVR seeks to minimize the ϵ-insensitive error function, which attributes error 0 if the prediction lies inside of the ϵ-tube or a positive value otherwise. Once defined a kernel function, we bring the problem to the dual space, where we do not need to define the shape of the predictor itself, turning into a quadratic programming [32] problem.

The computational burden of solving the SVR problem motivated the introduction of the LSSVR [33] method described before. While both of them possess usually the same asymptotic cost on the number of data examples ($O(N^3)$, N being the number of data examples), the later resorts to solving a linear system [33], for which very efficient exact or approximate methods are available [34]. However both are batch methods, meaning that they require all training data to be concomitantly available during the learning process, which is well suited for static problems, but they may represent a considerable barrier for online applications involving data streams, in which data is incrementally available and the learning complexity poorly scales with the number of incoming samples.

Consider a training dataset $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$, with inputs $\mathbf{x}_i \in \mathbb{R}^p$ and corresponding outputs $y_i \in \mathbb{R}$. The LSSVR method aims at finding a function $f(\cdot)$ that approximates the outputs y_i with considerable accuracy. This function is defined as:

$$f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle = \phi(\mathbf{x})^T \mathbf{w} \tag{8}$$

where $\phi(\cdot) : \mathbb{R}^p \to \mathbb{R}^{p_H}$ is a mapping (linear or nonlinear) that projects the input vectors onto some feature space, $\mathbf{w} \in \mathbb{R}^{p_H}$ is the parameters vector and $\langle \cdot, \cdot \rangle$ denotes inner product in that feature space. The learning problem is presented as the minimization of the following objective function:

$$J(\mathbf{w}) = \sum_{n=1}^{N} \left(f\left(\mathbf{x}_n\right) - y_n \right)^2 + \gamma \|\mathbf{w}\|^2 \tag{9}$$

where $\gamma \geq 0$ is the regularization coefficient controlling the penalty over the norm of the parameters vector.

The Representer Theorem [6, 35] ensures the existence of a solution that minimizes the cost function in (9) in the form of $f(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n k\left(\mathbf{x}, \mathbf{x}_n\right)$, where $k(\cdot, \cdot)$ denotes the kernel function, $k\left(\mathbf{x}, \mathbf{x}_n\right) = \langle \phi(\mathbf{x}), \phi\left(\mathbf{x}_n\right) \rangle$, and $\alpha_n \in \mathbb{R}$ is the weight for sample $\mathbf{x}_n$. From the kernel definition, the dual form of the optimization problem is given by:

$$\mathbf{w} = \sum_{n=1}^{N} \alpha_n \phi\left(\mathbf{x}_n\right) = \mathbf{\Phi} \alpha \tag{10}$$

$$\min_{\alpha} J(\alpha) = \min_{\alpha} (\|\mathbf{K}\alpha - \mathbf{y}\|^2 + \gamma \alpha^T \mathbf{K} \alpha) \tag{11}$$

where $\mathbf{K} = \mathbf{\Phi}^T\mathbf{\Phi}$, $\mathbf{y} = (y_1, \ldots, y_N)^T$, $\mathbf{\Phi} = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_N)]$ is a $p_H \times N$ matrix and $\alpha = (\alpha_1, \ldots, \alpha_N)^T$ is the vector of weights. The solution is obtained by the stationary points of $J(\alpha)$:

$$(\mathbf{K} + \gamma\mathbf{I})\alpha = \mathbf{y}, \tag{12}$$

$$\alpha = (\mathbf{K} + \gamma\mathbf{I})^{-1}\mathbf{y}, \tag{13}$$

where $\mathbf{I}$ is the identity matrix of suitable dimension. The prevalent computational burden stems from the fact that every training sample will contribute one parameter to the resulting model (all of them become support vectors).

Recursive kernel methods have been proposed to more efficiently learn in data streams. Based on the traditional Recursive Least Squares (RLS), the Kernel Recursive Least Squares (KRLS) [36] was developed as one of the first recursive versions of the LSSVR. Aiming at achieving a better compromise between computational complexity and convergence to the optimal solution of Eq. (11), its authors derived a recursive formulation, with two important approximations: the suppression of the regularization term in Eq. (11) from the learning iterations and the adoption of a dictionary of quasi-independent data examples that could spam most of the feature space, reducing the number of support vectors to the size of the dictionary at step t, m_t.

The RLS method traditionally solves Eq. (9) in a recursive manner and was the inspiration for the proposal of the KRLS, a recursive strategy to solve the LSSVR. As explained before, the removal of the regularization term and the substitution of all examples by a dictionary (better detailed in the next sections) of linearly independent support vectors were employed to allow for efficient recursive calculations. Other authors proposed different mechanisms, such as the adoption of sliding-windows [37] and growing-and-pruning [38] strategies.

Its reasonable cost and prominent results fomented the emergency of other methods, including the Online Sparse Least Squares Support Vector Regression (OSLSSVR) [6], whose main contribution was the return of the regularization term, adding it to the learning scheme of Eq. (11), and the derivation of its recursive formulas. One may assume that the importance of regularization is only due to its ability to properly regulate the flexibility of the predictor, aiming at improving its generalization performance [39], which could be enough to justify the relevance of the OSLSSVR proposal. However, the literature of ML presents a variety of fields in which regularization plays a primary role, for instance, the structural regularization area of the Multi-Task Learning (MTL) field [9], which employs specific regularization terms to promote information sharing between learning tasks, which tends to improve, depending on the relationship among the tasks, their individual performance. Then, the novelty of the OSLSSVR may also pave the way to the development of online proposals of well-established regularization-dependent methods.

The authors of the OSLSSVR kept the same dictionary mechanism of the original KRLS, adding the regularization term into the recursive formulation. They assessed its performance on regression benchmarks such as function approximation, system identification and time-series prediction, demonstrating its superiority in comparison to other kernel-based online methods. It is important to note that, as we shall see in the following section, the mechanism

of growing the dictionary may serve as reaction to novelty in the distribution of the input $\mathbf{x}_t$, a special case of Concept Drift (CD) [17]. A sudden change in the function $f(\mathbf{x}_t)$ is dealt, however, by the standard learning procedures, which may not respond as fast as the SOTA algorithms for handling CD.

3.4.1 Regularization-Dependent Regression Models

The importance of regularization in ML is described by several works in literature. For instance, it allows the regulation of the bias-variance trade-off [39], helps to define the contours of the optimization space [40], can be seen as hidden priors by Bayesian approaches [41], can promote sparsity [42] and can be employed in feature selection. As highlighted before, we exemplify its major importance with models that depend on the regularization to even exist, such as for the MTL structural regularization field [10].

We conjecture that, with appropriate kernel functions and a few adaptations, one could turn the OSLSSVR method into an online MTL formulation, reinforcing the importance of having a correct version of it, following the same strategy adopted in [43].

4 Reformulation of the OSLSSVR model

In this section, we present the OSLSSVR original formulation and its misconceptions in some of its recursive formulas (subsection 4.1), we then propose a corrected formulation (subsection 4.2) and describe our experimental setup in order to demonstrate the harmfulness of the original work and the benefits of our reformulated proposal (subsection 4.3), which are presented and discussed in subsection 4.4.

4.1 Original OSLSSVR formulation and misconceptions

The fundamental idea behind the OSLSSVR method is to recursively minimize the cost function in (11), thus avoiding the storage of all data instances and also preventing the necessity of solving the linear system (13) for every new incoming sample. The authors were inspired by the original KRLS, and have promoted some modifications to incorporate the regularization term. We present their formulation in the next paragraphs, point out the flaws in the formulation, and come up with the corrected version of the OSLSSVR method.

The data storage problem is addressed both in KRLS and OSLSSVR by the adoption of a dictionary containing only a subset of the input instances $\mathcal{D}_{t-1}^{sv} = \{\tilde{\mathbf{x}}_j\}_{j=1}^{m_{t-1}}$, which are supposed to be capable of spanning the entire feature space. This is done with the assistance of the Approximate Linear Dependency (ALD) criterion [36], that regulates the addition of a new training input to the dictionary in the following way

$$\delta_t \overset{\text{def}}{=} \min_{\mathbf{a}} \left\| \sum_{m=1}^{m_{t-1}} a_m \phi\left(\tilde{\mathbf{x}}_m\right) - \phi\left(\mathbf{x}_t\right) \right\|^2 \leq v \tag{14}$$

where v is the sparsity level parameter and $\mathbf{a} = \left(a_1, \ldots, a_{m_{t-1}}\right)^T$ is the vector containing the coefficients of the linear combination. We state that the input instance at time t can be approximated by a linear combination of the current dictionary within a squared error v if $\delta_t \leq v$. The association of the condition in (14) and the kernel definition results in:

$$\delta_t = \min_{\mathbf{a}} \left\{ \mathbf{a}^T \tilde{\mathbf{K}}_{t-1} \mathbf{a} - 2\mathbf{a}^T \tilde{\mathbf{k}}_{t-1}\left(\mathbf{x}_t\right) + k_{tt} \right\} \tag{15}$$

where $\tilde{\mathbf{K}}_{t-1} \in \mathbb{R}^{m_{t-1}} \times \mathbb{R}^{m_{t-1}}$ is the kernel matrix computed with the inputs of the dictionary, $\tilde{\mathbf{k}}_{t-1}\left(\mathbf{x}_t\right) \in \mathbb{R}^{m_{t-1}}$ and $k_{tt} = k\left(\mathbf{x}_t, \mathbf{x}_t\right) \in \mathbb{R}$. The (i,j)th entry of $\tilde{\mathbf{K}}_{t-1}$ is defined by $\left[\tilde{\mathbf{K}}_{t-1}\right]_{i,j} = k\left(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j\right)$, for $i, j = 1, \ldots, m_{t-1}$, and the ith component of $\tilde{\mathbf{k}}_{t-1}\left(\mathbf{x}_t\right)$ is defined by $\left(\tilde{\mathbf{k}}_{t-1}\left(\mathbf{x}_t\right)\right)_i = k\left(\tilde{\mathbf{x}}_i, \mathbf{x}_t\right)$. The solution of the formulation in (15) is given by

$$\mathbf{a}_t = \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}\left(\mathbf{x}_t\right) \tag{16}$$

for which we have

$$\delta_t = k_{tt} - \tilde{\mathbf{k}}_{t-1}\left(\mathbf{x}_t\right)^T \mathbf{a}_t \leq v \tag{17}$$

If otherwise $\delta_t > v$, the current dictionary must be expanded by adding $\mathbf{x}_t$. Thereby, $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv} \cup \{\mathbf{x}_t\}$ and $m_t = m_{t-1} + 1$.

In the following paragraphs we describe the closed-form solution to the learning problem - see Eqs. (11) to (13) - adapted to the dictionary approximation presented before. After, we present the recursive solution to this problem when our dictionary is not changed (Case 1), and when the dictionary is updated (Case 2). The identified misconception of the original formulation is presented in Case 2.

Thus, at time step t, we get

$$J\left(\alpha_t\right) = \left\|\mathbf{K}_t\alpha_t - \mathbf{y}_t\right\|^2 + \gamma\alpha_t^T\mathbf{K}_t\alpha_t, \tag{18}$$

which can also be written as

$$J\left(\alpha_t\right) = \left\|\mathbf{\Phi}_t^T\mathbf{\Phi}_t\alpha_t - \mathbf{y}_t\right\|^2 + \gamma\alpha_t^T\mathbf{\Phi}_t^T\mathbf{\Phi}_t\alpha_t \tag{19}$$

Then, we consider the following approximation:

$$\mathbf{w}_t = \mathbf{\Phi}_t\alpha_t \approx \tilde{\mathbf{\Phi}}_t\mathbf{A}_t^T\alpha_t = \tilde{\mathbf{\Phi}}_t\tilde{\alpha}_t, \tag{20}$$

where $\mathbf{A}_t = \left[\begin{array}{cccc}\mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_t\end{array}\right]^T \in \mathbb{R}^{t\times m_t}$. Then, the cost function in (18) becomes

$$J\left(\tilde{\alpha}_t\right) = \left\|\mathbf{\Phi}_t^T\tilde{\mathbf{\Phi}}_t\tilde{\alpha}_t - \mathbf{y}_t\right\|^2 + \gamma\alpha_t^T\mathbf{\Phi}_t^T\tilde{\mathbf{\Phi}}_t\tilde{\alpha}_t, \tag{21}$$

$$= \left\|\mathbf{A}_t\tilde{\mathbf{K}}_t\tilde{\alpha}_t - \mathbf{y}_t\right\|^2 + \gamma\tilde{\alpha}_t^T\tilde{\mathbf{K}}_t\tilde{\alpha}_t, \tag{22}$$

where $\tilde{\alpha}_t \in \mathbb{R}^{m_t}$ is a reduced vector of m_t coefficients. Then, the minimization of the cost function in (21) yields the following solution:

$$\tilde{\alpha}_t = \left(\tilde{\mathbf{K}}_t\mathbf{A}_t^T\mathbf{A}_t\tilde{\mathbf{K}}_t + \gamma\tilde{\mathbf{K}}_t\right)^{-1}\tilde{\mathbf{K}}_t\mathbf{A}_t^T\mathbf{y}_t, \tag{23}$$

$$= \left[\tilde{\mathbf{K}}_t\left(\mathbf{A}_t^T\mathbf{A}_t + \gamma\tilde{\mathbf{K}}_t^{-1}\right)\right]^{-1}\mathbf{A}_t^T\mathbf{y}_t, \tag{24}$$

$$= \tilde{\mathbf{K}}_t^{-1}\left(\mathbf{A}_t^T\mathbf{A}_t + \gamma\tilde{\mathbf{K}}_t^{-1}\right)^{-1}\mathbf{A}_t^T\mathbf{y}_t \tag{25}$$

By defining a matrix $\mathbf{P}_t$ as

$$\mathbf{P}_t = \left(\mathbf{A}_t^T\mathbf{A}_t + \gamma\tilde{\mathbf{K}}_t^{-1}\right)^{-1} \tag{26}$$

we finally get

$$\tilde{\alpha}_t = \tilde{\mathbf{K}}_t^{-1}\mathbf{P}_t\mathbf{A}_t^T\mathbf{y}_t \tag{27}$$

The next step requires the computation of the inverse matrices in (26) and (27) iteratively using the RLS algorithm. For this purpose, we have two possible situations following the same recursive procedure of the KRLS algorithm, which are described in what follows.

4.1.1 Case 1 - Unchanged dictionary ($m_t = m_{t-1}$)

In this case, $\delta_t \leq v$, meaning that $\phi(\mathbf{x}_t)$ is approximately linearly dependent of the dictionary vectors. Hence, $\mathbf{x}_t$ is not added to the dictionary and the kernel matrix is not changed. Mathematically, $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv}$ and $\tilde{\mathbf{K}}_t = \tilde{\mathbf{K}}_{t-1}$.

Since $\mathbf{a}_t$ needs to be computed using Eq. (17) to determine δ_t, matrix $\mathbf{A}_t$ is built iteratively by the inclusion of $\mathbf{a}_t$, i.e. $\mathbf{A}_t = \begin{bmatrix} \mathbf{A}_{t-1}^T & \mathbf{a}_t \end{bmatrix}^T$. Thus, let us define matrix $\mathbf{B}_t$ as

$$\begin{aligned} \mathbf{B}_t &= \mathbf{A}_t^T \mathbf{A}_t + \gamma \tilde{\mathbf{K}}_t^{-1} \\ &= \mathbf{A}_{t-1}^T \mathbf{A}_{t-1} + \gamma \tilde{\mathbf{K}}_{t-1}^{-1} + \mathbf{a}_t \mathbf{a}_t^T \\ &= \mathbf{B}_{t-1} + \mathbf{a}_t \mathbf{a}_t^T, \end{aligned} \tag{28}$$

where $\mathbf{A}_t^T \mathbf{A}_t = \mathbf{A}_{t-1}^T \mathbf{A}_{t-1} + \mathbf{a}_t \mathbf{a}_t^T$. Therefore, we can use the standard RLS algorithm based on the matrix inversion lemma [44] to recursively compute the matrix $\mathbf{P}_t$ as

$$\mathbf{P}_t = \mathbf{B}_t^{-1} = \mathbf{P}_{t-1} - \frac{\mathbf{P}_{t-1} \mathbf{a}_t \mathbf{a}_t^T \mathbf{P}_{t-1}}{1 + \mathbf{a}_t^T \mathbf{P}_{t-1} \mathbf{a}_t} \tag{29}$$

We define a gain vector $\mathbf{q}_t$ as

$$\mathbf{q}_t = \frac{\mathbf{P}_{t-1} \mathbf{a}_t}{1 + \mathbf{a}_t^T \mathbf{P}_{t-1} \mathbf{a}_t} \tag{30}$$

and consequently

$$\mathbf{P}_t = \mathbf{P}_{t-1} - \mathbf{q}_t \mathbf{a}_t^T \mathbf{P}_{t-1} \tag{31}$$

Finally, using the fact that $\mathbf{A}_t^T \mathbf{y}_t = \mathbf{A}_{t-1}^T \mathbf{y}_{t-1} + \mathbf{a}_t y_t$, the OSLSSVR update rule for $\tilde{\alpha}_t$ can be written as:

$$\begin{aligned} \tilde{\alpha}_t &= \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_t \mathbf{A}_t^T \mathbf{y}_t, \\ &= \tilde{\mathbf{K}}_t^{-1} \left(\mathbf{P}_{t-1} - \mathbf{q}_t \mathbf{a}_t^T \mathbf{P}_{t-1} \right) \left(\mathbf{A}_{t-1}^T \mathbf{y}_{t-1} + \mathbf{a}_t y_t \right), \\ &= \tilde{\alpha}_{t-1} + \tilde{\mathbf{K}}_t^{-1} \left(\mathbf{P}_t \mathbf{a}_t y_t - \mathbf{q}_t \mathbf{a}_t^T \tilde{\mathbf{K}}_t \tilde{\alpha}_{t-1} \right), \\ &= \tilde{\alpha}_{t-1} + \tilde{\mathbf{K}}_t^{-1} \left(\mathbf{q}_t y_t - \mathbf{q}_t \mathbf{a}_t^T \tilde{\mathbf{K}}_t \tilde{\alpha}_{t-1} \right), \\ &= \tilde{\alpha}_{t-1} + \tilde{\mathbf{K}}_t^{-1} \mathbf{q}_t \left(y_t - \tilde{\mathbf{k}}_{t-1} (\mathbf{x}_t)^T \tilde{\alpha}_{t-1} \right), \end{aligned} \tag{32}$$

where some handlings are based on $\mathbf{q}_t = \mathbf{P}_t \mathbf{a}_t$, and $\tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) = \tilde{\mathbf{K}}_t \mathbf{a}_t$.

4.1.2 Case 2 - Updating the dictionary ($m_t = m_{t-1} + 1$)

When $\delta_t > v$, we add $\mathbf{x}_t$ to the dictionary ($\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv} \cup \{\mathbf{x}_t\}$) and update $m_t = m_{t-1} + 1$. We, then, have to compute $\tilde{\mathbf{K}}_t$ and $\tilde{\mathbf{K}}_t^{-1}$ recursively using $\tilde{\mathbf{K}}_{t-1}$, $\tilde{\mathbf{K}}_{t-1}^{-1}$ and the information provided by the new sample:

$$\tilde{\mathbf{K}}_t = \begin{bmatrix} \tilde{\mathbf{K}}_{t-1} & \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \\ \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^T & k_{tt} \end{bmatrix} \tag{33}$$

and

$$\tilde{\mathbf{K}}_t^{-1} = \frac{1}{\delta_t}\begin{bmatrix} \delta_t\tilde{\mathbf{K}}_{t-1}^{-1} + \mathbf{a}_t\mathbf{a}_t^T & -\mathbf{a}_t \\ -\mathbf{a}_t^T & 1 \end{bmatrix}. \tag{34}$$

Unlike Case 1, we note that the matrices have their dimensions expanded in one unit due to augmentation of the dictionary.

In order to determine $\mathbf{P}_t$, we also need to increase the dimensions of $\mathbf{A}_t^T\mathbf{A}_t$. This is performed according to the following equation, which simply means that the new instance can only be computed within $\mathcal{D}_t^{sv}$ by the trivial linear combination with itself,

$$\mathbf{A}_t^T\mathbf{A}_t = \begin{bmatrix} \mathbf{A}_{t-1}^T\mathbf{A}_{t-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{35}$$

where $\mathbf{0}$ is a zero vector of appropriate size. This is the point where some misconceptions can be found in the original formulation [6]. First, the expression for $\mathbf{P}_t$ is presented as

$$\mathbf{P}_t = \begin{bmatrix} \mathbf{A}_{t-1}^T\mathbf{A}_{t-1} + \gamma\tilde{\mathbf{K}}_{t-1}^{-1} + \frac{\gamma}{\delta_t}\mathbf{a}_t\mathbf{a}_t^T & -\frac{\gamma}{\delta_t}\mathbf{a}_t \\ -\frac{\gamma}{\delta_t}\mathbf{a}_t^T & \frac{\gamma}{\delta_t} \end{bmatrix}^{-1} \tag{36}$$

lacking a $+1$ coming from (35), which is easily corrected by

$$\mathbf{P}_t = \begin{bmatrix} \mathbf{A}_{t-1}^T\mathbf{A}_{t-1} + \gamma\tilde{\mathbf{K}}_{t-1}^{-1} + \frac{\gamma}{\delta_t}\mathbf{a}_t\mathbf{a}_t^T & -\frac{\gamma}{\delta_t}\mathbf{a}_t \\ -\frac{\gamma}{\delta_t}\mathbf{a}_t^T & 1+\frac{\gamma}{\delta_t} \end{bmatrix}^{-1}. \tag{37}$$

Then, we are presented to a problematic formulation for $\mathbf{P}_t$:

$$\mathbf{P}_t = \begin{bmatrix} \mathbf{P}_{t-1} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + \frac{1}{\Delta_b}\begin{bmatrix} -\mathbf{P}_{t-1}\mathbf{b} \\ 1 \end{bmatrix}\cdot\begin{bmatrix} -\mathbf{P}_{t-1}\mathbf{b} \\ 1 \end{bmatrix}^T, \tag{38}$$

where $\Delta_b = b^* - \mathbf{b}^T\mathbf{B}_{t-1}^{-1}\mathbf{b}, b^* = 1 + \frac{\gamma}{\delta_t}$ and $\mathbf{b} = -\frac{\gamma}{\delta_t}\mathbf{a}_t$. Defining a constant $c = \gamma/\delta_t$, we get

$$\Delta_b = 1 + c - c^2\mathbf{a}_t^T\mathbf{P}_{t-1}\mathbf{a}_t, \tag{39}$$

and the matrix $\mathbf{P}_t$ can be written as

$$\mathbf{P}_t = \frac{1}{\Delta_b}\begin{bmatrix} \Delta_b\mathbf{P}_{t-1} + c^2\mathbf{P}_{t-1}\mathbf{a}_t\mathbf{a}_t^T\mathbf{P}_{t-1} & c\mathbf{P}_{t-1}\mathbf{a}_t \\ c\mathbf{a}_t^T\mathbf{P}_{t-1} & 1 \end{bmatrix} \tag{40}$$

Let us demonstrate that Eq. (40) does not hold true. In fact, by multiplying it by its inverse defined in Eq. (37), the identity is not obtained as the result. We employ the following property of block matrices to assist us in this task. Let $\mathbf{M}_1$ and $\mathbf{M}_2$ be two block matrices of compatible dimensions

$$\mathbf{M}_1 = \begin{bmatrix} \mathbf{M}_{1,1} & \mathbf{M}_{1,2} \\ \mathbf{M}_{1,3} & \mathbf{M}_{1,4} \end{bmatrix} \tag{41}$$

$$\mathbf{M}_2 = \begin{bmatrix} \mathbf{M}_{2,1} & \mathbf{M}_{2,2} \\ \mathbf{M}_{2,3} & \mathbf{M}_{2,4} \end{bmatrix}, \tag{42}$$

where $\mathbf{M}_{i,j}$ are matrices of appropriated dimensions. The product between $\mathbf{M}_1$ and $\mathbf{M}_2$ is given by

$$\mathbf{M}_1\mathbf{M}_2 = \begin{bmatrix} \mathbf{M}_{1,1}\mathbf{M}_{2,1} + \mathbf{M}_{1,2}\mathbf{M}_{2,3} & \mathbf{M}_{1,1}\mathbf{M}_{2,2} + \mathbf{M}_{1,2}\mathbf{M}_{2,4} \\ \mathbf{M}_{1,3}\mathbf{M}_{2,1} + \mathbf{M}_{1,4}\mathbf{M}_{2,3} & \mathbf{M}_{1,3}\mathbf{M}_{2,2} + \mathbf{M}_{1,4}\mathbf{M}_{2,4} \end{bmatrix} \tag{43}$$

if all the products are well-defined.

Then, we demonstrate that the block part $\mathbf{M}_{1,3}\mathbf{M}_{2,1} + \mathbf{M}_{1,4}\mathbf{M}_{2,3}$ of the product of $\mathbf{P}_t$ ($\mathbf{M}_1$) and $\mathbf{P}_t^{-1}$ ($\mathbf{M}_2$) does not result in a null matrix (the non diagonal elements of this product should all be zero), producing:

$$\begin{aligned} \mathbf{M}_{1,3}\mathbf{M}_{2,1} + \mathbf{M}_{1,4}\mathbf{M}_{2,3} = \\ \frac{c}{\Delta_b}\mathbf{a}_t^T\mathbf{P}_{t-1}(\mathbf{A}_{t-1}^T\mathbf{A}_{t-1} + \gamma\tilde{\mathbf{K}}_{t-1}^{-1} + c\mathbf{a}_t\mathbf{a}_t^T) - \frac{c}{\Delta_b}\mathbf{a}_t = \\ \frac{c}{\Delta_b}\mathbf{a}_t^T\mathbf{P}_{t-1}(\mathbf{P}_{t-1}^{-1} + c\mathbf{a}_t\mathbf{a}_t^T) - \frac{c}{\Delta_b}\mathbf{a}_t^T = \\ \frac{c^2}{\Delta_b}\mathbf{a}_t^T\mathbf{P}_{t-1}\mathbf{a}_t\mathbf{a}_t^T \end{aligned} \tag{44}$$

which proves that the expression for $\mathbf{P}_t$ is incorrect and, therefore, the original derivation of $\tilde{\alpha}_t$ is inaccurate as well.

4.2 Reformulation of $\mathbf{P}_t$ in Case 2

Since the misconceptions are only present in the formulation of $\mathbf{P}_t$ in Case 2, we resort to all the original derivations until and including Eq. (35). This section is devoted, thus, to the development of a correct recursive formulation for $\mathbf{P}_t$, defined by Eq. (37).

The computation of $\mathbf{P}_t$ can, however, be correctly derived if we introduce auxiliary matrices S_t and S_{t-1}, an auxiliary vector s_t, and the inversion lemma [45], guiding to:

$$\mathbf{S}_{t-1} = \begin{bmatrix} \mathbf{A}_{t-1}^T\mathbf{A}_{t-1} + \gamma\tilde{\mathbf{K}}_{t-1}^{-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{t-1}^{-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{45}$$

$$\mathbf{s}_t = \sqrt{\frac{\gamma}{\delta_t}} \begin{bmatrix} \mathbf{a}_t^T & 1 \end{bmatrix}^T, \tag{46}$$

where $\mathbf{P}_t = \mathbf{S}_t^{-1} = (\mathbf{S}_{t-1} + \mathbf{s}_t^T\mathbf{s}_t)^{-1}$. Thus,

$$\mathbf{P}_t = \mathbf{S}_t^{-1} = \mathbf{S}_{t-1}^{-1} - \frac{\mathbf{S}_{t-1}^{-1}\mathbf{s}_t\mathbf{s}_t^T\mathbf{S}_{t-1}^{-1}}{1 + \mathbf{s}_t^T\mathbf{S}_{t-1}^{-1}\mathbf{s}_t} \tag{47}$$

where $\mathbf{S}_{t-1}^{-1}$ is easily derived as follows due to its shape

$$\mathbf{S}_{t-1}^{-1} = \begin{bmatrix} \mathbf{P}_{t-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{48}$$

The $\mathbf{P}_t$ matrix can be lastly simplified to arrive at the following formulation:

$$\mathbf{P}_t = \frac{1}{\Delta_b} \begin{bmatrix} \Delta_b\mathbf{P}_{t-1} - c\mathbf{P}_{t-1}\mathbf{a}_t\mathbf{a}_t^T\mathbf{P}_{t-1} & c\mathbf{P}_{t-1}\mathbf{a}_t \\ c\mathbf{a}_t^T\mathbf{P}_{t-1} & \Delta_b - c \end{bmatrix} \tag{49}$$

with $\Delta_b = 1 + c + c\mathbf{a}_t^T\mathbf{P}_{t-1}\mathbf{a}_t$ and $c = \frac{\gamma}{\delta_t}$. Once we have derived $\mathbf{P}_t$, $\mathbf{A}_t$ and $\tilde{\mathbf{K}}_t^{-1}$, the computation of $\tilde{\alpha}_t$ is provided by Eq. (27).

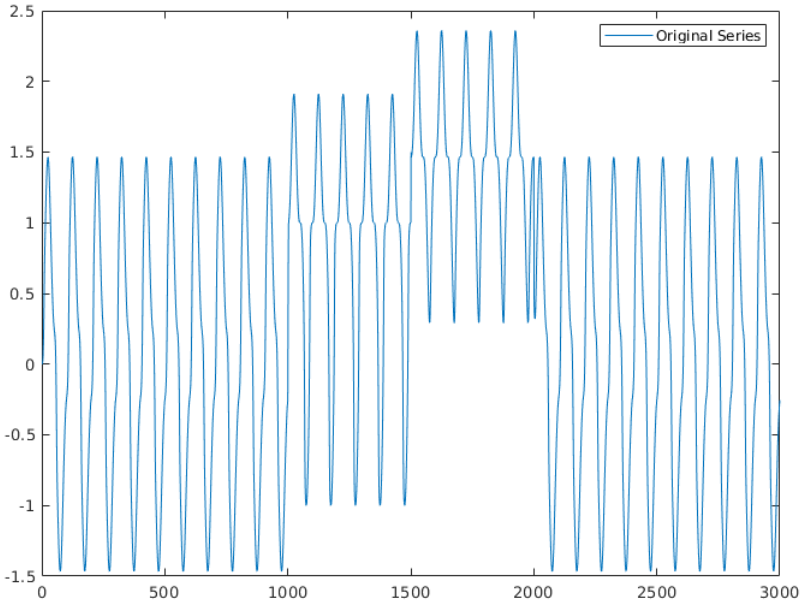


Figure 1: Original Series of the Nonlinear Dynamic Plant Identification with Time-Varying Experiment

4.3 Experimental Setup

The three main purposes here are: (1) to reveal the harmful effects of the misconceptions in the original OSLSSVR formulation; (2) to evidence the correctness of the OSLSSVR reformulation; and (3) to illustrate the potential of the new version of the OSLSSVR in practice. The original work in [6] already included a broad comparison of OSLSSVR with contenders in the literature. Even with the inconsistencies reported and corrected here, the original version of OSLSSVR was demonstrated to be the best candidate in other scenarios. Therefore, a single online regression case study is considered, for which we applied both versions of the OSLSSVR and compared the outcome, guided by two validation criteria, as described in the following paragraphs.

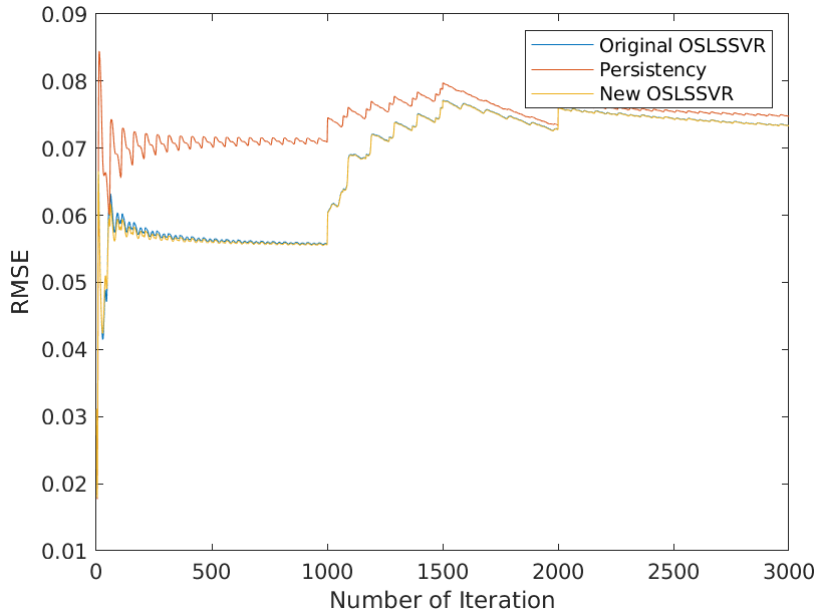
The Nonlinear Dynamic Plant Identification With Time-Varying Characteristics was first proposed by Nguyen et al. [46]. This experiment is based on the following model:

$$y(t+1) = \frac{y(t)}{1+y^2(t)} + u^3(t) + n(t), \tag{50}$$

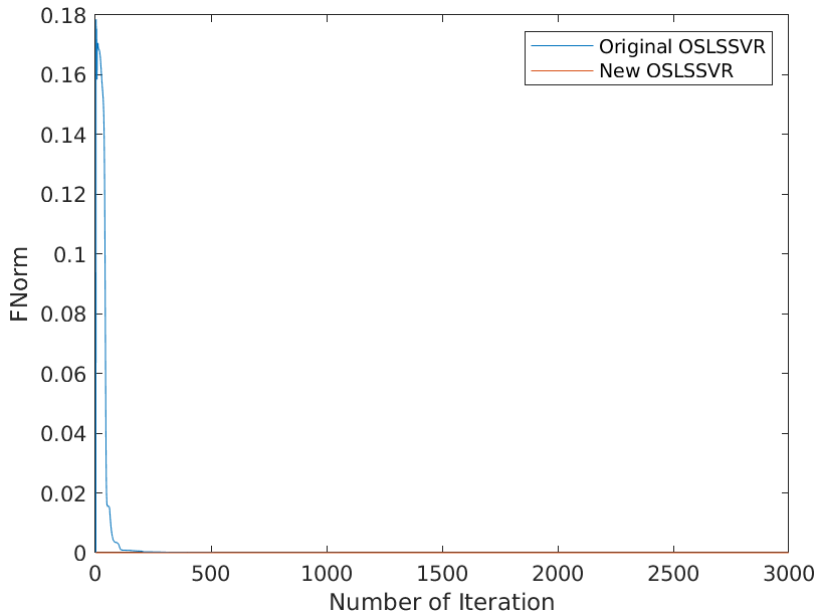
where $u(t) = \sin(2\pi t/100)$ and $n(t)$ is a time-varying factor, provided by (99):

$$n(t) = \begin{cases} 0, & 1 \leq t \leq 1000 \text{ and } t \geq 2001 \\ 0.5, & 1001 \leq t \leq 1500 \\ 1, & 1501 \leq t \leq 2000 \end{cases}. \tag{51}$$

As done in [47], we used the initial 3000 for training and testing both proposals, varying the hyperparameters to validate our reformulation under different conditions. We, then, fix $\lambda = 1$ and employ $(v, \gamma) \in \{(0.01, 0.01), (0.01, 10), (1, 1000)\}$.

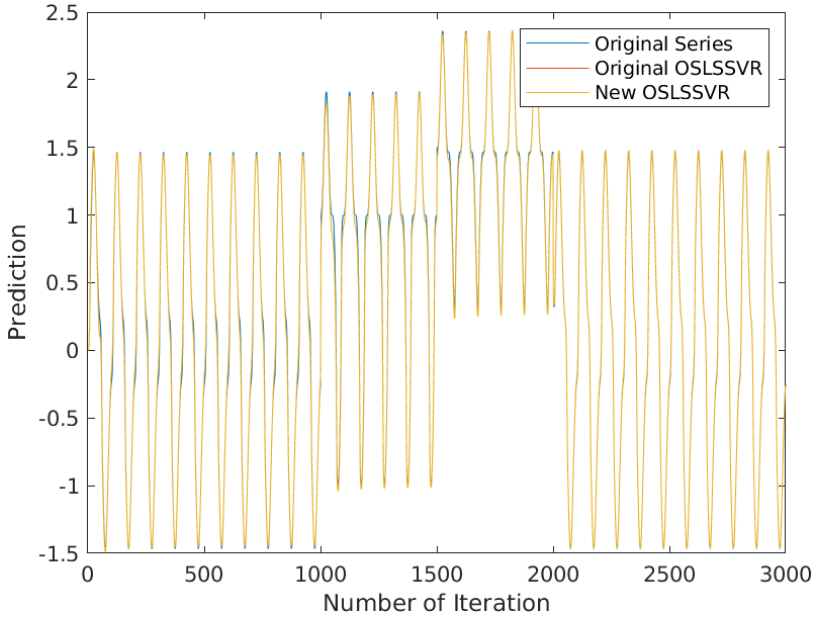


(a)



(b)

Figure 2: Cumulative RMSE (a), FNorm (b) and Predictions (c) over time for $(v, \gamma) = (0.01, 0.01)$



(c)

Figure 2: Cumulative RMSE (a), FNorm (b) and Predictions (c) over time for $(v, \gamma) = (0.01, 0.01)$

We report, for each set of hyperparameters, the Root Mean Squared Error (RMSE) - see Eq. (97) - and the Frobenius Norm (FNorm) [48] of the difference of the true value of $\mathbf{P}_t$, computed with the direct inversion of Eq. (37), and the recursively derived value of $\mathbf{P}_t$ for the original - see Eq. (40) - and the new - see Eq. (49) - formulations.

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(y^{[t]} - \hat{y}^{[t]})^2} \tag{52}$$

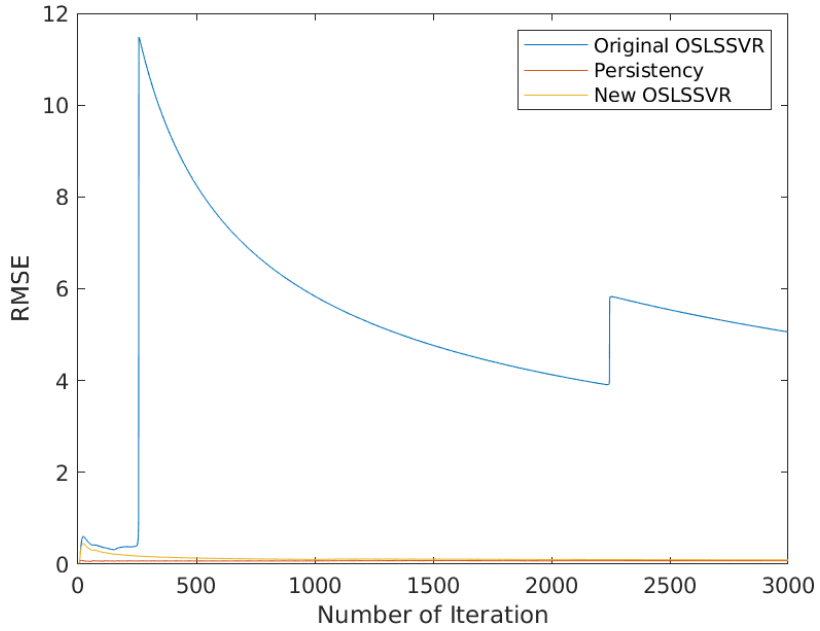
$$FNorm = \|\mathbf{P}_t - \mathbf{P}_t^{real}\|_F, \tag{53}$$

The RMSE is intended to demonstrate the gain in performance when adopting the correct derivations, while the FNorm illustrates how far the original and the new recursive computations of P_t are from its true value.

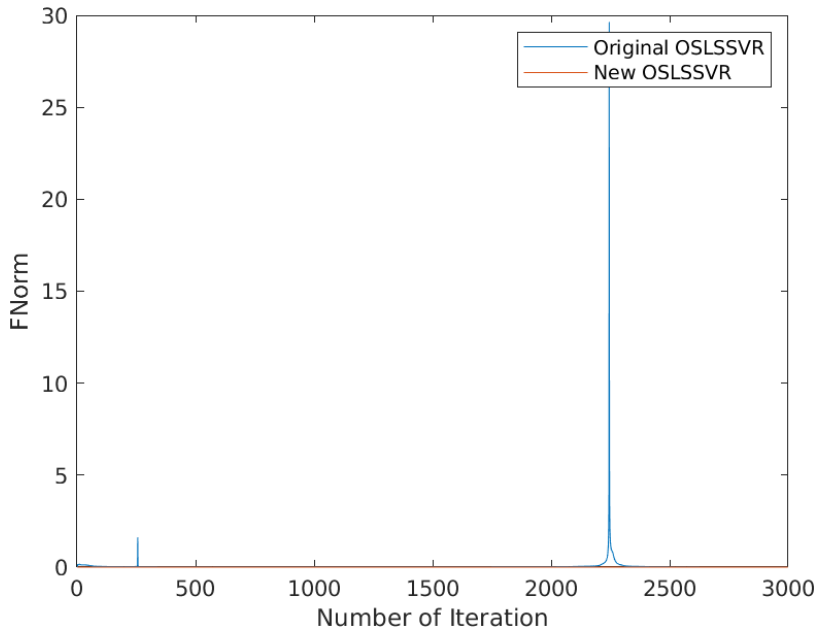
4.4 Results and Discussion

We present, in Figures 2(a), 2(b) and 2(c), the RMSE, the FNorm and the predictions for $(v, \gamma) = (0.01, 0.01)$. In this scenario, the similarity of the RMSE and the predictions may hide the effect of the errors found in the equations, although the high values of the FNorm for the original OSLSSVR already reveal that something is amiss.

It is when we increase γ, rising the penalty over the regularization term, that we are faced with some unexpected results. Figures 3(a), 3(b) and 3(c) show how the RMSE of the original OSLSSVR overtakes those of its new version and the persistence method for $(v, \gamma) = (0.01, 10)$, which is caused by the notable instability of the predictions, also evidenced by the two peaks in the FNorm around the steps 250 and 2250.

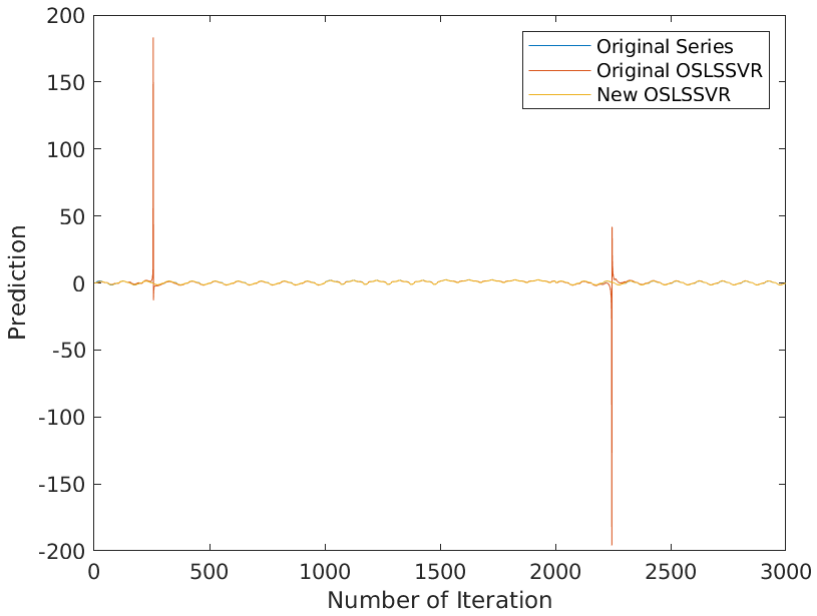


(a)



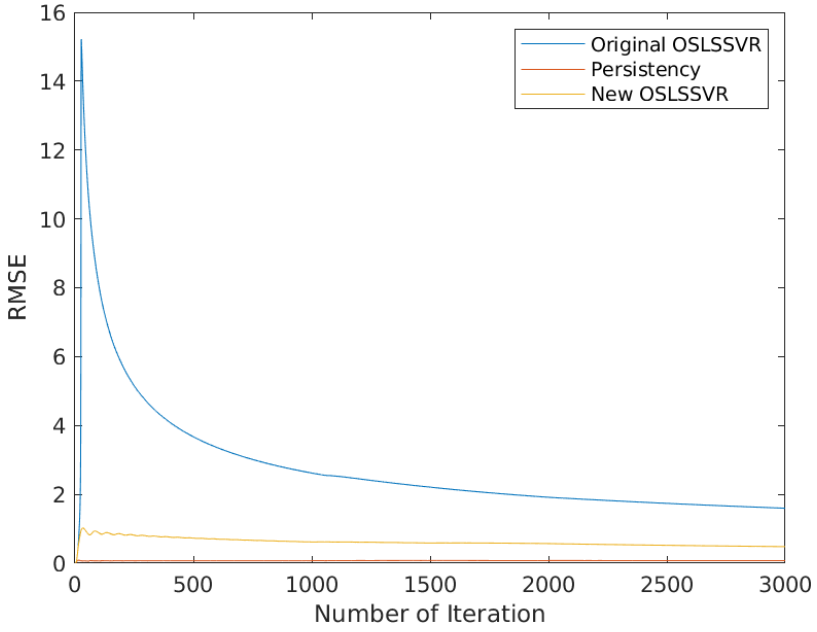
(b)

Figure 3: Cumulative RMSE (a), FNorm (b) and Predictions (c) over time for $(v, \gamma) = (0.01, 10)$



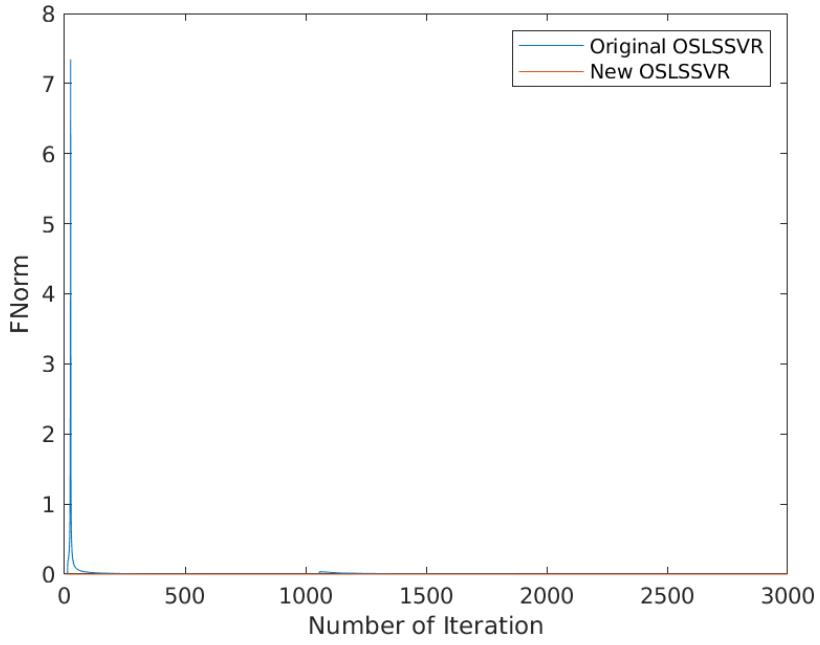
(c)

Figure 3: Cumulative RMSE (a), FNorm (b) and Predictions (c) over time for $(v, \gamma) = (0.01, 10)$



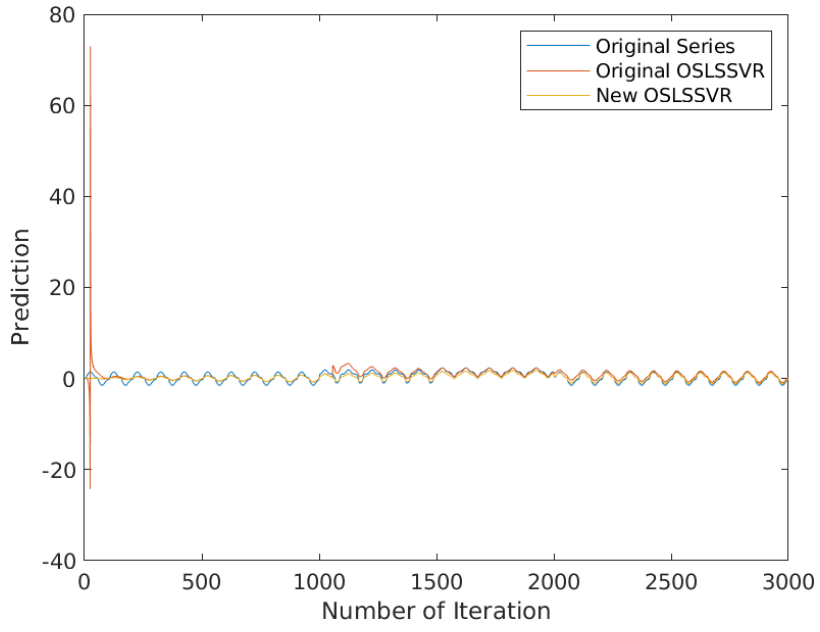
(a)

Figure 4: Cumulative RMSE (a), FNorm (b) and Predictions (c) over time for $(v, \gamma) = (1, 1000)$



(b)

Figure 4: Cumulative RMSE (a), FNorm (b) and Predictions (c) over time for $(v, \gamma) = (1, 1000)$



(c)

Figure 4: Cumulative RMSE (a), FNorm (b) and Predictions (c) over time for $(v, \gamma) = (1, 1000)$

For $(v, \gamma) = (1, 1000)$, Figures 4(a), 4(b) and 4(c) present similar behavior with instability of the predictions for the original method in the first steps, when the FNorm reaches high values. Since v is considerably high, new insertions to the dictionary are less likely to occur after some time, promoting mostly the Case 1, for which the original formulation is correct. Thus the learning process is reestablished and the cumulative RMSE drops to, though still detrimental, lower levels.

5 Proposed Online MTL Methods

As introduced in the last sections, the two main objectives of this research are to develop an online multi-task learning method with a better compromise between per-instance cost and convergence to the exact solution (subsections 5.1, 5.2 and 5.3). In subsections 5.4 and 5.5, we provide the details, results and discussion of the experiments performed on a real-world time-dependent multi-task benchmark, aiming at assessing the quality of our proposals and comparing them to other contenders in literature.

5.1 Graph-Based MTL Reformulation

Due to its quadratic optimization structure, convexity, and successful application to synthetic and real-world problems and to evolving systems, we selected the Graph-Based MTL formulation to have a recursive formulation pursued and derived in this work.

The quadratic shape of the terms composing the function to be minimized, allied to the fact that we are dealing with linear predictors, allows us to convert it to a single task problem with a loss term identical to the least squares linear regression and a quadratic regularization on the parameters vector, making its closed-form solution analogous to the one that is dealt in the Weighted Recursive Least Squares (WRLS), the algorithm that we exploit in our first proposal.

The existence of a multi-task kernel formalism for the graph-based regularization in the literature [21] also motivates and supports its adoption in this work, as we turn to recursive kernel-based methods to propose a second online MTL formulation.

We briefly rewrite the adopted MTL formulation in order to simplify its representation and the understanding of the development of the recursive proposals, specially for the one based on WRLS.

Let $\mathbf{X}_t \in \mathbb{R}^{N \times d}$ and $\mathbf{y}_t \in \mathbb{R}^N, t \in 1, ..., T$ be the input matrices and output vectors of the T tasks. Let $\mathbf{\Theta} \in \mathbb{R}^{d \times T}$ be the matrix containing in each column the parameters vector of each task t, $\mathbf{w}_t \in \mathbb{R}^d$, in such a way that the vector of predictions of task t is given by $\hat{\mathbf{y}}_t = \mathbf{X}_t \mathbf{w}_t$.

Recalling the general learning process of a multi-task problem, it is usually defined as the minimization of an objective function composed of a loss term, which represents the supervised learning component, and a regularization term, promoting information sharing among tasks [9].

$$\mathbf{\Theta}_* = \arg\min_{\Theta} J(\mathbf{\Theta}) \tag{54}$$

$$J(\mathbf{\Theta}) = \mathcal{L}(\mathbf{\Theta}, \mathbf{X_1}, ..., \mathbf{X}_T, \mathbf{y_1}, ..., \mathbf{y}_T) + \mathcal{R}(\mathbf{\Theta}) \tag{55}$$

We define, then, the neighbourhood of each task $\mathcal{E}_t, t \in 1, ..., T$ as the set of tasks that are connected to task t by two-way weighted edges. The weight of the edges is defined by similarities $sim(t, j)$ and $sim(j, t)$ between tasks t and j, as done in [8], representing the degree of relationship between tasks t and j. Therefore, we write the MTL regularization term as:

$$\mathcal{R}(\mathbf{\Theta}) = \sum_{t=1}^{T} \sum_{j \in \mathcal{E}_t} \|\mathbf{w}_t sim(t,j) - \mathbf{w}_j sim(j,t)\|^2 + \gamma \sum_{t=1}^{T} \|\mathbf{w}_t\|^2 \tag{56}$$

Adopting the quadratic error as loss function and the above regularization term, the Graph-Based Multi-Task Learning problem can be stated as follows:

$$\mathbf{\Theta}^* = \arg\min_{\Theta} \sum_{t=1}^{T} \| \mathbf{X}_t \mathbf{w}_t - \boldsymbol{y}_t\|^2 + \lambda \sum_{t=1}^{T} \sum_{j \in \mathcal{E}_t} \|\mathbf{w}_t sim(j,t) - \mathbf{w}_j sim(t,j)\|^2 + \gamma \sum_{t=1}^{T} \|\mathbf{w}_t\|^2 \tag{57}$$

5.2 Multi-task Weighted Recursive Least Squares

In this subsection, we present our first recursive MTL proposal. As mentioned before, the quadratic shape of the terms composing the MTL objective function motivates the development of a single task closed-form equivalent solution of (57). We shall see that this closed-form solution is identical to the one that is recursively solved by WRLS, whose iteration demands quadratic cost, presenting therefore, an intermediary complexity with an online exact solution of the problem.

We start with a brief introduction of the standard Weighted Recursive Least Squares. It will serve as the basic learning structure to be employed in our multi-task framework.

Let $\mathbf{X}(n) \in R^{n \times d}$ be the input matrix at time n, containing n past input vectors of dimension d, and let $\mathbf{y}(n) \in R^n$ be its respective output vector and $\mathbf{w}(n) \in R^d$ the parameters vector in time n. The analytical solution of the least squares problem is given by:

$$\mathbf{w}(n) = \mathbf{\Phi}(n)^{-1} \mathbf{\Psi}(n) \tag{58}$$

where $\mathbf{\Phi}(n) = [\mathbf{X}(n)'\mathbf{X}(n) + \mathbf{\Phi}(0)]$ e $\mathbf{\Psi}(\mathbf{n}) = \mathbf{X}(n)'\mathbf{y}(n)$.

We aim at developing a recursive procedure that solves $\mathbf{\Phi}(n)^{-1}$ and $\mathbf{\Psi}(n)$ from $\mathbf{\Phi}(n-1)^{-1}$ and $\mathbf{\Psi}(n-1)$. In order to achieve it, we perform the following decomposition with the introduction of the forgetting factor σ:

$$\mathbf{\Phi}(n) = \sigma \mathbf{\Phi}(n-1) + \mathbf{x}(n)\mathbf{x}(n)' \tag{59}$$

$$\mathbf{\Psi}(n) = \sigma \mathbf{\Psi}(n-1) + y(n)\mathbf{x}(n) \tag{60}$$

where $\mathbf{x}(n) \in R^d$ is the input vector at time n and $y(n)$ is the expected output value. Defining $\mathbf{P}(n) = \mathbf{\Phi}(n)^{-1}$ and employing the Woodbury's identity, we get the following recursive procedure:

$$\mathbf{k}(n) = \frac{\mathbf{P}(n-1)\mathbf{x}(n)}{\sigma + \mathbf{x}(n)'\mathbf{P}(n-1)\mathbf{x}(n)} \tag{61}$$

$$\alpha(n) = y(n) - \mathbf{x}(n)'\mathbf{w}(n-1) \tag{62}$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \alpha(n)\mathbf{k}(n) \tag{63}$$

$$\mathbf{P}(n) = \sigma^{-1}[\mathbf{P}(n-1) - \mathbf{k}(n)\mathbf{x}(n)'\mathbf{P}(n-1)] \tag{64}$$

The last step to complete the WRLS algorithm is choosing the initial conditions $\omega(0)$ and $\mathbf{P}(0)$, which are usually defined as:

- $\omega(0) = \mathbf{0}$
- $\mathbf{P}(0) = \lambda \mathbf{I}_{d\times d}$, $\lambda > 0$, to guarantee non-singularity of the initial $\mathbf{P}(n)$.

Recalling our graph-based MTL formulation (57), we derive, in the next paragraphs, a closed-form solution for this optimization problem in order to develop a multi-task WRLS. Since it is a convex unrestricted optimization problem [9], we shall only consider the third KKT condition, which guarantees that the stationary points of its Lagrangian are global minima.

We seek, therefore, the parameters for which the gradient of the objective function is zero.

$$\nabla_{\mathbf{w}_t} J(\mathbf{\Theta}) = -2\mathbf{X}_t'\mathbf{y}_t + 2\mathbf{X}_t'\mathbf{X}_t\mathbf{w}_t + 2\lambda \sum_{j\in\mathcal{E}_t} [\mathbf{w}_t sim(t,j) - \mathbf{w}_j sim(j,t)] + 2\lambda\gamma\mathbf{w}_t, t \in 1,...,T \quad (65)$$

$$\nabla_{\mathbf{w}_t} J(\mathbf{\Theta}) = \mathbf{0}, t \in 1,...,T \quad (66)$$

$$\implies \mathbf{X}_t'\mathbf{X}_t\mathbf{w}_t + \lambda \sum_{j\in\mathcal{E}_t} [\mathbf{w}_t sim(t,j) - \mathbf{w}_j sim(j,t)] + \lambda\gamma\mathbf{w}_t = \mathbf{X}_t'\mathbf{y}_t \quad (67)$$

$$\implies \{\mathbf{X}_t'\mathbf{X}_t + \lambda[\gamma + \sum_{j\in\mathcal{E}_t} sim(t,j)]\mathbf{I}\}\mathbf{w}_t - \lambda \sum_{j\in\mathcal{E}_t} \mathbf{w}_j sim(j,t) = \mathbf{X}_t'\mathbf{y}_t \quad (68)$$

We now perform a transformation to assist in developing our solution. Instead of working with a matrix $\mathbf{\Theta}$ and different input matrices for each task, we stack all the parameters vectors of each task into a single stacked vector, all the output vectors into a single output vector and all the input matrices into a block-diagonal input matrix, defined according to:

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ . \\ . \\ . \\ \mathbf{w}_T \end{bmatrix} \in R^{dT}, \mathbf{X} = \begin{bmatrix} \mathbf{X}_1 & \mathbf{0} & .... & \mathbf{0} \\ \mathbf{0} & \mathbf{X}_2 & .... & \mathbf{0} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ \mathbf{0} & \mathbf{0} & .... & \mathbf{X}_T \end{bmatrix} \in R^{nT\times dT}, \mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ . \\ . \\ . \\ \mathbf{y}_T \end{bmatrix} \in R^{nT} \quad (69)$$

Adopting this new representation scheme of our MTL basic elements, it is trivial to see that the equations (68) can be rewritten as:

$$\{\mathbf{X}'\mathbf{X} + \lambda \boldsymbol{A} \otimes \boldsymbol{I}_d\}\mathbf{w} = \mathbf{X}'\mathbf{y} \quad (70)$$

where $\boldsymbol{A} \otimes \boldsymbol{I}_d$ is the Kronecker Product of matrix $\boldsymbol{A} \in R^{T\times T}$ (defined bellow) and identity.

$$\boldsymbol{A} = \begin{bmatrix} \gamma + \sum_{j\in\mathcal{E}_1} sim(1,j) & -sim(1,2) & .... & -sim(1,T) \\ -sim(2,1) & \gamma + \sum_{j\in\mathcal{E}_2} sim(2,j) & .... & -sim(2,T) \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ -sim(T,1) & -sim(T,2) & .... & \gamma + \sum_{j\in\mathcal{E}_T} sim(T,j) \end{bmatrix} \quad (71)$$

Thus, the closed-form solution for the MTL parameters vector becomes:

$$\mathbf{w} = \{\mathbf{X}'\mathbf{X} + \lambda \boldsymbol{A} \otimes \boldsymbol{I}_d\}^{-1}\mathbf{X}'\mathbf{y} \quad (72)$$

which has the same form of (58). If we state that $\mathbf{P}(0) = \{\lambda \boldsymbol{A} \otimes \boldsymbol{I}_d\}^{-1}$, then we can make every new task input vector as an input vector of dimension dT with zeros in all indices other than those of the specific task and feed the WRLS algorithm with it.

The computational complexity of WRLS at each iteration for each task is $O(d^2)$. When we operate on the stacked multi-task space, this complexity becomes $O(d^2 \times T^2)$, which is lower than most non linear approximated solutions presented in literature. Its initialization requires the computation of $\{\lambda \boldsymbol{A} \otimes \boldsymbol{I}_d\}^{-1} = \{\lambda^{-1}\boldsymbol{A}^{-1} \otimes \boldsymbol{I}_d\}$ whose cost is usually $O(T^3)$. It may seem as expensive as the other proposals, however this cubic burden is, on the other hand, only required once before the arrival of the first instances into the MT-WRLS algorithm.

Algorithm 1 Multi-task Weighted Recursive Least Squares

Define the similarities $sim(t, s); t, s \in 1, 2, ..., T$
Set $\gamma, \lambda > 0$ and $i \leftarrow 0$
Compute $\boldsymbol{A}$ according to Eq. (71)
Set $\mathbf{P}(0) \leftarrow \lambda^{-1}\boldsymbol{A}^{-1} \otimes \boldsymbol{I}_d$
Set $\mathbf{w}(0) \leftarrow \mathbf{0}_{dT}$
for $n = 1, 2, ...$ **do**
  for $t = 1, 2, ..., T$ **do**
    Set $i \leftarrow i + 1$
    Predict $\hat{y}_t(n)$ with $\mathbf{w}_t(i-1)$ and $\mathbf{x}_t(n)$
    Set $\mathbf{x}(i)$ as the stacked vector of $\mathbf{x}_t(n)$
    Compute $\mathbf{k}(i)$ according to Eq. (61)
    Compute $\boldsymbol{\alpha}(i), \mathbf{w}(i)$ using Eqs. (62) and (63)
    Compute $\boldsymbol{P}(i)$ resorting to Eq. (64)

5.3 Multi-task Online Sparse Least Squares Support Vector

Kernel methods have established their fame and success due mostly to their ability of mapping, linearly and non linearly, the input space onto a feature hyperspace of finite or infinite dimensions without explicitly doing so. The kernel function, computed more easily on the original input space, can provide enough information to allow the methods to learn very difficult patterns. Since they usually work on the dual space, their batch-style optimization complexity is commonly defined by $O(N^3)$ (N the number of training examples), which does not scale well with the increasing amount of instances of data streams.

Recursive kernel methods for regression were proposed to easy this burden, but none of them have considered an online multi-task approach. As done in the MT-WRLS case, we selected a method that recursively solved a general single task static problem and adapted it to our MTL framework. In this case, it is done by choosing the appropriate kernel function and demonstrating the equivalence between the adapted optimization problem and our graph-based MTL original formulation.

First, let us define the Least Squares Support Vector problem as done in [6]. It is based on the dual formulation of the least squares problem and considers all of the examples in the training set as support vectors. Its general optimization structure is given by:

$$\boldsymbol{\alpha}_* = \arg\min_{\boldsymbol{\alpha}} \|\mathbf{K}\boldsymbol{\alpha} - \mathbf{Y}\|^2 + \lambda \boldsymbol{\alpha}' \mathbf{K} \boldsymbol{\alpha} \tag{73}$$

$$\hat{\mathbf{Y}} = \mathbf{K}\boldsymbol{\alpha} \tag{74}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^N$ is the parameters vector, N the number of examples and $\mathbf{K} \in \mathbb{R}^{N \times N}$ is the kernel matrix defined as follows:

$$\mathbf{K}_{[i,j]} = \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle \tag{75}$$

with $\boldsymbol{\phi}(\mathbf{x}_i)$ being the mapping of $\mathbf{x}_i$ (i-th row of $\mathbf{X}$) onto an specific hyperspace.

The work of [6] proposes a recursive solution for $\boldsymbol{\alpha}_*(n)$ (73), $n \in 1, ..., N$ (Online Sparse Least Squares Support Vector, OSLSSVR), having $\mathbf{K}(n)$ and $\mathbf{Y}(n)$ as time-dependent elements of the problem. Their contribution is incremental in the sense that a recursive solution of (73) with $\lambda = 0$ was already proposed by [36] with the so-called Kernel Recursive Least Squares (KRLS). In fact, $\lambda > 0$ includes a regularization term into the learning procedure, which is fundamental for us to achieve a dual equivalence of (57) and, therefore, apply the OSLSSVR method to obtain a recursive kernel-based multi-task method. This equivalence is supported by the Representer Theorem and is detailed in the next paragraphs.

Let $\mathbf{w} \in R^{dT}$ be the stacked parameters vector of the T tasks and let $\mathbf{x}_{(n,s)} = (0, ..., \mathbf{x}_s(n)', ..., 0)' \in R^{dT}$ be the stacked input vector of task s at time n. Let still the RKHS H of dimension dT with inner product $\langle \boldsymbol{u}, \boldsymbol{v} \rangle_H = \boldsymbol{u}'(\boldsymbol{A} \otimes \boldsymbol{I}_d)\boldsymbol{v}$, with $\boldsymbol{A}$ symmetric and positive-definite. The mapping function of the stacked input vector onto the RKHS is defined by:

$$\boldsymbol{\phi} : R^{dT} \to H, \boldsymbol{\phi}(\mathbf{x}_{(n,s)}) = \boldsymbol{A} \otimes^{-1} \mathbf{x}_{(n,s)} \tag{76}$$

for simplicity, we write $\boldsymbol{A} \otimes \boldsymbol{I}_d$ as $\boldsymbol{A}\otimes$ and $\boldsymbol{A}^{-1} \otimes \boldsymbol{I}_d$ as $\boldsymbol{A}\otimes^{-1}$

The kernel function of two input vectors of any tasks s and t, at any time points n and l, is given by:

$$\begin{aligned} K(\mathbf{x}_s(n), \mathbf{x}_t(l)) &= \langle \phi(\mathbf{x}_{(n,s)}), \phi(\mathbf{x}_{(l,t)}) \rangle \\ &= (\mathbf{x}'_{(n,s)} \boldsymbol{A} \otimes^{-1}) \boldsymbol{A} \otimes (\boldsymbol{A} \otimes^{-1} \mathbf{x}_{(l,t)}) \\ &= \mathbf{x}'_{(n,s)} \boldsymbol{A} \otimes^{-1} \mathbf{x}_{(l,t)} = \mathbf{x}_s(n)' \mathbf{x}_t(l) \boldsymbol{A}^{-1}_{[s,t]} \end{aligned} \tag{77}$$

We can see that the kernel function can be easily computed by the inner product of the input vectors in their original space ($dim = d$) multiplied by a factor that depends on their relationship. It will allow us to avoid computations on R^{dT}, implying in cost reduction.

Let us demonstrate that, with the appropriated choice for A, we can turn the multi-task optimization problem into the same kernel formulation as (73), enabling us to employ the OSSLSVR algorithm as a recursive multi-task method.

The Representer Theorem guarantees that, under certain conditions, there is a kernel-based solution for the Empirical Risk minimization problem:

$$f_* = \arg\min_f \|f(\mathbf{X}) - \mathbf{y}\|^2 + \lambda \|f\|^2_H \tag{78}$$

$$f_* = \sum_{t=1}^{T}\sum_{j}^{n} \alpha_{t,j} K(\mathbf{x}_t(j), .) \tag{79}$$

where we explicitly wrote double summations only to represent the entire dataset of all tasks stacked.

It is well known [6] that the combination of (78) and (79) results in (73), for which the parameters to be optimized are those associated with the support vectors. We now show that (78) is equivalent to our MTL problem (57).

Writing f in terms of a parameters vector $\boldsymbol{E}$ in the hyperspace:

$$f = \sum_{t=1}^{T}\sum_{j}^{n} \alpha_{t,j} K(\mathbf{x}_t(j), .) = \sum_{t=1}^{T}\sum_{j}^{n} \alpha_{t,j} \langle \boldsymbol{\phi}(\mathbf{x}_{(j,t)}), \boldsymbol{\phi}(.)\rangle$$
$$= \langle \sum_{t=1}^{T}\sum_{j}^{n} \alpha_{t,j} \boldsymbol{\phi}(\mathbf{x}_{(j,t)}), \boldsymbol{\phi}(.)\rangle = \langle \boldsymbol{e}, \boldsymbol{\phi}(.)\rangle \tag{80}$$

However $\langle \boldsymbol{e}, \boldsymbol{\phi}(\mathbf{x}_{(n,s)})\rangle = \langle \boldsymbol{e}, \boldsymbol{A}\otimes^{-1}\mathbf{x}_{(n,s)}\rangle = \boldsymbol{e}'\boldsymbol{A}\otimes\boldsymbol{A}\otimes^{-1}\mathbf{x}_{(n,s)} = \boldsymbol{e}'\mathbf{x}_{(n,s)}$. Hence, the parameters vector that acts in the hyperspace is equivalent to the vector $\mathbf{w}$ and, therefore, $f(\mathbf{X}) = \boldsymbol{X}\mathbf{w}$.

If we adopt $\boldsymbol{A}$ as done before in the MT-WRLS section (see Eq. (71)), then we obtain:

$$\|f\|_H^2 = \|\boldsymbol{e}\|^2 = \langle \boldsymbol{e}, \boldsymbol{e}\rangle = \boldsymbol{e}'\boldsymbol{A}\otimes\boldsymbol{e} = \sum_{t=1}^{T}\sum_{j\in\rceil_t} \|\mathbf{w}_t sim(j,t) - \mathbf{w}_j sim(t,j)\|^2 + \gamma\sum_{t=1}^{T}\|\mathbf{w}_t\|^2 \tag{81}$$

Therefore, Eq. (78) can be rewritten as:

$$f_* = \arg\min_f \|\boldsymbol{X}\mathbf{w} - \boldsymbol{Y}\|^2 + \lambda\{\sum_{t=1}^{T}\sum_{j\in\rceil_t} \|\mathbf{w}_t sim(j,t) - \mathbf{w}_j sim(t,j)\|^2 + \gamma\sum_{t=1}^{T}\|\mathbf{w}_t\|^2\} \tag{82}$$

which is fully analogous to our graph-based MTL formulation.

Then, we found a multi-task kernel (defined by $\boldsymbol{A}$) which can be adopted in the OSLSSVR algorithm to recursively learn and promote sharing information according to the degree of relationship between the tasks. As the iteration cost of OSLSSVR method is $O(m_n^2)$, with m_n being the size of the support vectors dictionary (usually $m_n << nT$ or limited to a constant), multiplied by the complexity of computing the kernel function, which is $O(d)$, then the per-instance cost of MT-OSLSSVR is $O(d \times m_n^2)$.

The major advantage of this method in comparison to the proposed MT-WRLS is that the multi-task kernel can be computed using individual task input vectors of dimension d, instead of working on the stacked space of dimension dT.

Algorithm 2 Multi-Task Online Sparse Least Squares Support Vector Regression

Define the similarities $sim(t, s); t, s \in 1, 2, ..., T$
Set $\gamma, \lambda > 0$ and $i \leftarrow 0$
Compute $\boldsymbol{A}$ according to Eq. (71)
Set the kernel function using Eq. (77)
Initialize the OSLSSVR algorithm as described in section 4
for $n = 1, 2, ...$ **do**
  for $t = 1, 2, ..., T$ **do**
    Predict $\hat{y}_t(n)$ with $\mathbf{x}_t(n)$ and OSLSSVR as reformulated in section 4
    Train the reformulated OSLSSVR (section 4) with $\mathbf{x}_t(n)$ and $y_t(n)$

5.4 Experimental Setup

5.4.1 Online MTL Contenders

We implement two online MTL contenders in order to be able to partially rank our proposals among the literature of online Multi-Task Learning, which was demonstrated to be lacking of reproducible time-dependent regression experiments. These methods are the Online Gradient Descent of our graph-based formulation, which was already proposed by [1] (hereinafter referred to as MOGD), and the proposal of [22] presented before as an online single-run of the ADMM optimization procedure (MADMM).

5.4.2 Nonlinear Online MTL with Extreme Learning Machines

We also combine the original WRLS and our MT-WRLS proposal with Extreme Learning Machines (ELM), which are Single Hidden Layer Feedforward Neural Networks [5] capable of providing a nonlinear input-output mapping with a simpler training process, sampling the hidden layer weights and estimating the output layer weights via ordinary least squares or some sort of regularized regression.

Let $\mathbf{x}_t \in \mathbb{R}^d$ be an input vector from task t. The respective SHLFNs output prediction is given by:

$$\hat{y}_t = \sum_{k=1}^{H} \beta_{t,k} f(\mathbf{x}_t^T \mathbf{v}_k) + \beta_{t,0}, \tag{83}$$

where H is the number of hidden nodes, $\mathbf{v}_k \in \mathbb{R}^d$ is the hidden layer weight vector of node k, f is the nonlinear activation function and $\boldsymbol{\beta_t} = [\beta_{t,0}, \beta_{t,1}, \beta_{t,2}, ..., \beta_{t,H}] \in \mathbb{R}^{H+1}$ is the parameters vector of the output layer.

The number of neurons in the hidden layer, the sampling method of their weights and the adoption of the hyperbolic tangent as activation function are important parameters of the ELM that may have a direct impact on its performance. We follow the same methodology to deal with these aspects as done by [4].

5.4.3 Online Regression Benchmark

Due to the lack of multi-task time-dependent regression benchmarks, we only test our proposals on the wind speed forecasting dataset proposed by [4]. This case study consists

of $T = 10$ time series of wind speed from wind sites located in Miami, United States. They were extracted from the Wind Integration National Dataset (WIND) Toolkit [49], which is an important meteorological database of thousands of wind sites from all over the U.S. territory, with data collected every 5 minutes from 2007 up to 2012. The wind speed intensities were registered at the height of 100 meters and are expressed in meters per second (m/s). Aiming to perform one-step-ahead forecasts (5 minutes), the authors adopted the first three months of 2012, resulting in approximately 24 thousand points for each time series.

We pursued the proposed methodology and sampled 30 multi-task subsets C_k ($k \in 1, 2..., 30$) from the 10 original time series, ending up with 30 multi-task datasets of wind speed prediction containing 10 tasks of 400 points each, which allows us to investigate the accuracy of the proposed methods in 30 different benchmarks.

Fig. 5 and 6 illustrate three differentiated wind speed series, 1, 5 and 10 (each one seen as a prediction task) for four different subsets, C_1, C_{13}, C_{23} and C_{29}. As we may note, the time series present a variety of patterns along time, reinforcing the need for efficient online learning methods capable of handling changes in data distribution. On the other hand, for each subset C_k, the similarity between the time series becomes evident, encouraging even more the adoption of multi-task learning.

The authors also aimed at studying the ability of MTL methods to better generalize in the presence of less training data as mentioned before, adopting, for each subset C_k, two different percentages of sequential training and test split: $\mu \in \{27.5\%, 45\%\}$. We shall refer to the first percentage as Experiment I and the second one as Experiment II.

In their work [4], the adoption of first-order differentiation to remove linear trends was responsible for achieving higher general performance, which motivated us to only work with differentiated series in this experiment. To ease the burden of selecting the best size of the auto-regressive vector, we set the past 9 occurrences of the series as the input for the step-ahead predictor as follows:

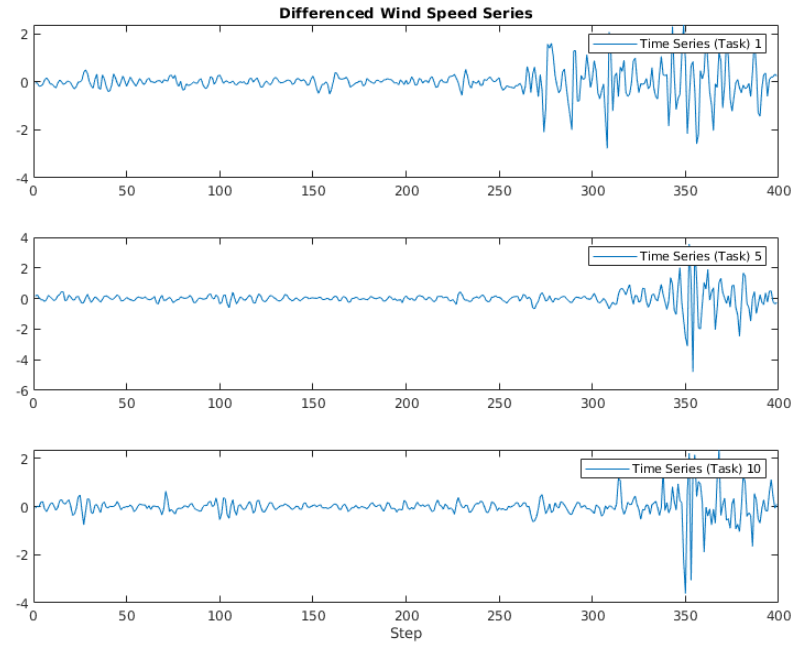
$$\hat{y}^{[i]} = w_1 y^{[i-1]} + w_2 y^{[i-2]} + w_3 y^{[i-2]} + w_4 y^{[i-4]} + w_5 y^{[i-5]} + w_6 y^{[i-6]} + w_7 y^{[i-7]} + w_8 y^{[i-8]} + w_9 y^{[i-9]} + w_0. \quad (84)$$

where the max lag for the auto-regressive vector is set to 9, as our preliminary experiments, following the same methodology of [4], indicated to be the most frequent result. This is done in order to reduce the number of hyperparameters to be tuned, improving the parsimony and the understanding of our experimental setup.

5.4.4 Optimization of Hyperparameters

For each subset C_k and train/test split percentage μ, we optimize the hyperparameters of the recursive methods on training set and run the online method on test set, as done in [4]. Then, the training set is not used to initialize the parameters of the online methods, which shall be learnt from scratch on test set. The pairwise similarities were computed by the Spearman Correlation between the series in the training sets.

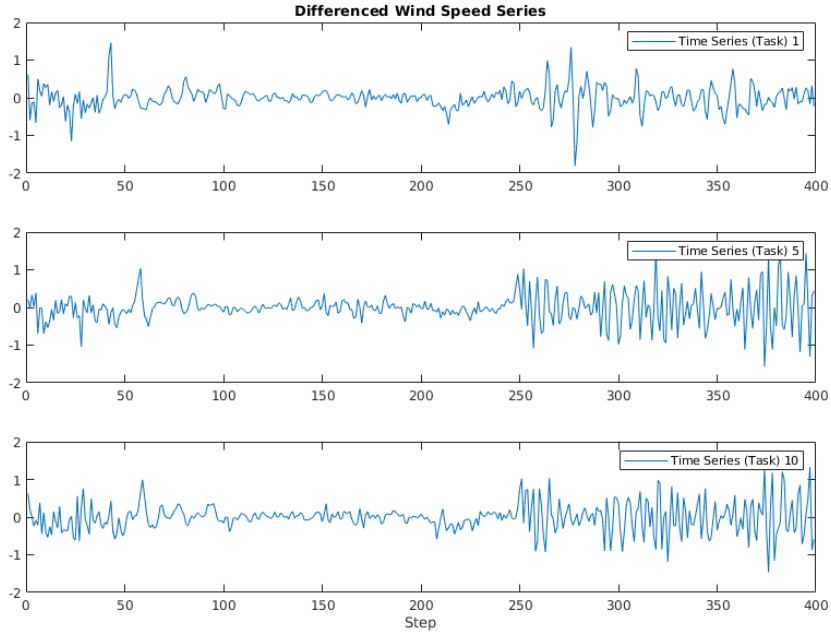
(a)



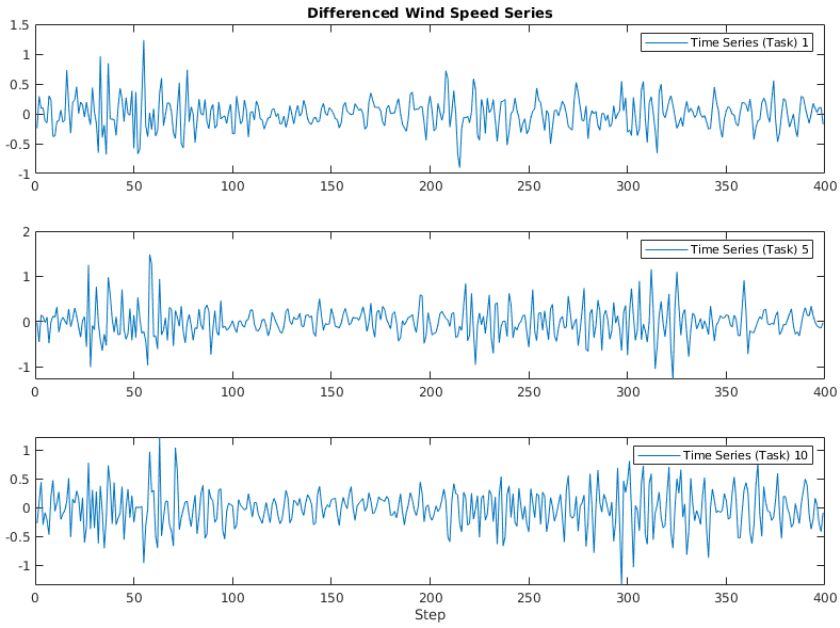
(b)

Figure 5: Differentiated wind speed series 1, 5 and 10, each one representing a prediction task t, for subsets (a) C_1 (b) C_{13}.

For the MT-WRLS proposal, we tune the hyperparameters σ (forgetting factor) and λ (penalization of the regularization term) within the ranges $\{0.01, 0.2, 0.4, 0.6, 0.8, 1.0\}$ and $\{10^{-10}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}, 10^{10}\}$, respectively. For MT-OSLSSVR method, we tune ν (sparsity control of OSLSSVR [6]) and λ (penalization of the regularization term) within $\{10^{-3}, 10^{-2}, 10^{-1}\}$ and $\{10^{-10}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}, 10^{10}\}$,



(a)



(b)

Figure 6: Differentiated wind speed series 1, 5 and 10, each one representing a prediction task t, for subsets (a) C_{23} (b) C_{29}.

respectively.

5.4.5 Procedures for Comparing our Proposals with Contenders

We compare our proposals with online methods, original single-task WRLS and OSLSSVR, and with batch methods, those studied in [4]. The relative root mean square error

(RELRMSE) and the relative mean absolute error (RELMAE), which are not scale sensitive, are the metrics employed by the authors to summarize the performance of all T tasks in each C_k, avoiding comparison in different scales. They are computed as the ratio of the root mean square error (RMSE) and the mean absolute error (MAE) produced by the predictions of the experimented methods and those produced by a benchmark model, detailed below.

$$RELRMSE(t,k,\mu) = \frac{RMSE(t,k,\mu)}{RMSE_{persistence}(t,k,\mu)} \tag{85}$$

$$RELMAE(t,k,\mu) = \frac{MAE(t,k,\mu)}{MAE_{persistence}(t,k,\mu)} \tag{86}$$

where the adopted benchmark is the persistence method, whose one-step-ahead forecasts are given simply by $\hat{y}_t^{[i]} = y_t^{[i-1]}$.

$$RMSE(t,k,\mu) = \sqrt{\frac{1}{n_\mu}\sum_{i=1}^{n_\mu}(\hat{y}_t^{[i]} - y_t^{[i]})^2} \tag{87}$$

$$MAE(t,k,\mu) = \frac{1}{n_\mu}\sum_{i=1}^{n_\mu}|\hat{y}_t^{[i]} - y_t^{[i]}| \tag{88}$$

where n_μ is the length of the respective test set.

We compute the mean metrics over all the T tasks in order to assess the average performance of each method per experiment (training percentage). Once we have the results for our two alternative proposals and the other contenders, we submit the mean $RELRMSE(k,\mu)$ to the Friedman Test followed by the post-hoc Fisher Test (or LSD test) [50], which are non-parametric hypothesis tests designed to measure the statistical significance of the difference between the ranks of distinguished treatments. Having different mean ranks at a significance level of 0.05, we account a victory to the learning method with a lower mean rank and a defeat for the other.

5.5 Results and Discussion

We present, in Table 1, the mean RELRMSE and mean RELMAE for each contender of Experiment I. As we shall see, the MT-WRLS, MT-WRLS+ELM and MT-OSLSSVR were the ones with the lowest errors among the investigated methods, providing up to 16% of accuracy improvement when comparing to their STL versions. Our proposals also surpass the performance of other multi-task recursive algorithms, demonstrating the benefits of having more controlled convergence guarantees.

One may note as well that the combination of online learning and ELM was not responsible for lower errors, showing that adding more complexity to the problem does not always translate to better solutions, possibly because our recursive proposal is already flexible enough.

In Table 2, we display the outcome of our hypothesis tests for Experiment I, corroborating to the results presented before. It shows that our online MTL proposals are responsible for

Table 1: Performance Metrics for Experiment I

Learning Method	RELRMSE	RELMAE
MT-OSLSSVR	**0.7498**	**0.7308**
MT-WRLS	**0.7502**	**0.7310**
MT-WRLS + ELM	**0.7700**	**0.7550**
WRLS	0.8467	0.8236
WRLS + ELM	0.8490	0.8329
OSLSSVR	0.8892	0.8214
MOGD + ELM	0.8981	0.8780
MOGD	0.9336	0.9191
MADMM	1.1244	1.0105

the highest amount of victories with the lowest mean ranks, implying in statistical significance of their superiority.

We can not tell, however, which of our two proposals performed better as there was not sufficient statistical evidence to reject the hypothesis of their mean ranks being equable.

Table 2: Results of Hypothesis Test for Experiment I

Learning Method	# Victories	# Defeats	Mean Rank
MT-WRLS	**6**	**0**	**1.77**
MT-OSLSSVR	**6**	**0**	**2.00**
MT-WRLS + ELM	**6**	**0**	**2.70**
WRLS	2	3	5.13
WRLS + ELM	2	3	5.40
OSLSSVR	1	3	6.10
MOGD + ELM	1	3	6.37
MOGD	0	5	7.37
MADMM	0	7	8.17

Tables 3 and 4 show the mean metrics and the outcome of hypothesis tests for Experiment 2. We draw similar analysis, endorsing the superiority of our proposals in comparison to online STL and other online MTL contenders.

We verify the relatedness of our proposals by their similar mean RELRMSE and mean RELMAE, evidencing, in practice, that both methods MT-WRLS and MT-OSLSSVR resort to the same MTL formulation.

Table 3: Performance Metrics for Experiment II

Learning Method	RELRMSE	RELMAE
MT-WRLS	**0.7470**	**0.7274**
MT-OSLSSVR	**0.7476**	**0.7279**
MT-WRLS + ELM	**0.7600**	**0.7406**
WRLS + ELM	0.8336	0.8196
WRLS	0.8406	0.8195
OSLSSVR	0.8765	0.8170
MOGD + ELM	0.9023	0.8824
MOGD	0.9320	0.9180
MADMM	1.1704	0.9779

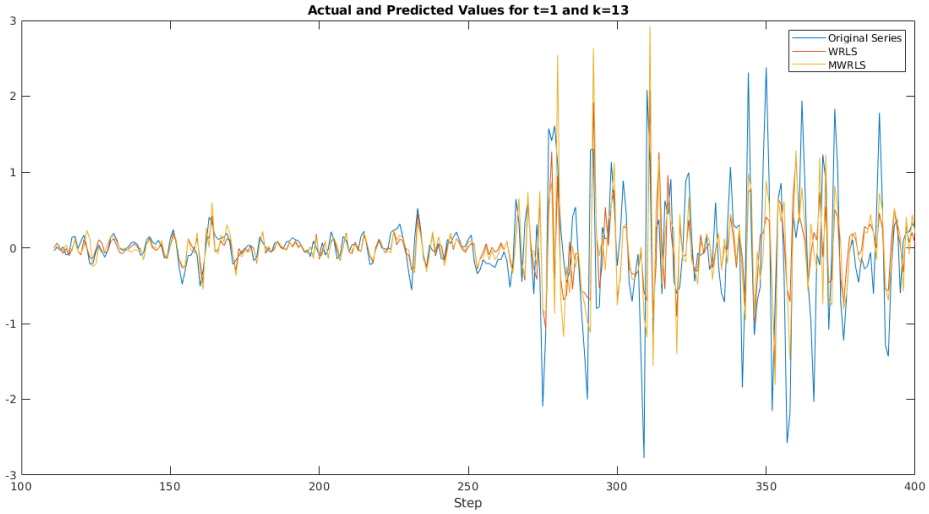


Figure 7: Actual and Predicted values for time series 1 and subset C_{13} of Experiment I.

Table 4: Results of Hypothesis Test for Experiment II

Learning Method	# Victories	# Defeats	Mean Rank
MT-WRLS	**6**	**0**	**1.8700**
MT-OSLSSVR	**6**	**0**	**2.2300**
MT-WRLS + ELM	**6**	**0**	**2.5000**
WRLS	2	3	5.3300
WRLS + ELM	2	3	5.1000
OSLSSVR	1	3	5.9300
MOGD + ELM	1	3	6.2700
MOGD	0	5	7.2700
MADMM	0	7	8.5000

Fig. 7 and 8 illustrate the benefits of one of our proposals, the MT-WRLS method, against its single-task version for time series (task) 1 and subsets C_{13} and C_{23} of Experiment I. The more appealing gains are located in the regions of higher frequencies and higher variations, where the online MTL method was able to provide more accurate predictions.

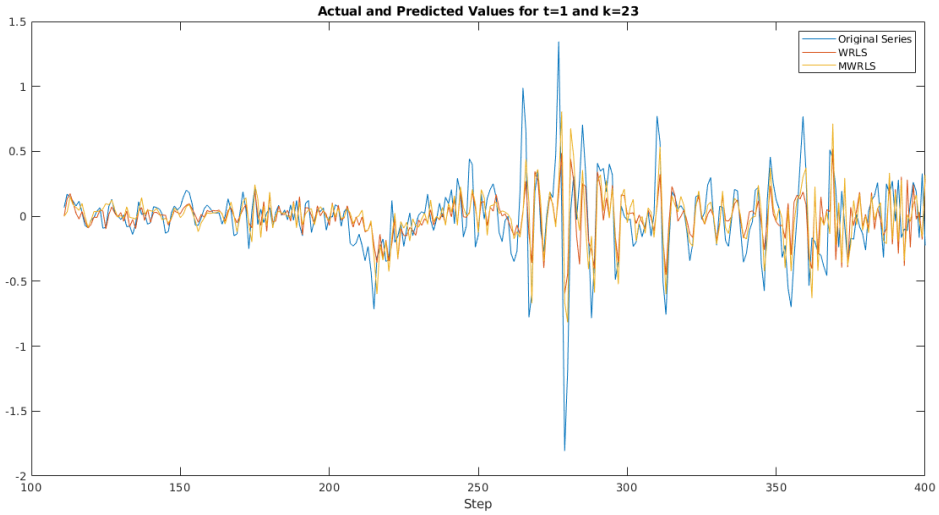


Figure 8: Actual and Predicted values for time series 1 and subset C_{23} of Experiment I.

6 Application of Online MTL to Evolving Systems

In this section, we apply a suitable online MTL method to EVeP in order to reduce its iteration complexity (subsections 6.1 and 6.2). In subsection 6.3, we describe the experimental setup, adopting synthetic and real-world benchmarks, designed to assess its performance and computational complexity against other candidates in literature. The results of these experiments are presented and discussed in subsection 6.4.

6.1 Extreme Value evolving Predictor (EVeP)

Founded on the Extreme Value Theory [51], the Extreme Value evolving Predictor (EVeP) [30] resorts to a statistically well-founded approach for the definition of the information granules at the antecedent and consequent parts of eFRB systems. It employs Weibull distributions, a statistically consistent limiting distribution capable of capturing the relative proximity among the data points supporting the rules. Equation (89) contains the expression for the membership function $\mu^i(\boldsymbol{z}^{[t]})$ of a data point $\boldsymbol{z}^{[t]}$, available at time instant t, for the rule R^i with center $\boldsymbol{z_0^i}$:

$$\mu^i(\boldsymbol{z}^{[t]}) = \Psi^i(||\boldsymbol{z_0^i} - \boldsymbol{z}^{[t]}||, \kappa^i, \lambda^i) = \exp\left[-\left(\frac{||\boldsymbol{z_0^i} - \boldsymbol{z}^{[t]}||}{\lambda^i}\right)^{\kappa^i}\right], \tag{89}$$

where $||\boldsymbol{z_0^i} - \boldsymbol{z}^{[t]}||$ is the distance from $\boldsymbol{z}^{[t]}$ to center $\boldsymbol{z_0^i}$, and κ^i, λ^i are, respectively, the Weibull shape and scale parameters obtained automatically by fitting the distribution taking into account the relative proximity between the data points belonging to that rule and all the data points of the other rules.

To manage the fuzzy granules forming the antecedent and the consequent parts of rules, EVeP introduces a novel evolving fuzzy clustering algorithm based on this membership function. The decision of allocating a new data point $\boldsymbol{x}^{[t]}$ to an existing rule or creating a new rule R^{i^*}, $i^* = c + 1$ is given by (90):

$$i^* = \begin{cases} \arg\ \max_{i=1}^{c} \mu^i & \text{if } \exists i \in \{1, \ldots, c\} | (\mu^i \geq \sigma) \\ c+1 & \text{otherwise} \end{cases}. \tag{90}$$

where $\sigma \in [0, 1]$ is a threshold parameter defining the boundary between the set of existing rules R^i, $i = 1, \ldots, c$, and the open space for which there is no rule defined. If there is at least one rule for which the membership degree is superior to this threshold degree, then the new data point is allocated to the rule of maximum membership degree. Otherwise, a new rule is created.

To calculate the parameters $\boldsymbol{\theta}^i$, $i = 1, \ldots, c$, of the TS consequent part of the rules, EVeP takes advantages of the proven benefits [28] of sharing information among the rules by means of a multi-task learning (MTL) approach, employing a generalization of the Sparse Structure-Regularized Learning with Least Squares Loss (Least SRMTL) [9, 29] to represent the structural dependencies among the rules. A sliding window [52] (mini-batch approach in MTL) is employed to keep the maximum number of input-output data points assigned to the i^{th} rule R^i fixed at N^*. Considering the training data set $\{\boldsymbol{x}^{i[t]}, y^{i[t]}\}_{t=1}^{N^*}$, $\boldsymbol{x}^{i[t]} \in \mathbb{R}^n$, $y^{i[t]} \in \mathbb{R}$, then the matricial form of the input-output dataset associated with the i^{th} rule is expressed by:

$$X^i = \begin{bmatrix} 1 & \boldsymbol{x}^{i^{[1]}T} \\ 1 & \boldsymbol{x}^{i^{[2]}T} \\ \vdots & \vdots \\ 1 & \boldsymbol{x}^{i^{[N^*]}T} \end{bmatrix}, \quad \boldsymbol{y}^i = \begin{bmatrix} y^{i^{[1]}} \\ y^{i^{[2]}} \\ \vdots \\ y^{i^{[N^*]}} \end{bmatrix}. \tag{91}$$

The final optimization problem to calculate the matrix of parameters $\Theta = [\boldsymbol{\theta}^1, \boldsymbol{\theta}^i, \ldots, \boldsymbol{\theta}^c]$ is represented by (92):

$$\Theta^* = \arg \min_{\Theta} \sum_{i=1}^{c} ||X^i \boldsymbol{\theta}^i - \boldsymbol{y}^i||_2^2 + \rho ||\Theta G||_F^2, \tag{92}$$

where $||\cdot||_F^2$ is the squared Frobenius norm and $||\cdot||_2^2$ is the squared l_2-norm. The expression $\rho||\Theta G||_F^2$ is a regularization term that encodes the structural dependencies among the learning tasks.

Once defining the similarity measure $s(R^{i_1}, R^{i_2})$ between every pair of rules $i_1, i_2 = 1, 2, \ldots, c$ (for more information one may consult [30], the similarity matrix S representing the pairwise dependencies among the rules is provided by (93):

$$S = \begin{bmatrix} s(R^1, R^1) & s(R^1, R^2) & \ldots & s(R^1, R^c) \\ s(R^1, R^2) & s(R^2, R^2) & \ldots & s(R^2, R^c) \\ \vdots & \vdots & \ddots & \vdots \\ s(R^1, R^c) & s(R^2, R^c) & \ldots & s(R^c, R^c) \end{bmatrix} \tag{93}$$

To define matrix $\mathbf{G}$ of (92), Ayres and Von Zuben [30] introduced a graph where each rule is a node, and an edge connects two nodes if their corresponding rules R^{i_1} and R^{i_2} are related, i.e. if $s(R^{i_1}, R^{i_2}) > 0$. To facilitate the manipulation of the algebraic structure, each edge of the graph is represented by a vector where the elements corresponding to the connected rules are set to the respective values of matrix S and all the other elements are set to zero. Mathematically, let $\mathcal{E}$ be the set of edges, the edge k is represented as a vector $\boldsymbol{e}^k \in \mathbb{R}^c$ defined as follows: $\boldsymbol{e}^k_{i_1} = s(R^{i_1}, R^{i_2})$, $\boldsymbol{e}^k_{i_2} = -s(R^{i_1}, R^{i_2})$ and $\boldsymbol{e}^k_i = 0, i = 1, \ldots, c, i \neq i_1, i \neq i_2$, if $s(R^{i_1}, R^{i_2}) > 0$. The complete graph is represented by matrix $\mathbf{G} = [\boldsymbol{e}^1, \boldsymbol{e}^2, \ldots, \boldsymbol{e}^{||\mathcal{E}||}] \in \mathbb{R}^{c \times |\mathcal{E}|}$, where $|\mathcal{E}|$ is the cardinality of set $\mathcal{E}$ [28, 29].

The user-defined parameter ρ of (92) controls the influence of the regularization term in the calculation of the matrix of Takagi-Sugeno parameters Θ. The term $||\Theta \mathbf{G}||_F^2$ forces the related tasks (informed in matrix $\mathbf{G}$) to exhibit a reduced Euclidean distance between the corresponding pair of columns of matrix Θ [28]. When compared to what would happen in the single-task learning approach ($\rho = 0$), the larger the relation, the more intense the reduction in such Euclidean distance.

6.2 Online Convex Optimization for Multi-task Learning in EVeP

An online gradient-based method that has proved to be class-efficient is the Adagrad (Adaptive Gradient Algorithm) [19, 53]. Essentially, Adagrad uses adjustable and distinct learning rates for each parameter, in response to the recent learning experience. In online learning,

Adagrad may exhibit a nice progress per iteration and holds theoretical convergence guarantees, better detailed in [19, 53]. Equations (94), (95) and (96) define the gradient of the MTL loss function and the Adagrad online optimization step for each rule $i \in [1, ..., c]$.

$$\nabla \boldsymbol{\theta}^i_{[t]} = 2{X^i_{[t]}}^T X^i_{[t]} \boldsymbol{\theta}^i_{[t]} - 2{X^i_{[t]}}^T \boldsymbol{y}^i_{[t]} + 2\rho \sum_{k=1}^{c} (s(R^i, R^j)\boldsymbol{\theta}^i_{[t]} - s(R^j, R^i)\boldsymbol{\theta}^j_{[t]}) \quad (94)$$

$$M^i_{[t]} = M^i_{[t-1]} + (\nabla \boldsymbol{\theta}^i_{[t]})(\nabla \boldsymbol{\theta}^i_{[t]})^T \quad (95)$$

$$\boldsymbol{\theta}^i_{[t+1]} = \boldsymbol{\theta}^i_{[t]} - \frac{\lambda \nabla \boldsymbol{\theta}^i_{[t]}}{\sqrt{diag(M^i_{[t]}) + \epsilon I}} \quad (96)$$

having $M^i_{[t]}$ as the gradient momentum matrix at step t, whose use is mostly for the ease of representation as we are only concerned with its diagonal elements. $\boldsymbol{\theta}^i_{[t]}$ represents the online learnt parameters of each consequent, while λ and ϵ are static parameters of Adagrad, which define the initial learning rate and assure non-singularity, respectively.

Still keeping its recursive nature, Adagrad was implemented here using a sliding window over which the gradient is calculated per step, being an online adaptation of batch learning. This approach is intended to smooth the effects of potential noise on gradient calculation. We let, therefore, $N*$, λ and ϵ be tuned along with the other inherited hyperparameters.

In EVeP, before being updated, the consequent parameters can be reset to zero. Several iterations of gradient descent drive the consequent parameters (close) to their optimal values. In our online method, the consequent parameters should not be learned from scratch. Therefore, zeroing their initial values is not a good option. So, we have to define what to do when new rules are created or removed, and/or rules are merged. The following heuristics was then conceived: after the creation of a rule, the consequent of old rules are kept and the new rule starts its consequent parameters as the simple average of the others. For the merging or removal of a rule, we keep all the consequent parameters of the remaining rules. In this manner, prior knowledge derived from past prediction tasks is used as our starting point for the online updating.

6.3 Experimental Setup

Our proposal was experimented on six benchmark datasets to assess its capability of maintaining a high performance while reducing the computational cost. These case studies are widely used in the evolving systems literature, allowing us to compare the accuracy of our model against other well established contenders.

We report, for each experiment, up to two performance metrics, depending on their availability in the literature, the Root Mean Squared Error (RMSE) and the Nondimensional Error Index (NDEI) (97). It is also provided statistics about the temporal cost per step, which is extended only to the original EVeP model, as we focus on assessing the gains and losses when adapting it to work with our online MTL framework.

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(y^{[t]} - \hat{y}^{[t]})^2}, \quad NDEI = \frac{RMSE}{std(y)} \tag{97}$$

As done in [8], we adopted the Optuna framework [54] to find the most suitable hyperparameters for our model. We use the same intervals [8] for the EVeP inherited hyperparameters of each benchmark and add the intervals $[10^{-5}, 10^{-2}]$ and $[10^{-3}, 10^{2}]$ for ϵ and λ, respectively. For the EVeP itself, we reproduce the same user-defined parameters as set in its original paper. It was also adopted the same prepocessing [8] for the first three experiments and that of [28] for the weather temperature benchmark.

Here we shall comment that some experiments were executed employing the same data for both hyperparameters optimization and online testing, which weakens the evidence of its performance in real-world scenarios, in which we usually do not have access to future information, forcing us to select all the defining parameters using only present and past data. Nonetheless, other models in literature adopted this procedure as well, what enables fair comparisons.

The experiments were performed in an Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz.

- **Nonlinear Dynamic Plant Identification With Time-Varying Characteristics**

First proposed by Nguyen et al. [46], this experiment is based on the following model:

$$y(t+1) = \frac{y(t)}{1+y^2(t)} + u^3(t) + n(t), \tag{98}$$

where $u(t) = \sin(2\pi t/100)$ and $n(t)$ is a time-varying factor, provided by (99):

$$n(t) = \begin{cases} 0, & 1 \leq t \leq 1000 \text{ and } t \geq 2001 \\ 0.5, & 1001 \leq t \leq 1500 \\ 1, & 1501 \leq t \leq 2000 \end{cases}. \tag{99}$$

As done in [8], we used the initial 3000 points for both hyperparameters optimization and online testing. They were set to $\sigma = 0.2370$, $\delta = 36$, $N^* = 1$, $\rho = 0.0168$, $\lambda = 0.1447$ and $\epsilon = 0.0023$. As in [8], EVeP hyperparameters were set to $\sigma = 0.3656$, $\delta = 6$, $N^* = 2$ and $\rho = 0.0142$.

- **Helicopter UAV Streaming Data**

The Helicopter UAV Streaming dataset [55] consists of streaming data from an Align Trex450 Pro Direct Flight Control helicopter made in Taiwan. It contains 6000 data points collected in different conditions to simulate nonstationarity. Equation (100) defines the regression model.

$$\hat{y}(k+1) = f(y(k), u(k)) \tag{100}$$

EVeP user-defined parameters were set to $\sigma = 0.1771$, $\delta = 70$, $N^* = 11$ and $\rho = 2.62 \times 10^{-2}$. As in [8] we used the initial 3600 points for tuning the hyperparameters, the

remaining are used for online testing. For our OCO-based EVeP, they were set to $\sigma = 0.1461$, $\delta = 58$, $N^* = 20$, $\rho = 0.0440$, $\lambda = 0.0316$ and $\epsilon = 0.00006$.

- **Box-Jenkins Gas Furnace System Identification**

This benchmark is composed of 290 data points forming the classical problem of Box-Jenkins gas prediction [56]. Aiming to infer the CO_2 concentration based on past values and the gas flow rate in a furnace, according to (101), we select the first 200 points for hyperparameters tuning and the last 90 points for online testing as in [8].

$$\hat{y}_k = f(y_{k-1}, u_{k-4}) \tag{101}$$

EVeP user-defined parameters were set to $\sigma = 0.092$, $\delta = 36$, $N^* = 4$ and $\rho = 2.009$ as in [8]. After tuning the hyperparameters for our proposal, they were set to $\sigma = 0.1404$, $\delta = 18$, $N^* = 2$, $\rho = 0.0185$, $\lambda = 0.0014$ and $\epsilon = 0.0053$.

- **Weather Temperature Prediction**

This problem was first proposed by [27] and studied in [28]. It aims to predict one step ahead monthly temperatures based on weather time series from Death Valley, Ottawa and Lisbon from, respectively, January 1901, 1895, and 1910 up to December 2009. The input is set to be the past 12 points for each prediction instance as in [28]. And, to be comparable to the other contenders in literature, we used the same data for tuning the hyperparameters and testing the proposed method.

After applying the optimization procedure within the intervals $[0, 0.3]$ for σ, $[1, 100]$ for δ, $[1, 50]$ for N^*, $[10^{-2}, 10^3]$ for ρ and the previously cited for λ and ϵ, EVeP hyperparameters were set to $\sigma = 0.3000$, $\delta = 48$, $N^* = 12$ and $\rho = 1.000$, and EVeP_OCO hyperparameters were set to $\sigma = 0.0644$, $\delta = 55$, $N^* = 19$, $\rho = 0.0483$, $\lambda = 0.02149$ and $\epsilon = 0.00006$.

6.4 Results and Discussion

6.4.1 Nonlinear Dynamic Plant Identification With Time-Varying Characteristics

We present, in Table 5, the metrics for our model and its comparison to other contenders in literature. As can be seen, there is a significant loss of performance, but our proposal still surpass most of the other methods, conquering the third place of the table. As for its computational cost, our online MTL version has demonstrated to be almost 30 times faster, which shall represent an important compromise depending on the application.

6.4.2 Helicopter UAV Streaming Data

Table 6 shows our results and the comparison with other contenders in the literature. The RMSE and NDEI increment of only 7% against EVeP demonstrates that our online MTL method was able to perform well at a computational cost 2.1 times lower on average, losing only for its batch-fashioned relative.

Table 5: Nonlinear Dynamic Plant Identification With Time-Varying Characteristics

Evolving system	No. of Rules (Avg)	RMSE	Time (s)	
			Avg	Stdev (Max)
EVeP	1 (1.0260)	0.0305	0.0495	0.0326 (0.2118)
SEFS [57]	6 (3.5610)	0.0428		
EVeP_OCO	1 (1.2003)	0.0532	0.0015	0.0011 (0.0117)
FBeM_MTL [28]	10 (3.6927)	0.0645		
GSETSK [46, 57]	11	0.0661		
eTS [57, 58]	77	0.0682		
FBeM [27]	10 (3.6927)	0.0711		
DENFIS [57, 59]	105	0.1749		

Table 6: Helicopter UAV Streaming Data

Evolving system	No. of Rules (Avg)	RMSE	NDEI	Time (s)	
				Avg	Stdev (Max)
EVeP	7 (6.4915)	0.0277	0.4323	0.0125	0.0158 (0.2298)
EVeP_OCO	4 (4.9349)	0.0296	0.4615	0.0058	0.0015 (0.0150)
EFS-SLAT [60]	11 (4.2770)	0.0305	0.4758		
Type-1 PALM(G) [55]	11	0.0313	0.4886		
GENEFIS [55, 61]	2	0.0355	0.5541		
PANFIS [55, 62]	9	0.0362	0.5652		
Type-1 PALM(L) [55]	6	0.0363	0.5668		
Simpl_eTS [55, 63]	3	0.0534	0.8336		
eTS [55, 58]	3	0.0535	0.8352		

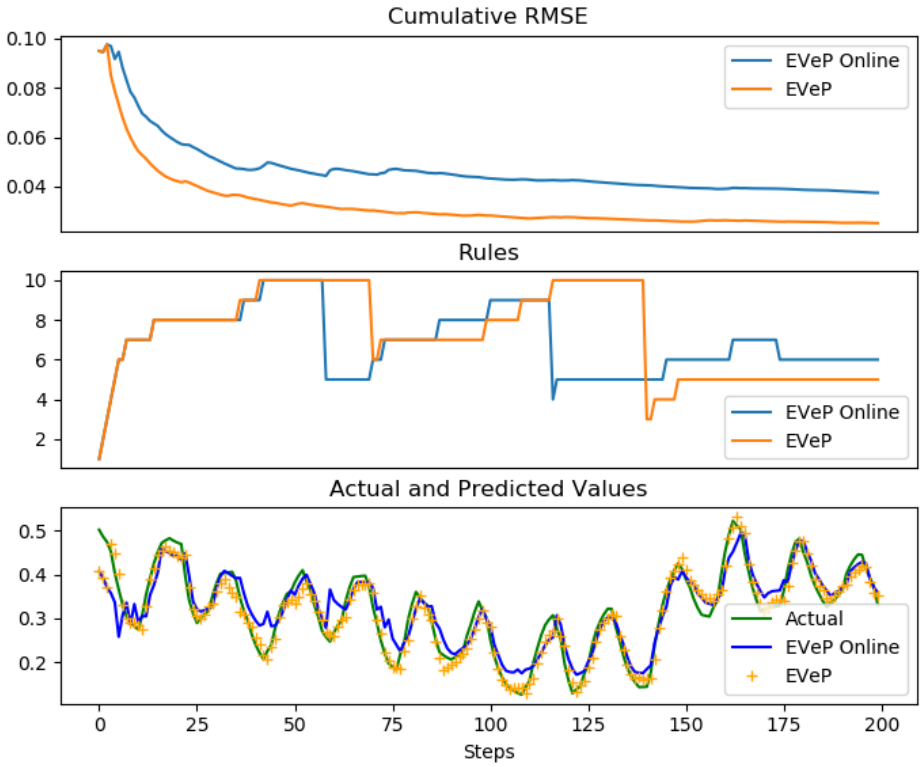


Figure 9: Cumulative RMSE, rules and predictions for the Helicopter UAV Stream experiment.

It is presented in Fig. 9 the cumulative RMSE, number of rules and predictions for EVeP and EVeP_OCO for the first 200 steps. At the very beginning, both methods follow similar patterns for the number of rules and cumulative RMSE. Then the cumulative RMSE

curves separate from each other as EVeP starts to get more accurate. Since the hyperparameters are considerably different from each other, it becomes more complicated to compare the batch and online performances apart from the effect of the evolving rules. It is, however, expected for online learning methods to present a performance gap from batch models as the regret tends to be minimized but continues to be higher than zero.

6.4.3 Box-Jenkins Gas Furnace System Identification

The outcome is presented in Table 7. We note that our proposal increased the RMSE by 45% while reducing the mean processing time to 11% of the EVeP execution time. The reduction in the maximal time (worst case along iterations) is still more significant (3.6% of the EVeP maximum value). It also occupies the third place among all the contenders, offering a consistent trade-off between accuracy and cost.

Table 7: Box-Jenkins Gas Furnace System Identification

Evolving system	No. of Rules (Avg)	RMSE	Time (s)	
			Avg	Stdev (Max)
EVeP	5 (4.4)	0.0085	0.0406	0.0615 (0.2475)
Stable EFS data cloud [56]	10	0.0107		
EVeP_OCO	6 (4.1)	0.0124	0.0045	0.0015 (0.0090)
AnYA (eClustering) [56, 64]	1	0.0166		
CEFNS [56, 65]	2	0.0207		
SOFMLS [56, 66]	5	0.0470		
SONFNN [56, 67]	4	0.4800		
Simpl_eTS [56, 63]	3	0.0485		
eTS [56, 58]	5	0.0490		
SAFIS [56, 68]	5	0.0710		
DENFIS [56, 59]	18	0.1774		

Fig. 10 presents the cumulative RMSE, number of rules and predictions for EVeP and EVeP_OCO for the entire test set. We can see that both methods performed similarly before the step 8, after which the EVeP_OCO presented a lagged series behaviour. It might be explained by the lack of information about values older than y_{k-1}. As the regret tend to decrease, we tend to approximate local solutions to global optimal ones. Supposing that the gas flow rate carries few information about the prediction of CO_2 concentration, it would mean that we resort only to its past value. However, the actual series displays an oscillatory pattern, which means that, when the values are increasing, the predictor parameter multiplying y_{k-1} should be greater than 1, while on a descent it would be more appropriate for it to be less than 1. The optimal solution considering all these points would, hence, try to compensate these two behaviours producing an estimator around 1. The EVeP does not suffer too much from this condition because it retrains the entire models using 4 past points at each updating step.

6.4.4 Weather Temperature Prediction

Table 8 presents our results and the other comparable candidates for each of the three temperature datasets. One can see that our proposal provided lower errors for Ottawa and Lisbon series, surpassing both EVeP (Ottawa and Lisbon) and FBeM_MTL (Ottawa), with

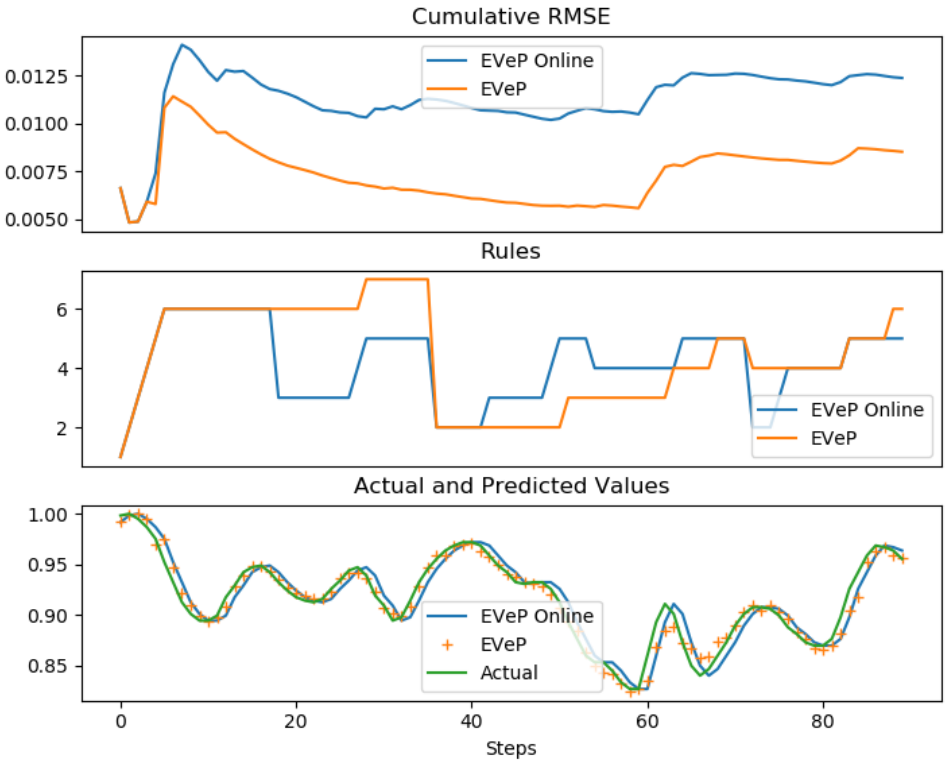


Figure 10: Cumulative RMSE, rules and predictions for the Gas Furnace experiment.

mean computational cost between 3 and 4 times lower. For Death Valley dataset, our online MTL evolving system performed a little worse (with RMSE around 10% greater than the best method), however maintaining the same level of temporal efficiency gains.

Table 8: Weather Temperature Prediction

Model	Death Valley				Ottawa				Lisbon			
			Time (s)				Time (s)				Time (s)	
	No. of Rules (Avg)	RMSE	Avg	Stdev (Max)	No. of Rules (Avg)	RMSE	Avg	Stdev (Max)	No. of Rules (Avg)	RMSE	Avg	Stdev (Max)
EVeP _OCO	4 (4.0)	0.041	0.0049	0.0013 (0.0138)	4 (4.0)	0.047	0.0048	0.0007 (0.0088)	4 (4.1)	0.051	0.0064	0.0032 (0.0239)
EVeP	7 (6.8)	0.038	0.0160	0.0274 (0.2854)	7 (7.0)	0.049	0.0171	0.0282 (0.2894)	6 (6.9)	0.053	0.0189	0.0313 (0.2903)
FBeM _MTL [28]	8	0.037			6	0.049			7	0.051		
FBeM _recursive [28]	8	0.046			6	0.054			7	0.058		
SWMA [27]		0.083				0.081				0.071		
xTS [27]	5	0.086			11	0.085			7	0.092		
eTS [27]	5	0.086			8	0.084			7	0.094		
DENFIS [27]	13	0.068			23	0.086			27	0.094		
MLP [27]	20	0.064			20	0.084			20	0.108		
MA [27]		0.167				0.162				0.141		
FBeM _batch [28]	8	0.175			6	0.168			7	0.165		

7 Conclusion and Future Steps

We successfully identified and corrected an important misconception in the original formulation of an otherwise powerful kernel-based online method denoted OSLSSVR. We conceptually demonstrated the incorrectness of decisive steps of the formulation and proposed an appropriate derivation with the same $O(m_t^2), m_t << N$ complexity.

Both the original and the reformulated versions are compared in practice, resorting to an online regression benchmark. The method implemented with the original equations presented high instability leading to severe loss of accuracy, while our reformulation performed according to the expected online behavior, thus properly revealing the potential of the OSLSSVR online regression framework. Having corrected its recursive formulation, we paved the way for the development of our kernel-based online MTL proposal (MT-OSLSSVR).

As detailed before, we developed two online MTL methods, one based on recursive least squares (MT-WRLS) and the other based on recursive kernel methods (MT-OSLSSVR). These two methods were proposed together due to their equivalence to the batch-fashioned multi-task graph-based formulation, as we properly deduce that the MT-WRLS and the MT-OSLSSVR provide online solutions to the same optimization problem, either in the primal or in the dual space, respectively. Profiting from efficient recursive procedures, they achieve an optimal solution at each iteration, in the case of MT-WRLS, or approximate it with controllable precision through the sparsity parameter v in the case of MT-OSLSSVR.

The per-instance cost of our proposals ($O(d^2 \times T^2)$ and $O(d \times m_n^2)$) are competitive when compared to other methods in the literature, which are either based on OGD, with lower cost $O(d \times T)$ and lower convergence, or on more complex optimization procedures ($O(d^3 \times T)$), whose convergence is guaranteed but not immediate such as the MADMM. Therefore, our proposals present a superior compromise between computational complexity and convergence.

We experimented the MT-WRLS and the MT-OSLSSVR on a real-world time-dependent regression benchmark of wind speed forecasting and evidenced the higher-ranking performance of our proposals against online STL and MTL contenders, with up to 16% of error reduction. Our results were submitted to statistical tests, providing statistical significance to the better ranking of our proposals.

The combination of online methods with Extreme Learning Machines was also proposed to experiment the potential of nonlinear online MTL methods, a novelty in the literature of online MTL, with positive positive impact on scenarios characterized by strong nonlinearities. This topic shall be better explored in the future steps of this research, specially with nonlinear multi-task kernels.

Their dependence on a $T \times T$ matrix inversion, performed just once prior to the online learning stage, can be specially justified in applications for which the computational constraints outside the data stream processing hardware are less restricted. We shall, however, search for methods to relieve this burden as part of the future agenda of this research.

Our evolving system proposal, denoted EVeP_OCO, converts EVeP to a fully recursive method, capable of maintaining a competitive performance, notably in comparison with other contenders in the literature. The observed substantial reduction in computational cost,

while still preserving an admissible prediction accuracy, demonstrated that it is possible to achieve a fast and effective online fuzzy rule-based multi-task formulation to handle data streams.

The association of EVeP and online MTL performed from 74% worse up to 7% better than the original EVeP on real-world and synthetic benchmarks, while providing an iteration step up to 30 times faster in average. The bottlenecks, standard deviation and max time, were also reduced by at least 10 times. Our formulation also enables an efficient distributed learning of the consequent parameters. As some experiments were conducted with the same data for both tuning and testing the evolving systems in literature, we lost some generalization ability assessment in order to gain comparability.

References

[1] G. R. Lencione, A. O. C. Ayres, and F. J. Von Zuben, "Online convex optimization of a multi-task fuzzy rule-based evolving system," in *2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2022, pp. 1–8.

[2] G. R. Lencione and F. J. V. Zuben, "Reformulation of a regularized estimation framework for online sparse lssvr models," 2023. [Online]. Available: https://doi.org/10.2139/ssrn.4474505

[3] ——, "Online multi-task learning with recursive least squares and recursive kernel methods," 2024.

[4] G. R. Lencione and F. J. Von Zuben, "Wind speed forecasting via multi-task learning," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.

[5] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, vol. 2, 2004, pp. 985–990 vol.2.

[6] J. D. A. Santos and G. A. Barreto, "A regularized estimation framework for online sparse lssvr models," *Neurocomputing*, vol. 238, pp. 114–125, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231217301169

[7] I. Škrjanc, J. A. Iglesias, A. Sanchis, D. Leite, E. Lughofer, and F. Gomide, "Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: A survey," *Information Sciences*, vol. 490, pp. 344–368, 2019.

[8] A. O. C. Ayres and F. J. Von Zuben, "The extreme value evolving predictor," *IEEE Transactions on Fuzzy Systems*, pp. 1–1, 2020.

[9] J. Zhou, J. Chen, and J. Ye, *User's Manual MALSAR: Multi-tAsk Learning via StructurAl Regularization*, 2012. [Online]. Available: www.MALSAR.org

[10] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 12, pp. 5586–5609, 2022.

[11] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, 07 1997.

[12] A. Argyriou, T. Evgeniou, and M. Pontil, "Multi-task feature learning," in *Advances in Neural Information Processing Systems*, B. Schölkopf, J. Platt, and T. Hoffman, Eds., vol. 19. MIT Press, 2006. [Online]. Available: https://proceedings.neurips.cc/paper/2006/file/0afa92fc0f8a9cf051bf2961b06ac56b-Paper.pdf

[13] J. Chen, L. Tang, J. Liu, and J. Ye, "A convex formulation for learning shared structures from multiple tasks," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 137–144. [Online]. Available: https://doi.org/10.1145/1553374.1553392

[14] S. Ji and J. Ye, "An accelerated gradient method for trace norm minimization," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 457–464. [Online]. Available: https://doi.org/10.1145/1553374.1553434

[15] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Review*, vol. 38, no. 1, pp. 49–95, 1996. [Online]. Available: https://doi.org/10.1137/1038003

[16] A. Gonçalves, F. Von Zuben, and A. Banerjee, "Multi-task sparse structure learning with gaussian copula models," *Journal of Machine Learning Research*, vol. 17, pp. 1–30, 04 2016.

[17] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2019.

[18] S. C. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online learning: A comprehensive survey," *Neurocomputing*, vol. 459, pp. 249–289, 2021.

[19] E. Hazan, "Introduction to online convex optimization," *CoRR*, vol. abs/1909.05207, 2019. [Online]. Available: http://arxiv.org/abs/1909.05207

[20] O. Dekel, P. M. Long, and Y. Singer, "Online multitask learning," in *Learning Theory*, G. Lugosi and H. U. Simon, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 453–467.

[21] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile, "Linear algorithms for online multitask classification," *Journal of Machine Learning Research*, vol. 11, no. 97, pp. 2901–2934, 2010. [Online]. Available: http://jmlr.org/papers/v11/cavallanti10a.html

[22] X. Cao and K. J. R. Liu, "Decentralized sparse multitask rls over networks," *IEEE Transactions on Signal Processing*, vol. 65, no. 23, pp. 6217–6232, 2017.

[23] J. Xu, P.-N. Tan, J. Zhou, and L. Luo, "Online multi-task learning framework for ensemble forecasting," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 6, pp. 1268–1280, 2017.

[24] D. Jin, J. Chen, C. Richard, and J. Chen, "Online proximal learning over jointly sparse multitask networks with $\ell_{\infty,1}$ regularization," *IEEE Transactions on Signal Processing*, vol. 68, pp. 6319–6335, 2020.

[25] R. Nassif, S. Vlaski, C. Richard, J. Chen, and A. H. Sayed, "Multitask learning over graphs: An approach for distributed, streaming machine learning," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 14–25, 2020.

[26] P. P. Angelov and D. P. Filev, "An approach to online identification of Takagi-Sugeno fuzzy models," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 484–498, 2004.

[27] D. Leite, R. Ballini, P. Costa, and F. Gomide, "Evolving fuzzy granular modeling from nonstationary fuzzy data streams," *Evolving Systems*, vol. 3, no. 2, pp. 65–79, 2012.

[28] A. O. C. Ayres and F. J. Von Zuben, "Multitask learning applied to evolving fuzzy-rule-based predictors," *Evolving Systems*, pp. 1–16, 2019.

[29] A. O. Ayres and F. J. Von Zuben, "An improved version of the fuzzy set based evolving modeling with multitask learning," in *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2020, pp. 1–8.

[30] A. O. C. Ayres and F. J. Von Zuben, "The extreme value evolving predictor," *IEEE Transactions on Fuzzy Systems*, pp. 1–14, 2020.

[31] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, pp. 273–297, 1995.

[32] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Advances in Neural Information Processing Systems*, M. Mozer, M. Jordan, and T. Petsche, Eds., vol. 9. MIT Press, 1996.

[33] C. Saunders, A. Gammerman, and V. Vovk, "Ridge regression learning algorithm in dual variables," in *Proceedings of the Fifteenth International Conference on Machine Learning*, ser. ICML '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, p. 515–521.

[34] T. A. Davis, S. Rajamanickam, and W. M. Sid-Lakhdar, "A survey of direct methods for sparse linear systems," *Acta Numerica*, vol. 25, p. 383–566, 2016.

[35] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *Computational Learning Theory*, D. Helmbold and B. Williamson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 416–426.

[36] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, 2004.

[37] S. Van Vaerenbergh, J. Via, and I. Santamaria, "A sliding-window kernel rls algorithm and its application to nonlinear channel identification," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 5, 2006, pp. V–V.

[38] S. Van Vaerenbergh, I. Santamaría, W. Liu, and J. C. Príncipe, "Fixed-budget kernel recursive least-squares," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2010, pp. 1882–1885.

[39] Y. Tian and Y. Zhang, "A comprehensive survey on regularization strategies in machine learning," *Information Fusion*, vol. 80, pp. 146–166, 2022.

[40] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[41] G. Pillonetto, T. Chen, A. Chiuso, G. De Nicolao, and L. Ljung, *Bayesian Interpretation of Regularization*. Cham: Springer International Publishing, 2022, pp. 95–134.

[42] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996. [Online]. Available: http://www.jstor.org/stable/2346178

[43] G. R. Lencione and F. J. V. Zuben, "Online multi-task learning with recursive least squares and recursive kernel methods," 2023.

[44] D. Tylavsky and G. Sohie, "Generalization of the matrix inversion lemma," *Proceedings of the IEEE*, vol. 74, no. 7, pp. 1050–1052, 1986.

[45] L. Surhone, M. Timpledon, and S. Marseken, *Woodbury Matrix Identity*. VDM Publishing, 2010. [Online]. Available: https://books.google.com.br/books?id=v8iZcQAACAAJ

[46] N. N. Nguyen, W. J. Zhou, and C. Quek, "Gsetsk: a generic self-evolving tsk fuzzy neural network with a novel hebbian-based rule reduction approach," *Applied Soft Computing*, vol. 35, pp. 29–42, 2015.

[47] G. R. Lencione, A. O. C. Ayres, and F. J. Von Zuben, "Online convex optimization of a multi-task fuzzy rule-based evolving system," in *2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2022, pp. 1–8.

[48] R. Horn and C. Johnson, *Matrix Analysis*. Cambridge University Press, 2012.

[49] C. Draxl, A. Clifton, B.-M. Hodge, and J. McCaa, "The wind integration national dataset (wind) toolkit," *Applied Energy*, vol. 151, pp. 355–366, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261915004237

[50] D. Pereira, A. Afonso, and F. Medeiros, "Overview of friedman's test and post-hoc analysis," Aug 2015. [Online]. Available: http://hdl.handle.net/10174/16113

[51] S. Coles, J. Bawa, L. Trenner, and P. Dorazio, *An introduction to statistical modeling of extreme values*. Springer, 2001, vol. 208.

[52] S. Van Vaerenbergh, J. Via, and I. Santamaría, "A sliding-window kernel RLS algorithm and its application to nonlinear channel identification," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 5. IEEE, 2006, pp. 789–792.

[53] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. 61, pp. 2121–2159, 2011. [Online]. Available: http://jmlr.org/papers/v12/duchi11a.html

[54] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[55] M. M. Ferdaus, M. Pratama, S. G. Anavatti, and M. A. Garratt, "Palm: An incremental construction of hyperplanes for data stream regression," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 11, pp. 2115–2129, 2019.

[56] H.-J. Rong, P. P. Angelov, X. Gu, and J.-M. Bai, "Stability of evolving fuzzy systems based on data clouds," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 5, pp. 2774–2784, 2018.

[57] D. Ge and X.-J. Zeng, "A self-evolving fuzzy system which learns dynamic threshold parameter by itself," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 8, pp. 1625–1637, 2019.

[58] P. P. Angelov and D. P. Filev, "An approach to online identification of takagi-sugeno fuzzy models," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 484–498, 2004.

[59] N. K. Kasabov and Q. Song, "Denfis: dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 144–154, 2002.

[60] D. Ge and X.-J. Zeng, "Learning data streams online—an evolving fuzzy system approach with self-learning/adaptive thresholds," *Information Sciences*, vol. 507, pp. 172–184, 2020.

[61] M. Pratama, S. G. Anavatti, and E. Lughofer, "Genefis: toward an effective localist network," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 3, pp. 547–562, 2013.

[62] M. Pratama, S. G. Anavatti, P. P. Angelov, and E. Lughofer, "Panfis: A novel incremental learning machine," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 55–68, 2013.

[63] P. Angelov and D. Filev, "Simpl_ets: A simplified method for learning evolving takagi-sugeno fuzzy models," in *The 14th IEEE International Conference on Fuzzy Systems, FUZZ'05.* IEEE, 2005, pp. 1068–1073.

[64] P. Angelov and R. Yager, "A new type of simplified fuzzy rule-based system," *International Journal of General Systems*, vol. 41, no. 2, pp. 163–185, 2012.

[65] R.-J. Bao, H.-J. Rong, P. P. Angelov, B. Chen, and P. K. Wong, "Correntropy-based evolving fuzzy neural system," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 3, pp. 1324–1338, 2017.

[66] J. de Jesús Rubio, "Sofmls: online self-organizing fuzzy modified least-squares network," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 6, pp. 1296–1309, 2009.

[67] G. Leng, T. M. McGinnity, and G. Prasad, "An approach for on-line extraction of fuzzy rules using a self-organising fuzzy neural network," *Fuzzy Sets and Systems*, vol. 150, no. 2, pp. 211–243, 2005.

[68] H.-J. Rong, N. Sundararajan, G.-B. Huang, and P. Saratchandran, "Sequential adaptive fuzzy inference system (safis) for nonlinear system identification and prediction," *Fuzzy Sets and Systems*, vol. 157, no. 9, pp. 1260–1275, 2006.