Sevilla-Salcedo, Carlos; Gallardo-Antolín, Ascensión; Gómez-Verdejo, Vanessa; Parrado-Hernández, Emilio

Bayesian learning of feature spaces for multitask regression

Published in:
Neural Networks

DOI:
10.1016/j.neunet.2024.106619

Published: 01/11/2024

Document Version
Publisher's PDF, also known as Version of record

Published under the following license:
CC BY-NC

Please cite the original version:
Sevilla-Salcedo, C., Gallardo-Antolín, A., Gómez-Verdejo, V., & Parrado-Hernández, E. (2024). Bayesian learning of feature spaces for multitask regression. *Neural Networks*, *179*, 1-16. Article 106619. https://doi.org/10.1016/j.neunet.2024.106619

Neural Networks 179 (2024) 106619

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Full Length Article

Bayesian learning of feature spaces for multitask regression

Carlos Sevilla-Salcedo [a,b,*], Ascensión Gallardo-Antolín [a], Vanessa Gómez-Verdejo [a], Emilio Parrado-Hernández [a]

[a] *Department of Signal Theory and Communications, Universidad Carlos III de Madrid, Leganés, 28911, Madrid, Spain*
[b] *Department of Computer Science, Aalto University, Espoo, 02150, Helsinki, Finland*

A R T I C L E   I N F O

Keywords:
Kernel methods
Random fourier features
Bayesian regression
Multitask regression
Extreme learning machine
Random vector functional link networks

A B S T R A C T

This paper introduces a novel approach to learn multi-task regression models with constrained architecture complexity. The proposed model, named RFF-BLR, consists of a randomised feedforward neural network with two fundamental characteristics: a single hidden layer whose units implement the random Fourier features that approximate an RBF kernel, and a Bayesian formulation that optimises the weights connecting the hidden and output layers. The RFF-based hidden layer inherits the robustness of kernel methods. The Bayesian formulation enables promoting multioutput sparsity: all tasks interplay during the optimisation to select a compact subset of the hidden layer units that serve as common non-linear mapping for every tasks. The experimental results show that the RFF-BLR framework can lead to significant performance improvements compared to the state-of-the-art methods in multitask nonlinear regression, especially in small-sized training dataset scenarios.

1. Introduction

This paper focused on MultiTask Regression (MTR) problems with two particular restrictions: small-sized training sets and constraints on the complexity of the model architecture. The demand for this type of models naturally arises in application domains subject to strong regulation, such as health or finances (Ketu & Mishra, 2021; Xiong, Bao, & Hu, 2014). Multitasking (Caruana, 1997) also appears naturally in these scenarios. Consider, for instance, the case of a clinical study focused on the characterisation of a disease. A common setup starts with a cohort of a few tenths (hundreds if we are lucky) of patients going through the same data acquisition process. Subsequently, these data can be used in the construction of machine learning models for the prediction of scores that can help capture patterns that characterise the progression of the disease, the probability of a successful response to particular treatment, etc. The interplay among those closely related tasks suggests that fitting all these models under the same joint optimisation can be more beneficial than fitting each model with a separate independent optimisation. In fact, several recent works (Spyromitros-Xioufis, Tsoumakas, Groves, & Vlahavas, 2016; Zhen, Yu, He, & Li, 2018; Zhen, Yu, Zheng, et al., 2018) show that the MTR paradigm can lead to significant improvements in performance achieved by learning an independent model per task. The outperformance of MTR is especially noticeable in cases with small training sets that demand non-linear modelling. These results have recently motivated a growing interest in MTR algorithms and in their successful application in numerous real-world problems. Among these applications, it is worth mentioning commerce and finance (*e.g.*, prediction of stock price Li et al., 2019), environmental modelling (*e.g.*, air quality Masmoudi, Elghazel, Taieb, Yazar, & Kallel, 2020 and weather forecast Li et al., 2019), robotics (Li et al., 2019), computer vision (Emambakhsh, Bay, & Vazquez, 2019; Farlessyost, Grant, Davis, Feil-Seifer, & Hand, 2021; Tan, Chen, Ji, & Geng, 2022), speech-related tasks (*e.g.*, speech enhancement Tu, Du, Gao, & Lee, 2020 and speech intelligibility prediction Ry et al., 2022), and biomedical applications (*e.g.*, medical image analysis Ma, Kundu, & for the Alzheimer's Disease Neuroimaging Initiative, 2024; Zhen et al., 2017).

The constrained model complexity feature points towards single-hidden-layer Neural Network (NN) as a strong candidate for core technology in these scenarios. Examples of such architectures are the Radial Basis Function (RBF) NNs (Hartman, Keeler, & Kowalski, 1990), Two Layer Perceptrons (TLP) (Rumelhart, Hinton, & Williams, 1985), and randomised feedforward networks such as the Extreme Learning Machine (ELM) (Huang et al., 2006; Huang, Zhu, & Siew, 2004) and the Random Vector Functional Link Network (RVFLN) (Pao & Takefuji, 1992). Moreover, broadly used Kernel Methods (KMs) such as the Support Vector Regressor (SVR) (Smola & Schölkopf, 2004), the Kernel Ridge Regressor (KRR) (Vovk, 2013) or Gaussian Processes (GPs) (Rasmussen, 2003) can also be regarded as NNs with a single hidden layer (Burges, 1998). The usual approach to solve MTR problems with these architectures involves combining various tasks within a layer

* Corresponding author at: Department of Signal Theory and Communications, Universidad Carlos III de Madrid, Leganés, 28911, Madrid, Spain.
E-mail address: carlos.sevillasalcedo@aalto.fi (C. Sevilla-Salcedo).

https://doi.org/10.1016/j.neunet.2024.106619
Received 3 November 2023; Received in revised form 23 July 2024; Accepted 6 August 2024
Available online 13 August 2024

of common latent variables and the use of KMs (Zhen, Yu, He, & Li, 2018; Zhen, Yu, Zheng, et al., 2018) or ELMs (da Silva, Inaba, Salles, & Ciarelli, 2020) to learn a non-linear mapping between the inputs features and these latent variables. These works have reported more accurate models than other approaches such as ensembles of regression trees (Spyromitros-Xioufis et al., 2016), or the use of gene expression programming to jointly learn a non-linear model for each task (Moyano, Reyes, Fardoun, & Ventura, 2021).

One of the most important pros in favour of RVFLN to implement these models is their universal approximator feature, despite their relative mathematical simplicity (Huang, Bai, Kasun, & Vong, 2015). The standard ELM consists of a single hidden layer where each neuron is connected to the input layer by a weight vector whose values are selected at random during the network instantiation. Moreover, the user is also free to select the non-linear activation of each neuron in the hidden layer. Once the network is created, an optimisation sets the values of the weights that connect the hidden layer to the output layer. This optimisation is significantly cheaper (a simple ridge regression) compared to those needed to fit more complex architectures. The RVFLN can be regarded as an ELM completed with functional links: a set of weights that connect the input and output layers. These functional links are jointly optimised with the weights connecting the hidden and the output layers.

However, these characteristics stand as double-edged weapons in the cases of interest in this research. This high degree of flexibility to design the hidden layer (number of neurons, activation function, and probability distributions that guide the choosing of the random weights) significantly increases the risk of overfitting in MTR with small datasets, as the complexity of the non-linear mapping that connects the input and hidden layers can be difficult to control.

One way to control the size of the hidden layer and help alleviate overfitting could be the incorporation of elastic net penalties into the optimisation of the output weights, as discussed in Martínez-Martínez et al. (2011). Moreover, ELMs combined with structured regularisation techniques, such as group lasso (Yuan & Lin, 2006), have been applied to multitask scenarios (da Silva et al., 2020; Inaba, Teatini Salles, Perron, & Caporossi, 2018). Nevertheless, these approaches typically involve a large number of hyperparameters to be tuned, often through a resource-intensive cross-validation procedure. Furthermore, establishing a meaningful connection with domain-specific prior knowledge can be challenging in certain types of scenario.

Another means of alleviating overfitting within the randomised feedforward NNs (RFNNs) family is the Sparse Bayesian ELM (Luo, Vong, & Wong, 2013). This Bayesian formulation enables the introduction of prior knowledge in the architecture design (Soria-Olivas et al., 2011), with an optimisation that controls overfitting by increasing the sparsity in the output layer. This sparsity serves as a regularisation by nullifying the effect of irrelevant nodes in the hidden layer. This way, the network prediction will in fact be a function of a smaller subset of neurons in the hidden layer. However, to the best of our knowledge, the Bayesian ELM and its sparse version are only formulated for single-target regression.

An efficient extension to the multitasking case requires a prior that enforces a joint sparsity over the different task-related sets of parameters, resulting in all tasks sharing the same hidden layer (with a different output layer per task) and leading to an effective transfer of knowledge among tasks. This idea underlies MTR models such as group lasso (Yuan & Lin, 2006), dirty models (Jalali, Sanghavi, Ruan, & Ravikumar, 2010), or multilevel lasso (Lozano & Swirszcz, 2012). In this context, Bayesian formulations offer notable advantages. Bayesian methods excel in achieving robust generalisation, particularly in scenarios with limited data samples. Additionally, they provide a predictive distribution that not only informs about the model's confidence but also helps handling uncertainty. Moreover, recent research shows that by defining appropriate priors, Bayesian approaches facilitate the transfer of knowledge between tasks, as demonstrated in recent research (Goncalves et al., 2019). However, this work primarily relies on linear mappings from the shared input space to task-dependent output spaces. While this approach offers interpretability, it may limit the model's ability to capture complex non-linear relationships inherent in the data.

Focusing on the two other aspects, the selection of the activation functions and of the random weights connecting the input and hidden layer, this paper proposes to use an interesting equivalence between KMs and ELMs or RVFLNs through RFFs (Rahimi & Recht, 2007). We show that an ELM is in fact an approximation to a KM endowed with an RBF kernel if these two characteristics are met:

- its hidden layer is formed by neurons implementing a cosine activation function,
- the weights connecting the input and hidden layer (that act as frequencies in the cosine activations) are selected at random following the Fourier Transform of the RBF kernel as detailed in Section 2.2.3.

The quality of this approximation converges exponentially with the size of the hidden layer. Moreover, this observation is extended to the RVFLNs approximating a KM endowed with a composite kernel resulting from the addition of an RBF kernel with a linear one.

The method proposed in this paper, termed RFF-BLR, builds on this connection between KMs and ELMs (or RVFLNs) to combine the advantages of these two technologies with those of the Sparse Bayesian ELM in the same framework targeted at MTR with small datasets. The RFF-BLR architecture is designed as an ELM (or RVFLN) whose hidden layer contains RFFs that approximate an RBF kernel with a certain lengthscale γ. The initial value of γ can capture valuable knowledge to draw an initial coarse resolution model that depends on a single parameter. Notice that the fact that this architecture approximates a KM somehow endows it with their robustness. Then, the weights of the output layer are optimised under a Sparse Bayesian formulation. This formulation introduces a shared-among-tasks prior that encourages multioutput sparsity: the relevance of each neuron in the hidden layer (that determines if this neuron is not pruned during the optimisation and therefore forms part of the final architecture) depends on all the tasks. Moreover, the Bayesian formulation can be easily extended to cover the value of γ, avoiding the need for cross-validation.

The main contributions of the paper can be summarised as follows:

- A formulation that establishes the connection between a KM endowed with a RBF kernel and an equivalent ELM by recognising their link through the RFF approximation of the kernel. This connection is extended to a RVFLN approximating a KM endowed with a composite kernel resulting from the addition of a RBF kernel and a linear one.
- A model for multi-target regression where a single and efficient non-linear mapping is jointly learned across all tasks via Bayesian optimisation.
- A probabilistic framework for automatically learning the RBF kernel lengthscale without the necessity of cross-validation procedures.
- An empirical evaluation of the advantages of this proposal in several small-sample MTR benchmarks.

2. Related work and background

This part of the paper starts with a general overview of the works in multitask and multiview machine learning that are somehow related with our proposal. The second part of this section focuses on a more detailed review of the proposed method building blocks: ELMs, RVFL networks, KMs, and RFFs.

2.1. Related work

Besides the works already mentioned in the introduction, there are several approaches in the literature that deal with multitask problems (Borchani, Varando, Bielza, & Larranaga, 2015). The most common methods are based on linear models that exploit the correlation between the output tasks by means of regularisation. This is the case of the aforementioned group lasso model (Yuan & Lin, 2006), dirty models (Jalali et al., 2010), multilevel lasso (Lozano & Swirszcz, 2012), Multiple output Regression with Output and Task Structures (MROTS) (Rai, Kumar, & Daume, 2012), the Multivariate Regression with Covariance Estimation (MRCE) (Rothman, Levina, & Zhu, 2010), that jointly learns the model weights and the task correlations, or the work of Liu, Wang, and Zhao (2015), where additional regularisation parameters are included to deal independently with the noise in each task.

Other methods rely on the building of ensemble architectures to improve multi-output regression performance, such as the FItted Rule Ensembles (FIREs) (Aho, Ženko, Džzeroski, Elomaa, & Brodley, 2012), the multi-objective random forests (Kocev, Vens, Struyf, & Džeroski, 2007), or the work of Moyano et al. (2021) which combines the ensemble architecture with gene expression programming.

Conversely, other works propose to simplify the modelling of the output space relationships employing a cluster structure. In this approach, each cluster of tasks is modelled separately, with tasks in the same cluster sharing similar weight vectors. Here, Clustered Multi-Target Learning (CMTL) (Jacob, Vert, & Bach, 2008) and Flexible Clustered Multi-Target (FCMTL) (Zhou & Zhao, 2015) stand out, where the cluster structure is learnt by identifying representative tasks. On the other hand, (Khan, Hu, Li, Diallo, & Du, 2023) proposes a method for jointly learning representations from multiple views using sparse and low-rank structures, enhancing clustering performance. Similarly, Multi-view Clustering for Multiple Manifold Learning via Concept Factorisation (Khan, Hu, Li, Diallo, & Wang, 2023) introduces a framework that learns multiple manifolds across different views by factorising the concept representation matrices, facilitating clustering tasks in multi-view scenarios.

However, the kernel-based multitask methods, capable of exploiting non-linear relationships between data and/or between tasks, have shown the best performance. This is the case for algorithms such as (Brouard, Szafranski, & d'Alché-Buc, 2016), where the authors use kernels to learn the non-linear structure of the observations, or (Dinuzzo, Ong, Pillonetto, & Gehler, 2011), where the kernel is used in the output. Within this framework, the Multi-layer Multi-target Regression (MMR) method delivers very competitive results in terms of accuracy (Zhen, Yu, He, & Li, 2018; Zhen, Yu, Zheng, et al., 2018). In these works the data are mapped into a reproducing kernel Hilbert space where a common feature representation and inter-target correlation are learnt.

Another set of techniques that can be used for multitasking problems are models based on deep learning. Under this paradigm, non-linear relationships can be easily modelled and can be efficiently applied to areas involving image or time series processing (Ruder, 2017). However, they do not meet any of the requirements of the scenarios we want to work with, as deep learning models demand a large amount of data to avoid overfitting, require long training time, and yield networks that are difficult to interpret.

2.2. Review of the building blocks of the proposal

This section focuses on a review from the point of view of regression of the main techniques (ELMS, RVLFNs, KMs and RFFs) that are required for the understanding of our method. For the sake of simplicity, this section considers a generic single-task regression problem. Section 4 details how these results are extended to the MTR setting, which is the main focus of this research.

The goal of a single task regression model is to find a mapping $f(\cdot)$ from observations in a certain input space $\mathbf{x} \in \mathbb{R}^D$ to real valued targets $y \in \mathbb{R}$. This mapping is learnt from a given training dataset formed by N observations with their corresponding targets $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$. Let us denote by $\mathbf{X}$ the $N \times D$ matrix whose rows contain the training observations and by $\mathbf{y}$ the N-dimensional vector resulting from the concatenation of the corresponding targets. For convenience, $\mathbf{x}_{n,:}$ denotes the nth row of matrix $\mathbf{X}$ (that corresponds to the nth training observation).

2.2.1. Review of extreme learning machines and random vector functional link networks

ELMs and RVFLs are two instances of a machine learning paradigm known as randomised feedforward NNs (Suganthan & Katuwal, 2021). This approach advocates for the random selection of all weights and parameters in hidden layers, while focusing optimisation efforts, often through ridge regression, solely on the connections connecting the last hidden layer with the output layer.

A typical ELM is characterised by a single hidden layer comprising M neurons, each implementing an activation function $\phi_m(\mathbf{x})$ that relies on a set of randomly chosen parameters during the initialisation of the network, which remains unoptimised. Non-linear activation functions frequently employed in ELMs include sigmoid, trigonometric, RBFs, or polynomial functions, as extensively discussed in (Huang, 2015).

Once all the M activations of the ELM hidden layer have been fixed, the mapping that predicts the target y for a given input observation $\mathbf{x}$ presents the following structure[1]:

$$f_{\text{ELM}}(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}_{\text{ELM}}(\mathbf{x}), \tag{1}$$

where the weight vector $\mathbf{w}$ results from the concatenation of the M weights, w_m, that connect each neuron in the hidden layer with the output layer, and feature vector $\boldsymbol{\phi}_{\text{ELM}}(\mathbf{x})$ results from the concatenation of the M activation functions, $\phi_m(\mathbf{x})$, $m = 1, \dots, M$.

The ELM fitting process involves an optimisation procedure that determines the value of coefficients w_m. A prevalent method for this optimisation is ridge regression, expressed as:

$$\min_{\mathbf{w}} \frac{1}{2}\|\boldsymbol{\Phi}_{\text{ELM}}\mathbf{w} - \mathbf{y}\|^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2, \tag{2}$$

where $\boldsymbol{\Phi}_{\text{ELM}}$ denotes the $N \times M$ matrix formed by the M activation functions evaluated across the N training samples: $(\boldsymbol{\Phi}_{\text{ELM}})_{n,m} = \phi_m(\mathbf{x}_n)$.

On the other hand, the RVFLN can be seen as an expansion of the ELM framework by incorporating direct connections, termed functional links, that link the input layer with the output layer. Consequently, the mapping function established by the RVFLN is expressed as:

$$f_{\text{RVFL}}(\mathbf{x}) = \sum_{m=1}^{M} w_m \phi_m(\mathbf{x}) + \sum_{m=M+1}^{M+D} w_m x_{m-M} \tag{3}$$

where x_d, $d = 1, dots, D$, denotes the dth feature of a generic input observation $\mathbf{x}$.

The optimisation that would determine the weight values w_m, $m = 1, \dots, M + D$ can also be carried out with ridge regression, introducing the functional links in the optimisation of Eq. (2):

$$\min_{\mathbf{w}} \frac{1}{2}\|\boldsymbol{\Phi}_{\text{RVFL}}\mathbf{w} - \mathbf{y}\|^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{4}$$

where $\boldsymbol{\Phi}_{\text{RVFL}} = [\boldsymbol{\Phi}_{\text{ELM}}\mathbf{X}]$ is the $N \times (M + D)$ matrix defined as the concatenation of $\boldsymbol{\Phi}_{\text{ELM}}$ and $\mathbf{X}$. Notice that weight vector $\mathbf{w}$ in Eq. (4) has $M + D$ components.

In the recent literature one can find a number of extensions of these basic ELM and RVFL networks with an augmented expressive

[1] This model can be extended by the introduction of a bias term b to yield $f_{\text{ELM}}(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}_{\text{ELM}}(\mathbf{x}) + b$. This can be easily handled in the optimisation by adding a column of 1s to the right of $\boldsymbol{\Phi}_{\text{ELM}}$ and b as the last element of $\mathbf{w}$. For the sake of simplicity we consider this extension could be included into all the models described in the remainder of this section.

power that enables the development of accurate complex models in these situations where the size of the training set invites crossing the shallow-to-deep learning threshold. For instance, in Huang et al. (2015) the concept of hidden neurons is extended to architectures where the units in the hidden layers become sub-networks. In the RFVL case, those extensions include Ensemble deep RVFL (edRVFL) (Shi, Katuwal, Suganthan, & Tanveer, 2021), which proposes an RVFLN with several hidden layers that can be further combined in ensembles, as well as other variants like (Gao, Li, Hu, Suganthan, & Yuen, 2023; He et al., 2023; Li et al., 2023; Sajid, Tanveer, & Suganthan, 2024) that use this technique in fuzzy networks, electroencephalogram-based recognition, online learning, and self-supervised learning, respectively. In addition, the Neuro-Fuzzy RVFL (NFRVFL) network (Sajid, Malik, Tanveer, & Suganthan, 2024) incorporates the principles of fuzzy logic to better deal with the uncertainty inherent in the datasets. The reader is referred to Malik, Gao, Ganaie, Tanveer, and Suganthan (2023) for a complete survey of RVFLNs and their extensions and their applications.

2.2.2. *Review of kernel methods for regression*

Kernels are symmetric and positive semidefinite real-valued functions with two arguments. Every kernel, denoted $\kappa(\mathbf{x}_i, \mathbf{x}_j)$, is connected with a lifting $\boldsymbol{h}(\cdot)$ of the input space into a particular feature space such that the evaluation of the kernel between two observations in the input space is equivalent to computing the inner product between $\boldsymbol{h}(\mathbf{x}_i)$ and $\boldsymbol{h}(\mathbf{x}_j)$ (Schölkopf & Smola, 2002):

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{h}(\mathbf{x}_i), \boldsymbol{h}(\mathbf{x}_j) \rangle. \tag{5}$$

Common examples of kernel functions are linear, RBF, or polynomial. Moreover, the sum of two given kernel functions $\kappa_1(\mathbf{x}_i, \mathbf{x}_j)$ with lifting $\boldsymbol{h}_1(\mathbf{x})$, and $\kappa_2(\mathbf{x}_i, \mathbf{x}_j)$ with lifting $\boldsymbol{h}_2(\mathbf{x})$ becomes a valid kernel (Schölkopf & Smola, 2002):

$$\kappa_+(\mathbf{x}_i, \mathbf{x}_j) = \kappa_1(\mathbf{x}_i, \mathbf{x}_j) + \kappa_2(\mathbf{x}_i, \mathbf{x}_j) \tag{6}$$

with a lifting $\boldsymbol{h}_+(\mathbf{x})$ that results from the concatenation of the liftings of the addends: $h_+(\mathbf{x}) = [h_1(\mathbf{x})\ h_2(\mathbf{x})]$. KMs can be used to obtain robust non-linear versions of linear models as long as the input observations appear combined exclusively in terms of dot products in the model formulation (Schölkopf & Smola, 2002). That being the case, one just needs to apply the *kernel trick*: replace each scalar product by the kernel of the corresponding observations to arrive at the non-linear version. For example, given a specific kernel function $\kappa(\cdot,\cdot)$, a kernelized regression model would formulate a mapping described by the expression:

$$f(\mathbf{x}) = \boldsymbol{\kappa}(\mathbf{x})^\top \boldsymbol{\beta} + b, \tag{7}$$

where $\boldsymbol{\kappa}(\mathbf{x}) = \left[\kappa(\mathbf{x}_1, \mathbf{x}), \dots, \kappa(\mathbf{x}_N, \mathbf{x})\right]^\top$ is the vector whose components are the evaluation of the kernel between $\mathbf{x}$ and the N training observations, and $\boldsymbol{\beta} = \left[\beta_1, \dots, \beta_N\right]^\top$ and b are the model parameters that are determined during an optimisation. This optimisation usually involves minimising a loss function evaluated in the training set, $\mathcal{L}(f(\mathbf{x}), y)$ plus a regularisation term $\Omega(\boldsymbol{\beta})$ that helps prevent overfitting:

$$\min_{\boldsymbol{\beta}, b} \sum_{n=1}^{N} \mathcal{L}(f(\mathbf{x}_n), y_n) + \lambda \Omega(\boldsymbol{\beta}). \tag{8}$$

For instance, in kernel ridge regression (Vovk, 2013) the loss function is the square error $\mathcal{L}(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$ and the regularisation penalty is $\Omega(\boldsymbol{\beta}) = 1/2\boldsymbol{\beta}^\top K \boldsymbol{\beta}$, where K is the $N \times N$ kernel matrix constructed with all pairwise kernel evaluations involving the training observations, *i.e.*, $K_{i,j} = \kappa(\mathbf{x}_{i,:}, \mathbf{x}_{j,:})$, $i, j = 1, \dots, N$.

If we focus on the ubiquitous RBF kernel, given by:

$$\kappa_{\text{RBF}}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right), \tag{9}$$

one can realise that a model with the structure of Eq. (7) and this kernel can be implemented as a single layer RBF NN. This involves particularising the model in Eq. (1) to:

- $M = N$ neurons in the hidden layer, one neuron per each observation in the training set.
- The activation function of the mth neuron in the hidden layer $\phi_m(\cdot)$ is a RBF activation function centred in the mth training observation $\mathbf{x}_m$:

$$\phi_m(\mathbf{x}) = \exp\left(-\gamma \|\mathbf{x}_m - \mathbf{x}\|^2\right)$$

- Coefficients $\{\beta_n\}_{n=1}^M$ optimised with (8) are used as the weights that connect the hidden layer to the output layer.
- The kernel hyperparameter γ is used as a spread parameter in all activation functions of the neurons in the hidden layer.

2.2.3. *Review of random fourier features to approximate RBF kernels*

RFFs are introduced in (Rahimi & Recht, 2007) as an alternative to construct approximations to RBF kernel machines in cases where the number of observations is so large that the optimisation needed to fit the model (which depends on the size of the kernel matrix) becomes extremely costly. The RFFs can be regarded as a deconstruction of the kernel trick. In other words, instead of using kernels and solving an optimisation with one variable per training observation, the RFFs approximate the lifting $\boldsymbol{h}(\cdot)$ associated with the kernel by a lower-dimensional mapping $\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x})$ (notice that for some kernels the corresponding $\boldsymbol{h}(\cdot)$ involves lifting into a feature space of infinite dimensions) so that

$$\kappa_{\text{RBF}}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right) = \langle \boldsymbol{h}(\mathbf{x}_i), \boldsymbol{h}(\mathbf{x}_j) \rangle \approx \boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}_i)^\top \boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}_j). \tag{10}$$

This way, the kernel machine is approximated by a linear model with features given by the components of $\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x})$. Such approximation can be fitted with an optimisation of reduced computational burden (Shalev-Shwartz, Singer, Srebro, & Cotter, 2011).

The dimensionality of the mapping $\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x})$, M_{RFF}, determines the quality of the kernel approximation in Eq. (10). The authors in Rahimi and Recht (2007) proves that an invariant kernels such as the RBF can be approximated within an accuracy of ϵ if $M_{\text{RFF}} = \mathcal{O}(D\epsilon^{-2} \log \frac{1}{\epsilon^2})$, where D is the dimensionality of the input data. Moreover, they empirically shows that smaller values of M_{RFF} can lead to approximations with excellent performances.

In particular, one of the algorithms presented in Rahimi and Recht (2007) proposes to form the mapping $\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x})$ as a concatenation of M_{RFF} RFF components as:

$$\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}) = \sqrt{\frac{2}{M_{\text{RFF}}}} \left[\phi_1(\mathbf{x}), \dots, \phi_{M_{RFF}}(\mathbf{x})\right]^\top, \tag{11}$$

The $\{\phi_m(\mathbf{x})\}_{m=1}^{M_{RFF}}$ are sinusoid functions with D-dimensional frequencies $\boldsymbol{\omega}_m$ sampled following the Fourier transform of the kernel:

$$\phi_m(\mathbf{x}) = \cos(\boldsymbol{\omega}_m^\top \mathbf{x} + v_m), \quad m = 1, \dots, M_{\text{RFF}} \tag{12}$$

where coefficients v_m, $m = 1, \dots, M_{\text{RFF}}$, are uniformly sampled from the interval $[0, 2\pi]$. Since the Fourier transform of an RBF kernel with the spread parameter γ is given by

$$p(\boldsymbol{\omega}) = \sqrt{\frac{\pi}{\gamma}} \exp\left(\frac{-\pi^2 \|\boldsymbol{\omega}\|^2}{\gamma}\right) \tag{13}$$

frequencies $\{\boldsymbol{\omega}_m\}_{m=1}^{M_{\text{RFF}}}$ are sampled from $\mathbb{R}^D$ (the input space) according to a Gaussian distribution with zero mean and spherical covariance matrix with variances proportional to γ (Rahimi & Recht, 2007).

3. Random fourier features connect kernel methods with extreme learning machines and random vector functional link networks

RFFs establish a connection between KMs and ELMs: a KM endowed with an RBF kernel can be approximated by an ELM with a hidden layer formed by the RFFs that approximate the kernel.

Let us start from an already fitted KM endowed with an RBF kernel with spread parameter γ:

$$f(\mathbf{x}) = \boldsymbol{\beta}^\top \boldsymbol{\kappa}_{\text{RBF}}(\mathbf{x}) + b, \tag{14}$$

where $\boldsymbol{\kappa}_{\text{RBF}}(\mathbf{x}) = \left[\kappa_{\text{RBF}}(\mathbf{x}_1,\mathbf{x}),\dots,\kappa_{\text{RBF}}(\mathbf{x}_N,\mathbf{x})\right]^\top$.

Next, $\kappa_{\text{RBF}}(\mathbf{x}_i,\mathbf{x}_j)$ is approximated by a mapping $\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x})$ like the one in Eqs. (11)–(12). Therefore $\boldsymbol{\kappa}_{\text{RBF}}(\mathbf{x})$ can be approximated as

$$\boldsymbol{\kappa}_{\text{RBF}}(\mathbf{x}) \approx \left[\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}_1)^\top\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}),\dots,\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}_N)^\top\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x})\right]^\top = \boldsymbol{\Phi}_{\text{RFF}}\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}), \tag{15}$$

where $\boldsymbol{\Phi}_{\text{RFF}}$ is the $N\times M_{\text{RFF}}$ matrix whose rows are $\{\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}_n)^\top\}_{n=1}^N$.

Therefore, the KM regressor of Eq. (14) can be approximated by

$$f(\mathbf{x}) \approx \boldsymbol{\beta}^\top\boldsymbol{\Phi}_{\text{RFF}}\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}) + b = \mathbf{w}^\top\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}) + b, \tag{16}$$

where $\mathbf{w} = \boldsymbol{\Phi}_{\text{RFF}}^\top\boldsymbol{\beta}$. Notice how the last expression in Eq. (16) resembles that of Eq. (1) plus the bias term. This implies that the RFF based approximation to the regressor in Eq. (14) can be implemented by an ELM following Steps 1–5 and Step 7 of Algorithm 1.

Moreover, Algorithm 1 can be tuned to construct from scratch an ELM that approximates a certain KM regressor without first obtaining the regressor, that is, without actually solving the problem given by Eq. (8). This procedure would reproduce all the steps in Algorithm 1 but the 7th and 9th ones, and replace Step 7 with an optimisation that yields $\mathbf{w}$ through the adaptation of Eq. (8). For example, in the case of kernel ridge regression, Eq. (8) becomes

$$\min_{\boldsymbol{\beta},b} \frac{1}{2}\|\mathbf{K}\boldsymbol{\beta} + b\mathbb{1}_N - \mathbf{y}\|^2 + \frac{\lambda}{2}\boldsymbol{\beta}^\top\mathbf{K}\boldsymbol{\beta} \tag{17}$$

where $\mathbb{1}_N$ is a vector with its N components set to 1 and $\mathbf{K}$ is the $N\times N$ kernel matrix containing all pairwise kernel products among the training data. After introducing the RFF approximation to the RBF kernel, the optimisation in Eq. (17) becomes

$$\min_{\boldsymbol{\beta},b} \frac{1}{2}\|\boldsymbol{\Phi}_{\text{RFF}}\boldsymbol{\Phi}_{\text{RFF}}^\top\boldsymbol{\beta} + b\mathbb{1}_N - \mathbf{y}\|^2 + \frac{\lambda}{2}\boldsymbol{\beta}^\top\boldsymbol{\Phi}_{\text{RFF}}\boldsymbol{\Phi}_{\text{RFF}}^\top\boldsymbol{\beta}. \tag{18}$$

Now we introduce $\mathbf{w} = \boldsymbol{\Phi}_{\text{RFF}}^\top\boldsymbol{\beta}$ to arrive at

$$\min_{\mathbf{w},b} \frac{1}{2}\|\boldsymbol{\Phi}_{\text{RFF}}\mathbf{w} + b\mathbb{1}_N - \mathbf{y}\|^2 + \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w} \tag{19}$$

which is the ridge regression problem of Eq. (2) that optimises the weights connecting the hidden layer with the output layer in the ELM. As long as M_{RFF} is large enough, the RFF approximation of the kernel will be sharp enough to guarantee the quality of the ELM approximation to the KM.

However, if the KM regressor is endowed with a composite kernel resulting from the addition of an RBF kernel with spread parameter γ and a linear kernel, $\boldsymbol{\kappa}_{\text{C}}(\mathbf{x}) = \boldsymbol{\kappa}_{\text{RBF}}(\mathbf{x}) + \mathbf{X}\mathbf{x}$, the RFF approximation of the RBF kernel would lead to a RVFLN approximation of the KM regressor. Notice that the RFF will only approximate the RBF kernel, as the linear one does not need a linear approximation

$$\boldsymbol{\kappa}_{\text{C}}(\mathbf{x}) \approx \boldsymbol{\Phi}_{\text{RFF}}\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}) + \mathbf{X}\mathbf{x}$$

and the regressor model with the composite kernel $f_c(\mathbf{x}) = \boldsymbol{\beta}^\top\boldsymbol{\kappa}_{\text{C}}(\mathbf{x}) + b$ is approximated by

$$f_c(\mathbf{x}) \approx \boldsymbol{\beta}^\top\boldsymbol{\Phi}_{\text{RFF}}\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}) + \boldsymbol{\beta}^\top\mathbf{X}\mathbf{x} + b = \mathbf{w}_{\text{H}}^\top\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}) + \mathbf{w}_{\text{FL}}^\top\mathbf{x} + b \tag{20}$$

where $\mathbf{w}_{\text{H}} = \boldsymbol{\Phi}_{\text{RFF}}^\top\boldsymbol{\beta}$ are the weights connecting the hidden layer units with the neuron of the output layer, and $\mathbf{w}_{\text{FL}} = \mathbf{X}^\top\boldsymbol{\beta}$ are the functional links. Therefore, the RFF based approximation to $f_c(\mathbf{x})$ can be implemented by a RVFL following Algorithm 1.

Analogously to what we showed for the ELM, the RVFL that approximates the KM regressor with composite kernel $\boldsymbol{\kappa}_{\text{C}}(\mathbf{x})$ can be fitted without explicitly solving problem (8). For this purpose, one needs to reproduce the all the steps of Algorithm 1 but the 7th one, and replace Step 7 by an optimisation in terms of variables $\mathbf{w}_{\text{H}}$ and $\mathbf{w}_{\text{FL}}$. This procedure is again illustrated for the case of kernel ridge regression. The starting point is the optimisation that fits the KM with the composite kernel:

$$\min_{\boldsymbol{\beta},b} \frac{1}{2}\|(\mathbf{K}_{\text{RBF}} + \mathbf{K}_{\text{L}})\boldsymbol{\beta} + b\mathbb{1}_N - \mathbf{y}\|^2 + \frac{\lambda}{2}\boldsymbol{\beta}^\top(\mathbf{K}_{\text{RBF}} + \mathbf{K}_{\text{L}})\boldsymbol{\beta} \tag{21}$$

where $\mathbf{K}_{\text{RBF}}$ and $\mathbf{K}_{\text{L}}$ are the RBF and linear kernel matrices obtained with the training set, respectively. Then the RBF kernel is approximated by the RFFs, transforming the optimisation of Eq. (21) in:

$$\min_{\boldsymbol{\beta},b} \frac{1}{2}\|(\boldsymbol{\Phi}_{\text{RFF}}\boldsymbol{\Phi}_{\text{RFF}}^\top + \mathbf{X}\mathbf{X}^\top)\boldsymbol{\beta} + b\mathbb{1}_N - \mathbf{y}\|^2 + \frac{\lambda}{2}\boldsymbol{\beta}^\top(\boldsymbol{\Phi}_{\text{RFF}}\boldsymbol{\Phi}_{\text{RFF}}^\top + \mathbf{X}\mathbf{X}^\top)\boldsymbol{\beta} \tag{22}$$

The next step is to rewrite $(\boldsymbol{\Phi}_{\text{RFF}}\boldsymbol{\Phi}_{\text{RFF}}^\top + \mathbf{X}\mathbf{X}^\top)$ as a matrix product:

$$\boldsymbol{\Phi}_{\text{RFF}}\boldsymbol{\Phi}_{\text{RFF}}^\top + \mathbf{X}\mathbf{X}^\top = [\boldsymbol{\Phi}_{\text{RFF}}\mathbf{X}]\begin{bmatrix}\boldsymbol{\Phi}_{\text{RFF}}^\top\\ \mathbf{X}^\top\end{bmatrix} = [\boldsymbol{\Phi}_{\text{RFF}}\mathbf{X}]\,[\boldsymbol{\Phi}_{\text{RFF}}\mathbf{X}]^\top \tag{23}$$

This rewriting enables

$$\begin{aligned}(\boldsymbol{\Phi}_{\text{RFF}}\boldsymbol{\Phi}_{\text{RFF}}^\top + \mathbf{X}\mathbf{X}^\top)\boldsymbol{\beta} &= [\boldsymbol{\Phi}_{\text{RFF}}\mathbf{X}]\begin{bmatrix}\boldsymbol{\Phi}_{\text{RFF}}^\top\\ \mathbf{X}^\top\end{bmatrix}\boldsymbol{\beta}\\ &= [\boldsymbol{\Phi}_{\text{RFF}}\mathbf{X}]\begin{bmatrix}\boldsymbol{\Phi}_{\text{RFF}}^\top\boldsymbol{\beta}\\ \mathbf{X}^\top\boldsymbol{\beta}\end{bmatrix} = [\boldsymbol{\Phi}_{\text{RFF}}\mathbf{X}]\begin{bmatrix}\mathbf{w}_{\text{H}}\\ \mathbf{w}_{\text{FL}}\end{bmatrix}\end{aligned}$$

and

$$\boldsymbol{\beta}^\top(\boldsymbol{\Phi}_{\text{RFF}}\boldsymbol{\Phi}_{\text{RFF}}^\top + \mathbf{X}\mathbf{X}^\top)\boldsymbol{\beta} = \boldsymbol{\beta}^\top[\boldsymbol{\Phi}_{\text{RFF}}\mathbf{X}]\begin{bmatrix}\boldsymbol{\Phi}_{\text{RFF}}^\top\\ \mathbf{X}^\top\end{bmatrix}\boldsymbol{\beta} = [\mathbf{w}_{\text{H}}^\top\mathbf{w}_{\text{FL}}^\top]\begin{bmatrix}\mathbf{w}_{\text{H}}\\ \mathbf{w}_{\text{FL}}\end{bmatrix}$$

Defining $\boldsymbol{\Phi}_{\text{RVFL}} = [\boldsymbol{\Phi}_{\text{RFF}}\mathbf{X}]$ and $\mathbf{w}_{\text{RVFL}} = [\mathbf{w}_{\text{H}}^\top\mathbf{w}_{\text{FL}}^\top]^\top$ enables writing the optimisation as

$$\min_{\mathbf{w}_{\text{RVFL}},b} \frac{1}{2}\|\boldsymbol{\Phi}_{\text{RVFL}}\mathbf{w}_{\text{RVFL}} + b\mathbb{1}_N - \mathbf{y}\|^2 + \frac{\lambda}{2}\mathbf{w}_{\text{RVFL}}^\top\mathbf{w}_{\text{RVFL}} \tag{24}$$

what results in the ridge regression that one needs to solve to obtain the output weights of a RVFLN, as shown in Eq. (4).

4. Random fourier features Bayesian linear regression (RFF-BLR) for multitask problems

Algorithm 1: Approximation of a KM by either a ELM or a RVFL network

```
Data: $M_{\text{RFF}}$, $\gamma$, $\mathbf{X}$, $\mathbf{y}$
1 Set up a single hidden layer with $M_{\text{RFF}}$ neurons with cosine
    activations;
2 Randomly sample $\boldsymbol{\omega}_m$, $m = 1,\dots,M_{\text{RFF}}$ from $p(\boldsymbol{\omega})$ (see Eq. (13));
3 Set the weights that connect the input layer with each neuron
    in the hidden layer to $\boldsymbol{\omega}_m$, $m = 1,\dots,M_{\text{RFF}}$;
4 Randomly sample $\upsilon_m$, $m = 1,\dots,M_{\text{RFF}}$ from $[0,2\pi]$;
5 Set the bias terms of the neurons in the hidden layer to $\upsilon_m$,
    $m = 1,\dots,M_{\text{RFF}}$;
6 Construct $\boldsymbol{\Phi}_{\text{RFF}}$;
7 Get $\boldsymbol{\beta}$ as the solution to the optimization problem that gives
    the KM Eq. (8);
8 Set weights $\mathbf{w}$ connecting the hidden layer with the single
    neuron of the output layer to $\mathbf{w} = \boldsymbol{\Phi}_{\text{RFF}}^\top\boldsymbol{\beta}$.;
9 If RVFL, set the functional links to $\mathbf{w}_{\text{FL}} = \mathbf{X}^\top\boldsymbol{\beta}$ (weights $\mathbf{w}_{\text{H}}$
    connecting the hidden layer with the single neuron of the
    output layer are also obtained from $\mathbf{w}_{\text{H}} = \boldsymbol{\Phi}_{\text{RFF}}^\top\boldsymbol{\beta}$);
```

The derivation of the Bayesian formulation starts with the definition of the corresponding generative model, followed by the development of the variational inference of the model parameters.

The description of the Bayesian linear regression in the space defined by the RFFs model is started from the single task case.

$$y_{n,c} = \boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}_n)^\top\mathbf{w}_c + b_c + \eta_{n,c} \tag{25}$$

where $\mathbf{w}_c$ and b_c are the weight vector and bias term of task c, respectively, and $\eta_{n,c}$ represents the Gaussian noise with precision τ related to the nth observation in the training set in the cth task. The model in Eq. (25) is easily extended to multitask regression by introducing

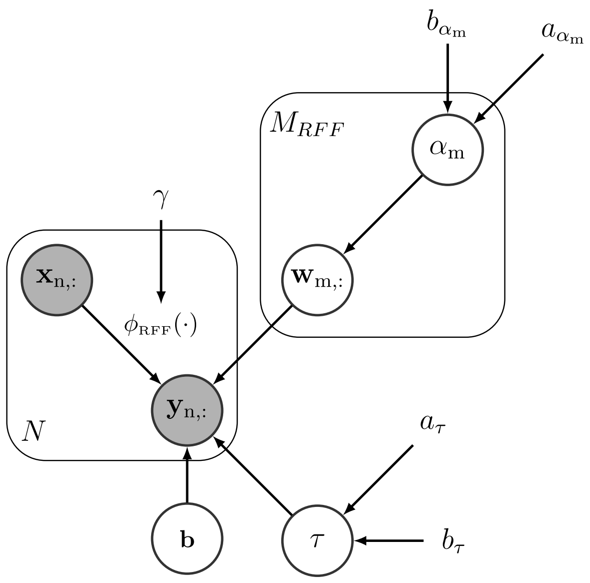


Fig. 1. Plate diagram for the RFF-BLR model. Grey circles denote observed variables, white circles unobserved r.v. Symbols without a circle correspond to deterministic (although unknown) hyperparameters. Symbol $\boldsymbol{\phi}_{\mathrm{RFF}}(\cdot)$ indicates the fact that the input variables are connected with the targets through that mapping.

the multitask targets $\mathbf{y}_{n,:}$, and stacking the weight vectors $\{\mathbf{w}_c\}_{c=1}^{C}$ in a $M_{\mathrm{RFF}} \times C$ matrix $\mathbf{W}$ and the C bias terms in vector $\mathbf{b} \in \mathbb{R}^{1\times C}$:

$$\mathbf{y}_{n,:} = \boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_n)^{\top}\mathbf{W} + \mathbf{b} + \boldsymbol{\eta}_{n,:}, \tag{26}$$

where C-dimensional vector $\boldsymbol{\eta}_{n,:}$ contains the noises related to the nth observation of the training set in all the tasks. The final model is completed by stacking the equations corresponding to the N training data:

$$\mathbf{Y} = \boldsymbol{\Phi}_{\mathrm{RFF}}\mathbf{W} + \mathbb{1}_N\mathbf{b} + \boldsymbol{\eta}, \tag{27}$$

where $\mathbb{1}_N$ is a column vector of ones with dimension N and $\boldsymbol{\eta}$ is a $N\times C$ matrix with the all the noise elements. Fig. 1 depicts a plate diagram detailing the dependence relationships among all the random variables and deterministic hyperparameters involved in the generative model associated with Eq. (27). Distributions associated with the random variables (r.v) of the model can be defined as

$$\mathbf{y}_{n,:} \sim \mathcal{N}\left(\boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:})^{\top}\mathbf{W} + \mathbf{b}, \tau^{-1}\mathbf{I}_C\right), \tag{28}$$

$$\mathbf{w}_{m,:} \sim \mathcal{N}\left(\mathbf{0}, {\alpha_m}^{-1}\mathbf{I}_C\right), \tag{29}$$

$$\alpha_m \sim \Gamma\left(\mathrm{a}^0_{\alpha_\mathrm{m}}, \mathrm{b}^0_{\alpha_\mathrm{m}}\right), \tag{30}$$

$$\mathbf{b} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}_C\right), \tag{31}$$

$$\tau \sim \Gamma\left(\mathrm{a}^0_\tau, \mathrm{b}^0_\tau\right), \tag{32}$$

where $\mathcal{N}(\mathbf{a}, \mathbf{A})$ denotes a Gaussian distribution with mean $\mathbf{a}$ and covariance matrix $\mathbf{A}$, $\Gamma(\alpha, \beta)$ is a Gamma distribution with parameters α and β, $\mathbf{I}_C$ is an identity matrix of dimension C; $\mathrm{a}^0_{\alpha_\mathrm{m}}$, $\mathrm{b}^0_{\alpha_\mathrm{m}}$ are α_m prior hyperparameters, and a^0_τ, b^0_τ are τ prior hyperparameters. Moreover, $\mathbf{w}_{m,:}$ denotes the mth row of matrix $\mathbf{W}$.

In the definition of the generative model, r.v. $\mathbf{w}_{m,:}$ and α_m form an Automatic Relevance Determination (ARD) prior (Neal, 2002) that promotes sparsity over the rows of $\mathbf{W}$. According to the model, each $\mathbf{w}_{m,:}$ is drawn from a Gaussian with zero mean and precision α_m. Therefore, an α_m ending up with a high value after the inference forces that all the elements in $\mathbf{w}_{m,:}$ will tend to zero. Since $\mathbf{w}_{m,:}$ connects the mth hidden unit with the output layer, zeroing all its elements forces this neuron to become irrelevant for the computation of the output of the model for all the tasks. In summary, this formulation allows for the automatic selection of the most relevant RFFs jointly for all tasks, enabling the transfer of knowledge among them.

Once the probabilistic model is defined, the next step is to find values for the model parameters $\mathbf{W}$, $\mathbf{b}$, $\{\alpha_m\}_{m=1}^{M_{\mathrm{RFF}}}$ and τ through an inference process that maximises the posterior probability of such parameters conditioned on the observed training set. Let us denote by Θ an array that includes all unobserved r.v. and parameters of the model: $\Theta = [\mathbf{W}, \boldsymbol{\alpha}, \mathbf{b}, \tau]$, where $\boldsymbol{\alpha}$ is the vector resulting from the concatenation of $\{\alpha_m\}_{m=1}^{M_{\mathrm{RFF}}}$. The usual procedure in the Bayesian framework is to approximate the untractable posterior $p(\Theta|\boldsymbol{\Phi}_{\mathrm{RFF}}, \mathbf{Y})$ by a tractable distribution $q(\Theta)$. In this case we opt for the fully factorised mean-field approximation:

$$p(\Theta \mid \boldsymbol{\Phi}_{\mathrm{RFF}}, \mathbf{Y}) \approx q(\mathbf{W})\, q(\tau)\, q(\mathbf{b}) \prod_{m=1}^{M_{\mathrm{RFF}}} q\left(\alpha_m\right). \tag{33}$$

The goal of the approximation is to find values for the elements in Θ so that $q(\Theta)$ becomes a close approximation of $p(\Theta|\boldsymbol{\Phi}_{\mathrm{RFF}}, \mathbf{Y})$ in the sense that the Kullback–Leibler (KL) divergence between the two distributions is minimised. This KL divergence is defined as

$$KL(q \parallel p) = \int q(\Theta) \ln\left(\frac{q(\Theta)}{p(\Theta|\mathbf{X},\mathbf{Y})}\right) d\Theta = \tag{34}$$

$$\int q(\Theta) \ln\left(\frac{q(\Theta)}{p(\mathbf{Y},\Theta|\mathbf{X})}\right) d\Theta + \ln\left(p(\mathbf{Y}|\mathbf{X})\right). \tag{35}$$

Let us now denote

$$L(q) = -\int q(\Theta) \ln\left(\frac{q(\Theta)}{p(\Theta,\mathbf{Y}|\mathbf{X})}\right) d\Theta, \tag{36}$$

then

$$KL(q \parallel p) = \ln\left(p(\mathbf{Y}|\mathbf{X})\right) - L(q)$$

Since the evidence $\ln(p(\mathbf{X},\mathbf{Y}))$ is fixed for any given training set and the KL divergence is non-negative, $L(q)$ becomes a lower bound of the evidence:

$$\ln\left(p(\mathbf{X},\mathbf{Y})\right) \geq L(q).$$

As a consequence, the maximisation of Eq. (36) yields $q^*(\Theta)$ that approximates the posterior $p(\Theta|\mathbf{X},\mathbf{Y})$. The joint distribution in $L(q)$ can be developed as

$$\begin{aligned} &p(\mathbf{Y}, \mathbf{W}, \boldsymbol{\alpha}, \tau, \mathbf{b} \mid \boldsymbol{\Phi}_{\mathrm{RFF}}) \\ &= \prod_{\mathrm{n}=1}^{\mathrm{N}} p\left(\mathbf{y}_{n,:} \mid \boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:}), \mathbf{W}, \mathbf{b}, \tau\right) p(\mathbf{W} \mid \boldsymbol{\alpha})\, p(\boldsymbol{\alpha})\, p(\mathbf{b})\, p(\tau), \end{aligned} \tag{37}$$

where we have introduced the fact that the observations $\{\mathbf{x}_n\}_{n=1}^{N}$ exclusively appear in the model through the RFFs.

The mean-field posterior structure along with the lower bound results in a feasible coordinate-ascent-like optimisation algorithm (Bishop, 2006, Chapter 10) in which the optimal value of each factor in Eq. (33) can be computed if the rest remain fixed using the following expression

$$\log(q^*(\theta_i)) \propto \mathbb{E}_{\Theta_{-i}}\left[\log p(\Theta, \mathbf{y}_{1,:}, \ldots, \mathbf{y}_{N,:} \mid \boldsymbol{\Phi}_{\mathrm{RFF}})\right], \tag{38}$$

where Θ_{-i} comprises all r.v. in Θ but θ_i. This new formulation is generally feasible as it does not require a complete marginalisation of Θ from the joint distribution, which is calculated with Eq. (37). Therefore, model update rules for each r.v. can be obtained through Eq. (38). The resulting approximate distributions are shown in Table 1. As the estimated distribution for each r.v. also depends on some other r.v., *e.g.* $\langle \mathbf{w}_{:,c}\rangle$ depends on $\langle\tau\rangle$ and $\langle \mathrm{b}_\mathrm{c}\rangle$, we need to iterate over the variables, analysing the evolution of the lower bound after each iteration until convergence. Appendix A describes the complete development of all q^* distributions as well as the final mean-field factor update rules.

Table 1
Update rules for the distributions q of the r.v. in the graphical model. Here, $diag(\mathbf{x})$ is an operator that transforms a vector into a diagonal matrix with diagonal $\mathbf{x}$, $\mathbb{1}_N$ is a column vector of ones of dimension N, and $< x >$ denotes the expectation with respect to $q()$ of random variable x. These expressions have been obtained using the update rules of the mean-field approximation in Eq. (38).

Variable	q^* distribution	Parameters
$\mathbf{W}$	$\prod_{c=1}^{C} \mathcal{N}(\mathbf{w}_{:,c} \mid \langle \mathbf{w}_{:,c} \rangle, \Sigma_{\mathbf{W}})$	$\langle \mathbf{w}_{:,c} \rangle = \langle \tau \rangle \Sigma_{\mathbf{W}} \boldsymbol{\Phi}_{\mathrm{RFF}}^\top (\mathbf{y}_{:,c} - \mathbb{1}_N \langle \mathrm{b_c} \rangle)$ $\Sigma_{\mathbf{W}}^{-1} = diag(\langle \boldsymbol{\alpha} \rangle) + \langle \tau \rangle \boldsymbol{\Phi}_{\mathrm{RFF}}^\top \boldsymbol{\Phi}_{\mathrm{RFF}}$
α_m	$\Gamma(\alpha_m \mid \mathbf{a}_{\alpha_m}, \mathbf{b}_{\alpha_m})$	$\mathbf{a}_{\alpha_m} = \mathrm{a}^0_{\alpha_\mathrm{m}} + \frac{C}{2}$ $\mathbf{b}_{\alpha_m} = \mathrm{b}^0_{\alpha_\mathrm{m}} + \frac{1}{2} \mathbf{w}_{m,:}{}^{\top} \mathbf{w}_{m,:}$
$\mathbf{b}$	$\mathcal{N}(\mathbf{b} \mid \langle \mathbf{b} \rangle, \Sigma_{\mathbf{b}})$	$\langle \mathbf{b} \rangle = \langle \tau \rangle \sum_{\mathrm{n=1}}^{\mathrm{N}} (\mathbf{y}_{n,:} - \boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:}) \langle \mathbf{W} \rangle) \Sigma_{\mathbf{b}}$ $\Sigma_{\mathbf{b}}^{-1} = (N \langle \tau \rangle + 1) \mathbf{I}_C$
τ	$\Gamma(\tau \mid \mathbf{a}_\tau, \mathbf{b}_\tau)$	$\mathbf{a}_\tau = \mathrm{a}^0_\tau + \frac{NC}{2}$ $\mathbf{b}_\tau = \mathrm{b}^0_\tau + \frac{1}{2} \sum_{\mathrm{n=1}}^{\mathrm{N}} \sum_{\mathrm{c=1}}^{\mathrm{C}} y^2_{n,c} + \frac{1}{2} Tr\{\langle \mathbf{W}^\top \mathbf{W} \rangle \boldsymbol{\Phi}_{\mathrm{RFF}} \boldsymbol{\Phi}_{\mathrm{RFF}}^\top\}$ $-Tr\{\mathbf{Y} \langle \mathbf{W}^\top \rangle \boldsymbol{\Phi}_{\mathrm{RFF}}^\top\} - \sum_{\mathrm{n=1}}^{\mathrm{N}} \mathbf{y}_{n,:} \langle \mathbf{b}^\top \rangle$ $+ \sum_{\mathrm{n=1}}^{\mathrm{N}} \boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:}) \langle \mathbf{W} \rangle \langle \mathbf{b}^\top \rangle + \frac{N}{2} \langle \mathbf{b}\mathbf{b}^\top \rangle$

4.1. Optimisation of the kernel parameter γ in RFF-BLR

RFFs rely on a crucial hyperparameter, γ, which controls the spread of the kernel. Traditionally, determining an optimal value for γ involves a costly cross-validation that introduces additional complexity and uncertainty into the fitting of the model. To streamline this process, we propose to introduce the optimisation of γ within the Bayesian framework.

To achieve this, we decompose the process of generating the RFF frequencies, $\boldsymbol{\omega}_m$, into two steps (instead of directly applying Eq. (13)). Firstly, we generate a set of M_{RFF} normalised frequencies, $\bar{\boldsymbol{\omega}}_m$, by sampling from a Gaussian distribution with zero mean and an identity covariance matrix. Then, we scale these frequencies by γ to obtain the actual $\boldsymbol{\omega}_m$ values, *i.e.*,

$$\boldsymbol{\omega}_m = \gamma\ \bar{\boldsymbol{\omega}}_m \qquad m = 1, \dots, M_{RFF}.$$

The set of vectors $\{\bar{\boldsymbol{\omega}}_m\}_{m=1}^{M_{RFF}}$ is initially generated and kept fixed during inference process, while γ is iteratively updated. Thus, this process allows us to automatically learn the optimal variance for the set of the RFF frequencies without the need for cross-validation of γ.

The optimisation problem that yields the update of γ results from the particularisation of the lower bound defined in Appendix B to the terms that depend on γ, leaving the other terms as constant. Since $\boldsymbol{\Phi}_{\mathrm{RFF}}$ is the only element that depends on γ, this optimisation results in the maximisation of

$$\begin{aligned} LB' \quad = \quad & \langle \tau \rangle \sum_{\mathrm{n=1}}^{\mathrm{N}} \sum_{\mathrm{c=1}}^{\mathrm{C}} \Big(y_{n,c} \langle \mathbf{w}_{:,c}^\top \rangle \boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:})^\top \\ & - \frac{1}{2} \langle \mathbf{w}_{:,c} \mathbf{w}_{:,c}^\top \rangle \boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:})^\top \boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:}) \\ & - \boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:}) \langle \mathbf{w}_{:,c} \rangle \langle \mathrm{b_c} \rangle \Big) + \mathrm{const.} \end{aligned} \tag{39}$$

where const includes the terms that do not depend on $\boldsymbol{\Phi}_{\mathrm{RFF}}$. Then, the model can alternate between mean-field updates over the variational bound and direct maximisation of Eq. (39) w.r.t. γ using any gradient ascend method (*e.g.* Adam Kingma & Ba, 2015).

4.2. RFF-BLR with functional links

The RFF-BLR formulation can be straightforwardly extended to include functional links in the architecture of the network, allowing it to effectively handle both linear and nonlinear relations in the data. This can be obtained by simply:

- Adding D rows to $\mathbf{W}$ to accommodate the functional links so that now $\mathbf{W} \in \mathbb{R}^{(M+D)\times C}$.
- Add the corresponding precisions $\{\alpha_m\}_{m=M+1}^{M+D}$, one per each new row $\mathbf{w}_{m,:}$, to $\boldsymbol{\alpha}$ so now $\boldsymbol{\alpha} \in \mathbb{R}^{M+D}$.
- Append matrix $\mathbf{X}$ as new columns to $\boldsymbol{\Phi}_{\mathrm{RFF}}$: $\boldsymbol{\Phi}_{\mathrm{RFF}} \leftarrow [\boldsymbol{\Phi}_{\mathrm{RFF}} \mathbf{X}]$.

To unify the notation between ELM and RVFL architectures, let us define M_o as the dimension of the output layer. In practice, for the ELM $M_\mathrm{o} = M_{\mathrm{RFF}}$ and for the RVFL $M_\mathrm{o} = M_{\mathrm{RFF}} + D$.

A summary of the RFF-BLR method for multitask regression including all the relevant steps described in these Sections is shown in Algorithm 2.

Algorithm 2: RFF-BLR for multitask problems

```
   Input: $M_{\mathrm{RFF}}$, $\mathbf{X}$, $\mathbf{Y}$, $RVFL$ (bool), $\gamma_{ini}$
1  sample $\{\bar{\boldsymbol{\omega}}_m\}_{m=1}^{M_{\mathrm{RFF}}}$ from $\mathcal{N}(\mathbf{0}, \mathbf{I}_D)$;
2  calculate $\boldsymbol{\omega}_m = \gamma_{ini} \bar{\boldsymbol{\omega}}_m$ ;
3  sample $\{\upsilon\}_{m=1}^{M_{\mathrm{RFF}}}$ from $[0, 2\pi]$;
4  construct $\boldsymbol{\Phi}_{\mathrm{RFF}}$ with $\mathbf{X}$, $\{\boldsymbol{\omega}\}_{m=1}^{M_{\mathrm{RFF}}}$ and $\{\upsilon\}_{m=1}^{M_{\mathrm{RFF}}}$;
5  if $RVFL$ then
6      $\boldsymbol{\Phi} = [\boldsymbol{\Phi}_{\mathrm{RFF}} \mathbf{X}]$;
7  else
8      $\boldsymbol{\Phi} = \boldsymbol{\Phi}_{\mathrm{RFF}}$;
9  end
10 initialize $\langle \mathbf{W} \rangle$, $\Sigma_{\mathbf{W}}^{-1}$, $\mathbf{a}_{\alpha_m}$, $\mathbf{b}_{\alpha_m}$, $\langle \mathbf{b} \rangle$, $\mathbf{a}_\tau$ and $\mathbf{b}_\tau$;
11 repeat
12     run 10 iterations of the inference with the update equations
         of Table 1;
13     prune the irrelevant components ($\mathbf{w}_{m,:}$, $\alpha_m$ and the
         corresponding hidden unit);
14     update $\gamma$ according to Section 4.1;
15     update the non-pruned vectors $\boldsymbol{\omega}_m = \gamma \bar{\boldsymbol{\omega}}_m$;
16     recompute $\boldsymbol{\Phi}_{\mathrm{RFF}}$ with the updated frequencies;
17 until convergence;
```

4.3. Theoretical computational cost analysis

Table 2 includes the computational cost of each of the update steps in Table 1. The overall cost of the whole algorithm is shown in the last row of Table 2, highlighting the most common case in which the dimension of the output layer (M_o) is greater than the number of data (N), and also greater than the number of output tasks (C). This is the case contemplated in the experiments, as we have set $M_{\mathrm{RFF}} = 1.5\ N$.

The theoretical cost of the algorithm is linear with C, quadratic with N and cubic with M_o. However, this result is, in fact, an upper bound to the actual cost of fitting the model. As the experimental section shows, the sparsity inducing prior allows to automatically eliminate those components of the hidden layer (and of the functional links) that are not relevant. This pruning effectively reduces the number of parameters of the model and, therefore, the computational cost of the training.

5. Experimental work

This section starts with an extense performance analysis on broadly used MTR benchmarks to help gain insight into the true capabilities of the proposal in real-world scenarios. Afterwards, we focus on the analysis of the sparsity imposed by the Bayesian formulation. Next, we move to an ablation study of RFF-BLR that illustrates the impact of each of the contributions included in the model definition. The section concludes with an experimental computational cost study that completes that of Section 4.3. Furthermore, a *Python* implementation of RFF-BLR together with some example notebooks is openly available in this GitHub repository: https://github.com/sevisal/RFF-BLR.

Table 2
Upper bound of the computational cost of each update step and final cost of the model. N is the number of data, M_o the number of RFF components (plus the number of links if applicable) and C the number of output tasks.

Variable update	Computational cost
$\mathbf{W}$	$\mathcal{O}(CNM_o^2) + \mathcal{O}(M_o^3) + \mathcal{O}(CM_oN^2)$
$\boldsymbol{\alpha}$	$\mathcal{O}(CM_o)$
$\mathbf{b}$	$\mathcal{O}(CNM_o)$
τ	$\mathcal{O}(CM_o^2) + \mathcal{O}(CN^2) + \mathcal{O}(NM_o^2)$
RFF-BLR	$\mathcal{O}(CNM_o^2) + \mathcal{O}(M_o^3) + \mathcal{O}(CM_oN^2) + \mathcal{O}(CN^2)$
RFF-BLR ($M_o > N > C$)	$\mathcal{O}(CNM_o^2) + \mathcal{O}(M_o^3)$

5.1. Performance analysis with multitaks regression benchmarks

5.1.1. Baselines

The first baseline is the **Multi-layer Multi-target Regression** (MMR) method proposed in Zhen, Yu, Zheng, et al. (2018). This method uses an RBF kernel to construct a non-linear mapping from the input space into a set of latent intermediate features. These latent features are connected with the output space with a linear mapping that captures the interdependencies among the different tasks. The fitting of this model depends on three hyperparameters that control the regularisation and the kernel lengthscale. Following (Zhen, Yu, Zheng, et al., 2018) we adjusted the values of the regularisation parameters exploring 9 values in the range of 10^{-5} to 10^{3} on the logarithmic scale and used the average of all pairwise distances between training instances to estimate the kernel lenghtscale.

The next set of baselines cover the randomised feedforward NNs spectrum:

- **Generalised Outlier Robust ELM (GOR-ELM)** (da Silva et al., 2020) is a feed-forward NN that combines $\ell_{2,1}$ regularisation with Elastic Net theory to work with MTR problems. The neurons forming its hidden layer are equipped with sigmoid functions as nonlinearities. Following (da Silva et al., 2020) we used 1000 neurons in the hidden layer and cross-validated the value of the model hyperparameters α in the range $\{0, 0.25, 0.5, 0.75, 1\}$ and λ in the range $2^{-20:1:20}$.
- **Random Vector Functional Link (RVFL)** (Husmeier, 1999) is a feedforward NN with stochastically generated weights and biases for the hidden layer. The direct connectivity between the input and output layers bypasses this layer. The activation function used was a sigmoid.
- **Ensemble deep RVFL (edRVFL)** (Li et al., 2023) uses an ensemble of diverse RVFL models to improve prediction accuracy and model robustness. We used the sigmoid activation function. The embedding consisted of 50 RVFLs, chosen to facilitate complete representation learning.
- **Neuro-Fuzzy RVFL (NFRVFL)** (Sajid, Malik, Tanveer, & Suganthan, 2024) incorporates the principles of fuzzy logic into the RVFL framework, addressing the issues of uncertainty and imprecision of the data. Consequently, it augments the RVFL's capacity to learn intricate and ambiguous patterns in data. The activation function used was a sigmoid, the number of hidden neurons was set at 1.5 times the total number of training samples, and the clustering method used was K-means. We cross-validated the regularisation hyperparameter C exploring a grid of $10^{-5:1:5}$ and for the number of Fuzzy Nodes we used a grid of $5:5:50$, as suggested in Sajid, Malik, Tanveer, and Suganthan (2024).

In addition, we have considered another set of baselines that directly learn a linear combination of the RFFs and impose sparsity on the weight matrix ($\mathbf{W}$):

- **Multi-Task Feature Learning (MTFL)** (Argyriou, Evgeniou, & Pontil, 2006) uses the group LASSO to regularise the features used by different tasks, imposing sparsity on the weight matrix. We cross-validated the regularisation hyperparameter exploring a grid of $10^{-5:1:5}$ and for the RBF kernel hyperparameter γ we used a grid of $\frac{1}{D}2^{-10:1:0}$, where D is the dimensionality of the input space.
- **SB-ELM** (Luo et al., 2013) induces sparsity in the output layer by using an ARD prior to perform automatic feature selection. Since the SB-ELM is not a proper multitask method, we train an independent model for each output task.

Finally, we also included two NN models using the RBF nonlinearities in several hidden layers that offer a larger expressive power than that of the single hidden layer NNs of this study:

- **Feed-forward Neural Network (FNN)** (Murtagh, 1991) with non-linear relations between the inputs and the multiple outputs. We validated five configurations: (i) one hidden layer with 100 neurons, (ii) two hidden layers with 100 and 50 neurons, (iii) three hidden layers with 100, 50 and 100 neurons, (iv) four hidden layers with 100, 50, 50 and 100 neurons, and (v) five hidden layers with 100, 50, 25, 50 and 100 neurons.
- **Heterogeneous Incomplete - Variational AutoEncoder (HI-VAE)** (Nazabal, Olmos, Ghahramani, & Valera, 2020) is an adaptation of the Variational AutoEnconder that captures the latent representation of the data while being able to work with heterogeneous data. We used the layer configuration suggested in Nazabal et al. (2020): three layers of dimensions 50-50-50, respectively.

Table 3
Characteristics of the multitask databases from the *Mulan* repository.

Database	Samples	Features	Tasks	Application
atp1d	337	411	6	Airline ticket prices
atp7d	296	411	6	Airline ticket prices
oes97	334	263	16	Employment rate
oes10	403	298	16	Employment rate
edm	154	16	2	Electrical discharge machining
jura	359	15	3	Soil metal concentrations
wq	1,060	16	14	Water quality
enb	768	8	2	Building energy efficiency

5.1.2. Datasets and experimental setup

We used eight MTR datasets from the *Mulan* repository (Džeroski, Demšar, & Grbović, 2000; Karalič & Bratko, 1997; Spyromitros-Xioufis et al., 2016). Table 3 summarises their main characteristics, including the specific applications for which they were intended. These applications cover a variety of domains, from the commerce field, like the forecasting of airline ticket prices, to the demographic and socioeconomic characterisation of a community, like the estimation of the number of full-time employees in a particular area. Other applications are related to the sustainability and environment protection domains, such as the estimation of the concentration of certain metals in the soil of a region, the forecasting of the chemical composition of river water from its biological traits, and the prediction of the energy efficiency of a building. Finally, the industrial domain is also present through the electrical discharge machining application consisting of the optimisation of several parameters of this manufacturing technique.

Each dataset was evaluated following a 10-fold Cross-Validation (CV). For models that need to cross-validate hyperparameters, we adopted a nested cross-validation scheme within each training partition of the main 10-fold cross-validation. We standardised the input data for all models except for MMR, which validates whether to standardise the data, and GOR-ELM, which uses min–max scaling, as suggested in da Silva et al. (2020). We use the coefficient of determination (R^2) to compare the performance of the different methods and adjust their hyperparameters. For a test set of size N_t the value of the coefficient of determination is given by the following expression:

$$R^2 = 1 - \frac{\sum_{t=1}^{N_t} (y_t - \hat{y}_t)^2}{\sum_{t=1}^{N_t} (y_t - \bar{y})^2}$$

where $\{y_t\}_{t=1}^{N_t}$ are the true targets of the test set, $\{\hat{y}_t\}_{t=1}^{N_t}$ are the predictions output by the model for the N_t test data and $\bar{y}$ is the average of the targets in the training set. This accuracy score achieves a maximum value of 1 when the model can approximate all the targets in the test set without error. Therefore, the higher the value of this score, the more accurate the model. For each dataset, we report as the final score the arithmetic average (and standard deviations) of all R^2 scores obtained in all tasks and CV folds.

For the methods RVFL, edRVFL, SB-ELM, and RFF-BLR, we have established $M_{\text{RFF}} = 1.5\ N$ and conducted 10 random initialisations, choosing the model that exhibited the best performance. To determine the number of iterations of the inference process of RFF-BLR and SB-ELM, we used a convergence criteria based on the evolution of the lower bound. For both models, latent factors are automatically pruned through the application of the ARD prior embedded within the projection matrix $\mathbf{W}$. This pruning occurs when a latent factor's magnitude falls below 10^{-1} across all features, at which point it is eliminated. In particular, we stop the algorithm either when $mean(LB[-101 : -2]) > LB[-1](1-10^{-6})$, where $LB[-1]$ is the lower bound at the last iteration and $mean(LB[-101 : -2])$ the mean value of the previous values of the lower bound, or when it reaches 10^3 iterations. Concerning our approach and the intrinsic randomness of the RFF, we perform 10 random initialisations of the frequencies, $\boldsymbol{\omega}_m$, retaining the one that achieves the optimal lower bound after a single mean-field update iteration. We decided to initialise parameter $\gamma = \frac{1}{2\sigma^2} = \frac{1}{2}$. Besides, we adopt an asymmetrical alternation approach, carrying out 10 mean-field update iterations for each maximisation step with respect to γ.

5.1.3. Experimental results

Table 4 displays the results of the empirical comparison between RFF-BLR and the baselines. These results show that RFF-BLR outperforms the baselines in all databases except two. In particular, it achieves an improvement of around 0.14 in *oes97* and of 0.21 in *jura* over the R^2 achieved by the best competitors. For the rest of the databases, it consistently provides good performance, obtaining an R^2 average score of 0.72 – 0.73 while the best baselines, GOR-ELM and MTFL, obtain an average R^2 of 0.62. Note that, even though the mean performance of edRVFL is better than our proposal's in *wq*, the low standard deviation makes our approach the most stable.

With respect to the impact of the functional links in RFF-BLR, it can be observed that on average their introduction produces a slight improvement of the results. Specifically, it is worth mentioning the good performance of RFF-BLR with functional links in *oes97* and *oes10*, the datasets with the largest number of tasks.

With respect to the baselines performance, the first remark is the clear superiority of the MTR models over the single-task SB-ELM. Moreover, two of the single hidden layer NNs with multitask formulation, MTFL and GOR-ELM, outperform multitask NNs with several hidden layers (HI-VAE and FNN). Our intuition behind this fact is that the small size of the training datasets hampers the models with more expressive power. With respect to the influence of the non-linear function on the performance of the model, the behaviour is not consistent along all the datasets. For example, the baseline equipped with a sigmoid as non-linearity (GOR-ELM) achieves the best performance in *atp1d* and *jura*, whereas the MMR endowed with an RBF kernel becomes the best baseline in dataset *wq*, and the MTFL with RFF produces the best results in *oes97* and *oes10* datasets. In this context, it is worth noting the fact that RFF-BLR incorporates both RBF and RFF dual views of the same model in a formulation that does a good job adjusting the expressive power of the model to the needs of each dataset, hence achieving this significantly better performance.

5.2. Sparsity analysis

Let us now focus the discussion on the level of sparsity achieved by the RFF-BLR method (with and without functional links) in the benchmarks. Fig. 2 shows the dependence of the final accuracy of the model with the quotient of the initial value of M_o (number of RFFs + number of functional links) and the number of training data N. We include here the results in two databases, while the results for the additional six databases are available in Appendix C. The curves in the plots show the R^2 averaged for all tasks for each algorithm under study. The bars below the curves indicate the number of M_o that define the final model once the optimisation is finished, and the vertical dashed line marks the point where the initial number of neurons in the hidden layer equals the size of the training set. We include as a baseline for comparison the SB-ELM (orange curves), as it also follows a sparse Bayesian framework specially targeted to develop ELMs with a very compact hidden layer. The results show that the two versions of the RFF-BLR achieve clearly higher R^2 scores than SB-ELM; besides, the RFF-BLR performance is not very sensitive to the initial value of M_o, unlike for SB-ELM. However, it is noteworthy the decay of the R^2 value in the RFF-BLR with functional links when the number of initial components is too high; this behaviour is due to the high number of parameters generated in the model (compared to the amount of data) that limit its learning. This effect is not observed in the RFF-BLR without functional links because it needs to optimise a smaller number of parameters.

The final number of components used by each model indicates that Bayesian optimisation succeeds in selecting this quantity. Note that the version with functional links uses $M_o = M_{\text{RFF}} + D$ so the percentage of selected components is relative to that number, which is D elements lower than for the other two cases. We can observe a higher sparsity in the version with functional links as the number of M_{RFF} increases. Our intuition is that the introduction of these links incorporates some redundant RFFs that the algorithm is able to prune.

5.3. Ablation study

The synthetic dataset employed in this study tries to mimic scenarios pertinent to the subsequent analyses. It comprises 500 samples with 100 features generated using a polynomial regression framework. The 16 output tasks are computed as $\mathbf{Y} = \nu(\mathbf{X})\mathbf{W} + \eta$, where $\nu(\cdot)$ is a polynomial function of degree 4 in which terms with power of 2 or higher of the same input feature are excluded. Besides, weight matrix $\mathbf{W}$ is randomly generated imposing that only 20 features of $\nu(\mathbf{X})$ are informative. In addition, η contains zero-mean Gaussian noise components, with a standard deviation of 2.

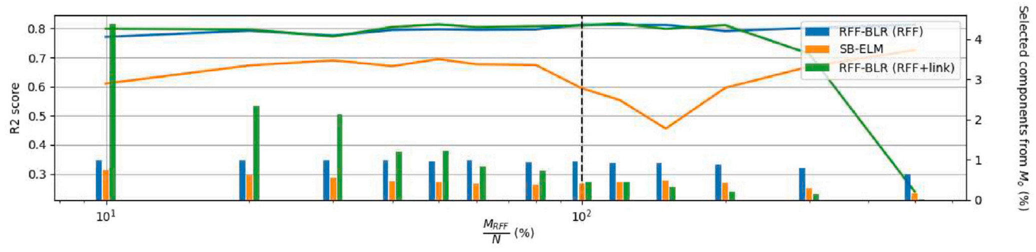
The proposed ablation study pursues to isolate the impact of each of the three main elements that define the RFF-BLR model: (1) addition or not of functional links; (2) Bayesian formulation, or non-Bayesian formulation, for automatic elimination of irrelevant components; (3) γ is optimised within the Bayesian framework or cross-validated. Table 5 shows the results of this analysis.

The study's findings highlight the pivotal role of the Bayesian formulation in automatically selecting relevant RFF components and/or functional links. While the inclusion of functional links leads to a slight performance enhancement, this effect is only noticeable when the network is optimised using the Bayesian formulation. Moreover, this case remarks how the inclusion of functional links tends to yield more compact solutions.

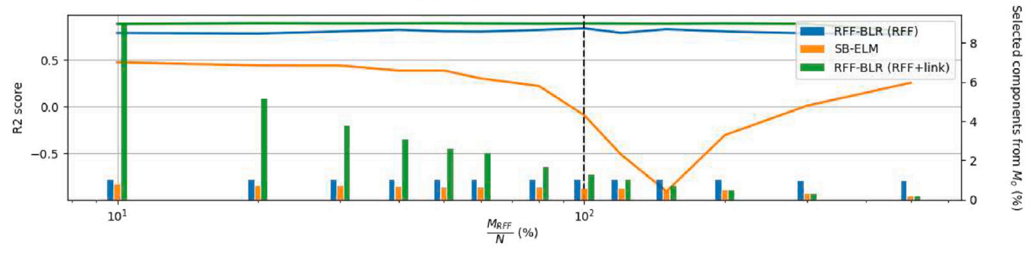
Regarding the automatic choice of γ, this study reveals that its optimisation does not directly improve accuracy, but represents a significant computational improvement by circumventing the parameter selection process through CV. However, in some cases, such as RFF-BLR with functional links and Bayesian optimisation, optimisation of γ under the Bayesian framework produces models that achieve an expected accuracy that seems dramatically higher than that of models with the same configuration, but with a cross-validated γ (R^2 of 0.79 vs. 0.38). This disparity remarks the importance of a good γ selection strategy.

Table 4
Results in multitask benchmark datasets. The values represent the mean and standard deviation of the R^2 scores obtained by each method for all tasks in each dataset and all CV folds. The second column specifies the non-linearity used, namely, RBF kernels (RBF), sigmoid function (sig), RFF projection (RFF) or a combination of these with the input links (link).

Model	Kernel	*atp1d*	*atp7d*	*oes97*	*oes10*	*edm*	*jura*	*wq*	*enb*	average
HI-VAE	(RBF)	0.72 ±0.07	0.42 ±0.12	0.50 ±0.22	0.66 ±0.10	0.34 ±0.11	0.54 ±0.07	0.07 ±0.02	0.91 ±0.01	0.52 ±0.09
FNN	(RBF)	0.80 ±0.10	0.64 ±0.12	0.60 ±0.16	0.77 ±0.09	0.10 ±0.34	0.35 ±0.19	0.13 ±0.03	**0.99 ±0.02**	0.55 ±0.13
MTFL	(RFF)	0.78 ±0.09	0.55 ±0.13	0.69 ±0.12	0.83 ±0.07	0.36 ±0.15	0.61 ±0.10	0.12 ±0.01	0.98 ±0.01	0.62 ±0.08
MMR	(RBF)	0.80 ±0.09	0.53 ±0.51	0.45 ±0.26	0.57 ±0.31	0.36 ±0.22	0.60 ±0.10	0.15 ±0.01	0.91 ±0.05	0.55 ±0.19
GOR-ELM	(sig)	0.81 ±0.09	0.63 ±0.14	0.68 ±0.14	0.82 ±0.05	0.26 ±0.30	0.64 ±0.11	0.12 ±0.03	0.98 ±0.01	0.62 ±0.11
SB-ELM	(RFF)	0.60 ±0.16	0.60 ±0.14	0.31 ±0.32	0.46 ±0.23	0.38 ±0.20	0.64 ±0.09	−0.02 ±0.07	0.97 ±0.01	0.49 ±0.15
RVFL	(sig+link)	0.74 ±0.15	0.53 ±0.16	0.60 ±0.23	0.81 ±0.07	0.18 ±0.32	0.58 ±0.11	0.12 ±0.03	0.98 ±0.01	0.55 ±0.13
edRVFL	(sig+link)	0.77 ±0.12	0.57 ±0.14	0.66 ±0.17	0.83 ±0.07	−0.63 ±0.55	0.42 ±0.24	**0.35 ±0.20**	0.90 ±0.02	0.49 ±0.19
NFRVFL	(sig+link)	0.69 ±0.28	0.53 ±0.14	0.60 ±0.16	0.77 ±0.17	0.06 ±0.27	0.35 ±0.20	−2.41 ±0.57	0.90 ±0.02	0.19 ±0.23
RFF-BLR	(RFF)	**0.83 ±0.06**	**0.74 ±0.15**	0.83 ±0.05	0.83 ±0.12	**0.49 ±0.14**	**0.85 ±0.07**	0.21 ±0.02	0.96 ±0.02	0.72 ±0.08
	(RFF+link)	0.81 ±0.07	0.73 ±0.08	**0.90 ±0.04**	**0.93 ±0.04**	0.47 ±0.12	0.83 ±0.08	0.21 ±0.02	0.95 ±0.04	**0.73 ±0.06**



(a) *atp1d* database.



(b) *oes97* database.

Fig. 2. Evaluation of the R^2 score as a function of the initial and final number of RFFs. This experiment compares the performance of SB-ELM (orange), RFF-BLR with functional links (green) and RFF-BLR without functional links (blue) averaged over 10-folds. The vertical dotted line shows the point where the initial number of RFFs is equal to the size of the training set. The bars represent the final number of components selected after pruning for each number of initial RFF features. These bars are measured with the right y-axis while the R^2 score is measured with the left y-axis.

5.4. Computational cost

The last part of this section is devoted to a more in-depth assessment of the computational efficiency of our approach. This evaluation consists in measuring the computational time needed to fit an instance of RFF-BLR in a synthetic dataset covering the different configurations defined in these ranges:

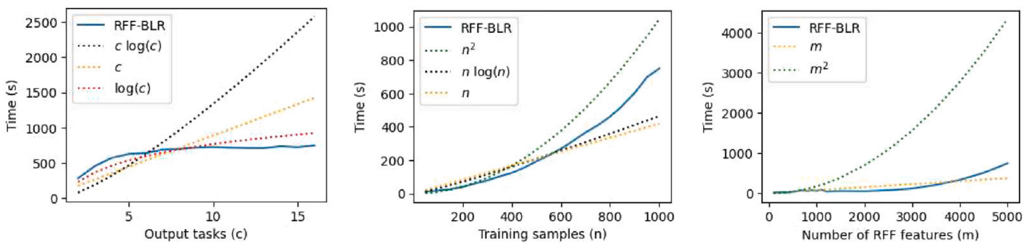
- For the number of output tasks, C, we have explored all the values in the range $[2, 16]$, fixing $N = 1,000$ and $M_{\text{RFF}} = 1.5\ N$.
- 50 values between 50 and 1000 for the size of the training set, N, fixing $C = 16$ and $M_{\text{RFF}} = 1.5\ N$.
- 20 values between 100 and 5000 for the number of RFFs, M_{RFF}, fixing $N = 1,000$ and $C = 16$.

We carried out these experiments five times with different initialisations and calculated the average results across these repetitions. and using a set number of features $D = 20$. The results of this study are shown in Fig. 3.

The plots show that the dependence of the computational cost on the number of output tasks is slightly less than logarithmic (Fig. 3(a)). Moreover, the dependence of this cost on the size of the training set turns out to be less than quadratic (Fig. 3(b)), and less than linear with respect to the number of RFFs up to 3500 (3.5 N) and less

Table 5
Results in the MTR synthetic dataset with 500 samples, 100 features and 16 output tasks. The values represent the mean and standard deviation of the R^2 scores obtained by each method for all tasks in a 10 fold evaluation, as well as the final number of RFF components and, if applicable, the number of functional links used in the hidden layer. All models are trained with $M_{RFF} = 675$. The dataset was evaluated using a 10-fold CV.

Configuration			R^2	Hidden layer dim.	
Links	Bayesian	γ optimisation		Links	RFF
✗	✗	✗	-0.52 ± 0.12	–	675
✓	✗	✗	0.27 ± 0.06	100	675
✗	✓	✗	0.38 ± 0.05	–	323 ± 3
✗	✗	✓	-2.58 ± 0.17	–	675
✓	✓	✗	0.82 ± 0.01	25 ± 1	45 ± 4
✓	✗	✓	0.34 ± 0.04	100	675
✗	✓	✓	0.79 ± 0.02	–	211 ± 14
✓	✓	✓	0.82 ± 0.01	26 ± 1	42 ± 52



(a) Analysis of the number of output tasks (C). (b) Analysis of the number of training samples (N). (c) Analysis of the number of RFF features (M_{RFF}).

Fig. 3. Evaluation of the computational cost of the model on a synthetic dataset.

than quadratic from then on (Fig. 3(c)). The reduction relative to the number of RFFs is especially notable, as the sparsity induced by the Bayesian framework allows a decrease in computational cost from a cubic dependency on M_{RFF} to sublinear. These results show that in practice the computational cost of fitting the model turns out to be much lower than the upper bounds of Table 2. These upper bounds predicted a cost linear with C, quadratic in N and cubic with M_{RFF}.

This behaviour means that our proposal is particularly efficient and robust when dealing with a large number of RFF components and output tasks. In addition, the fact that it is quadratic with N indicates that our model also outperforms common kernel methods that typically exhibit a cubic increase in computational cost with the size of the training set.

6. Conclusions

This paper has presented the RFF-BLR, a framework suited to learn accurate sparse MTR models. In essence, RFF-BLR consists in a randomised feedforward NN built on two fundamental pillars: a single hidden layer that implements an RFF approximation to a RBF kernel, and a Bayesian formulation that optimises the weights connecting the hidden and output layers. The RFF-based hidden layer inherits the robustness of kernel methods to serve a non-linear mapping common to all tasks. The Bayesian formulation controls the expressive power of the network by (1) controlling the convergence of the training in scenarios where the number of parameters significantly exceeds the size of the training dataset; (2) enforcing sparsity in the hidden layer, delivering solutions of constrained architecture complexity; and (3) optimising the kernel parameter to avoid expensive cross-validations.

The experimental results show that RFF-BLR achieves significantly better performance than state-of-the-art single hidden layer NN models with various non-linearities on several widely used MTR benchmark problems with small training datasets. This advantage is more noticeable in the scenarios with fewer samples, where the Bayesian nature of the proposal gives it the robustness necessary to achieve satisfactory learning with a very small dataset. Additionally, RFF-BLR also outperforms NNs with multiple hidden layers in these datasets. However, as expected, this advantage diminishes in problems with a larger number of samples, particularly when compared to multi-layer NN models.

From a practitioner's perspective, our method is a general purpose algorithm that can be successfully employed in a variety of real-world applications and domains with limited training data, where the Bayesian framework can offer tangible advantages in terms of generalisation capabilities. A particular application use case could be the prediction of the number of stocks traded in a particular time period in all the components of a financial index, such as the SP&500 or the NASDAQ, where each stock would correspond with a regression task and the observations would be measures of the technical indicators that capture market relevant information. A use case in the health domain could be the simultaneous modelling of a set of scores that help with the characterisation of a disease using as observations a cohort of patients and control subjects. Moreover, in the sustainability and environment protection domain another application would be the estimation of the energy production of the different aerogenerators within a same wind farm.

CRediT authorship contribution statement

Carlos Sevilla-Salcedo: Writing – original draft, Validation, Software, Methodology, Formal analysis. **Ascensión Gallardo-Antolín:** Writing – review & editing, Validation, Data curation. **Vanessa Gómez-Verdejo:** Writing – review & editing, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Emilio Parrado-Hernández:** Writing – review & editing, Supervision, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

Acknowledgement

The authors acknowledge support from the Spanish State Research Agency (MCIN/AEI/10.13039/5011000110) through project PID2020-115363RB-I00.

Appendix A. RFF-BLR variational inference updates

This section includes the complete mathematical derivation of the update rules of the model parameters using variational inference. For simplicity, we can start by determining the log-probability of the output data given the model parameters

$$\begin{aligned}
&\ln p\left(\mathbf{Y} \mid \boldsymbol{\Phi}_{\mathrm{RFF}}, \mathbf{W}, \tau, \mathbf{b}\right) \\
&= \sum_{\mathrm{n}=1}^{\mathrm{N}} \ln\left(\mathcal{N}\left(\boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:})\mathbf{W} + \mathbf{b}, (\tau)^{-1}\,\mathbf{I}_C\right)\right) \\
&= \sum_{\mathrm{n}=1}^{\mathrm{N}} \left(\frac{1}{2}\ln|(\tau)^{-1}\,\mathbf{I}_C| - \frac{\tau}{2}\left(\mathbf{y}_{n,:} - \boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:})\mathbf{W} - \mathbf{b}\right)\right. \\
&\quad \left.\left(\mathbf{y}_{n,:} - \boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:})\mathbf{W} - \mathbf{b}\right)^T\right) + \mathrm{const} \\
&= \frac{NC}{2}\ln(\tau) - \frac{\tau}{2}\sum_{\mathrm{n}=1}^{\mathrm{N}}\left(\mathbf{y}_{n,:}\mathbf{y}_{n,:}^\top - 2\mathbf{y}_{n,:}\mathbf{W}^\top\boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:}^\top) + \mathbf{b}\mathbf{b}^\top\right. \\
&\quad -2\mathbf{y}_{n,:}\mathbf{b}^\top + 2\boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:})\mathbf{W}\mathbf{b}^\top \\
&\quad \left.+\boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:})\mathbf{W}\mathbf{W}^\top\boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:}^\top)\right) + \mathrm{const},
\end{aligned} \tag{A.1}$$

where const include the terms without r.v.

A.1. Distribution of W

The approximate log probability of variable $\mathbf{W}$ is given by

$$\begin{aligned}
\ln\left(q^*(\mathbf{W})\right) &= \mathbb{E}_{\tau,\boldsymbol{\alpha}\mathbf{b}}\left[\ln\left(p\left(\mathbf{Y}, \mathbf{W}, \boldsymbol{\alpha}, \tau, \mathbf{b} \mid \boldsymbol{\Phi}_{\mathrm{RFF}}\right)\right)\right] \\
&= \mathbb{E}_{\tau,\mathbf{b}}\left[\ln\left(p\left(\mathbf{Y} \mid \boldsymbol{\Phi}_{\mathrm{RFF}}, \mathbf{W}, \tau, \mathbf{b}\right)\right)\right] \\
&\quad + \mathbb{E}_{\boldsymbol{\alpha}}\left[\ln\left(p\left(\mathbf{W} \mid \boldsymbol{\alpha}\right)\right)\right] + \mathrm{const},
\end{aligned} \tag{A.2}$$

where the first term is

$$\begin{aligned}
&\ln\left(p\left(\mathbf{Y} \mid \boldsymbol{\Phi}_{\mathrm{RFF}}, \mathbf{W}, \tau, \mathbf{b}\right)\right) \\
&= -\frac{\tau}{2}\sum_{\mathrm{n}=1}^{\mathrm{N}}\sum_{\mathrm{c}=1}^{\mathrm{C}}\left(-2y_{n,c}\mathbf{w}_{:,c}^\top\boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:}^\top) + 2\mathbf{w}_{:,c}^\top\boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:}^\top)\mathrm{b_c}\right. \\
&\quad \left.+\mathbf{w}_{:,c}^\top\boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:}^\top)\boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:})\mathbf{w}_{:,c}\right) + \mathrm{const} \\
&= \tau\sum_{\mathrm{c}=1}^{\mathrm{C}}\left(\mathbf{w}_{:,c}^\top\boldsymbol{\Phi}_{\mathrm{RFF}}^\top\mathbf{y}_{:,c} + \mathbf{w}_{:,c}^\top\boldsymbol{\Phi}_{\mathrm{RFF}}^\top\mathbb{1}_N\mathrm{b_c}\right. \\
&\quad \left.+\frac{1}{2}\mathbf{w}_{:,c}^\top\boldsymbol{\Phi}_{\mathrm{RFF}}^\top\boldsymbol{\Phi}_{\mathrm{RFF}}\mathbf{w}_{:,c}\right) + \mathrm{const},
\end{aligned} \tag{A.3}$$

where $\mathbb{1}_N$ is a column vector of ones of dimension N. Then, the second term is

$$\begin{aligned}
\ln\left(p\left(\mathbf{W} \mid \boldsymbol{\alpha}\right)\right) &= \sum_{\mathrm{m}=1}^{\mathrm{M}}\ln\left(p\left(\mathbf{w}_{m,:} \mid \alpha_m\right)\right) \\
&= \sum_{\mathrm{m}=1}^{\mathrm{M}}\sum_{\mathrm{c}=1}^{\mathrm{C}}\frac{\alpha_m}{2}\mathrm{w_{m,c}}^2 + \mathrm{const} \\
&= \frac{1}{2}\sum_{\mathrm{c}=1}^{\mathrm{C}}\mathbf{w}_{:,c}^\top diag(\boldsymbol{\alpha})\mathbf{w}_{:,c} + \mathrm{const}.
\end{aligned} \tag{A.4}$$

Then, by calculating the expectation, we get

$$\begin{aligned}
\ln\left(q^*(\mathbf{W})\right) = &\sum_{\mathrm{c}=1}^{\mathrm{C}}\left(\langle\tau\rangle\mathbf{w}_{:,c}^\top\boldsymbol{\Phi}_{\mathrm{RFF}}^\top(\mathbf{y}_{:,c} - \mathbb{1}_N\langle\mathrm{b_c}\rangle)\right. \\
&-\frac{1}{2}\mathbf{w}_{:,c}^\top(diag(\langle\boldsymbol{\alpha}\rangle) \\
&\left.+\langle\tau\rangle\boldsymbol{\Phi}_{\mathrm{RFF}}^\top\boldsymbol{\Phi}_{\mathrm{RFF}})\mathbf{w}_{:,c}\right) + \mathrm{const}.
\end{aligned} \tag{A.5}$$

Identifying terms, we see that the q distribution of the variable is

$$q^*(\mathbf{W}) = \prod_{\mathrm{c}=1}^{\mathrm{C}}\mathcal{N}\left(\mathbf{w}_{:,c} \mid \langle\mathbf{w}_{:,c}\rangle, \Sigma_{\mathbf{w}_{:,c}}\right), \tag{A.6}$$

where the covariance matrix is common for all output tasks, C, and can be expressed as

$$\Sigma_{\mathbf{W}}^{-1} = diag(\langle\boldsymbol{\alpha}\rangle) + \langle\tau\rangle\boldsymbol{\Phi}_{\mathrm{RFF}}^\top\boldsymbol{\Phi}_{\mathrm{RFF}}, \tag{A.7}$$

and mean

$$\langle\mathbf{W}\rangle = \langle\tau\rangle\Sigma_{\mathbf{W}}\boldsymbol{\Phi}_{\mathrm{RFF}}^\top\left(\mathbf{Y} - \mathbb{1}_N\langle\mathbf{b}\rangle\right), \tag{A.8}$$

where $\langle\mathbf{W}\rangle$ is a stacked version of $\langle\mathbf{w}_{:,c}\rangle$.

A.2. Distribution of α

The approximate distribution of $\boldsymbol{\alpha}$ follows

$$\begin{aligned}
\ln\left(q^*(\boldsymbol{\alpha})\right) = &\ \mathbb{E}_{\mathbf{W}}\left[\ln\left(p\left(\mathbf{Y}, \mathbf{W}, \boldsymbol{\alpha}, \tau, \mathbf{b} \mid \boldsymbol{\Phi}_{\mathrm{RFF}}\right)\right)\right] \\
= &\ \mathbb{E}_{\mathbf{W}}\left[\ln\left(p\left(\mathbf{W} \mid \boldsymbol{\alpha}\right)\right)\right] \\
&+ \mathbb{E}\left[\ln\left(p\left(\boldsymbol{\alpha}\right)\right)\right] + \mathrm{const},
\end{aligned} \tag{A.9}$$

where the first term corresponds to Eq. (A.4) and the second term is

$$\begin{aligned}
\mathbb{E}\left[\ln\left(p\left(\boldsymbol{\alpha}\right)\right)\right] &= \sum_{\mathrm{m}=1}^{\mathrm{M}}\left(\ln\left(p\left(\alpha_m\right)\right)\right) \\
&= \sum_{\mathrm{m}=1}^{\mathrm{M}}\left(-\mathrm{b}_{\alpha_\mathrm{m}}^0\alpha_m + \left(\mathrm{a}_{\alpha_\mathrm{m}}^0 - 1\right)\ln\left(\alpha_m\right)\right) + \mathrm{const}.
\end{aligned} \tag{A.10}$$

Then, joining both terms, we get

$$\begin{aligned}
\ln\left(q^*(\boldsymbol{\alpha})\right) = &\sum_{\mathrm{m}=1}^{\mathrm{M}}\left(\left(\frac{C}{2} + \mathrm{a}_{\alpha_\mathrm{m}}^0 - 1\right)\ln\left(\alpha_m\right)\right. \\
&\left.-\left(\mathrm{b}_{\alpha_\mathrm{m}}^0 + \frac{1}{2}\langle\mathbf{w}_{m,:}^\top\mathbf{w}_{m,:}\rangle\right)\alpha_m\right) + \mathrm{const}.
\end{aligned} \tag{A.11}$$

Therefore, the q distribution of $\boldsymbol{\alpha}$ is

$$q(\boldsymbol{\alpha}) = \prod_{m=1}^{M_{\mathrm{RFF}}}\Gamma\left(\alpha_m \mid \mathbf{a}_{\alpha_m}, \mathbf{b}_{\alpha_m}\right), \tag{A.12}$$

with the distribution parameters calculated as

$$\mathbf{a}_{\alpha_m} = \mathrm{a}_{\alpha_\mathrm{m}}^0 + \frac{C}{2}, \tag{A.13}$$

$$\mathbf{b}_{\alpha_m} = \mathrm{b}_{\alpha_\mathrm{m}}^0 + \frac{1}{2}\langle\mathbf{w}_{m,:}^\top\mathbf{w}_{m,:}\rangle. \tag{A.14}$$

A.3. Distribution of b

The distribution of variable $\mathbf{b}$ is given by

$$\begin{aligned}
\ln\left(q^*(\mathbf{b})\right) = &\ \mathbb{E}_{\mathbf{W},\tau}\left[\ln\left(p\left(\mathbf{Y}, \mathbf{W}, \boldsymbol{\alpha}, \tau, \mathbf{b} \mid \boldsymbol{\Phi}_{\mathrm{RFF}}\right)\right)\right] \\
= &\ \mathbb{E}_{\mathbf{W},\tau}\left[\ln\left(p\left(\mathbf{Y} \mid \boldsymbol{\Phi}_{\mathrm{RFF}}, \mathbf{W}, \tau, \mathbf{b}\right)\right)\right] \\
&+ \mathbb{E}\left[\ln\left(p\left(\mathbf{b}\right)\right)\right] + \mathrm{const},
\end{aligned} \tag{A.15}$$

where the effect of the prior of the bias is given by

$$\ln\left(p\left(\mathbf{b}\right)\right) = \ln\left(\mathcal{N}\left(\mathbf{0}, \mathbf{I}_C\right)\right) = -\frac{1}{2}\mathbf{b}\mathbf{b}^\top + \mathrm{const},$$

and the remaining term of the distribution can be calculated similarly to Eq. (A.3). Then, by calculating the expectation, we get

$$\begin{aligned}
\ln\left(q^*(\mathbf{b})\right) = &\sum_{\mathrm{n}=1}^{\mathrm{N}}\left(\langle\tau\rangle(\mathbf{y}_{n,:} - \boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:})\langle\mathbf{W}\rangle)\mathbf{b}^\top\right. \\
&\left.-\frac{1 + N\langle\tau\rangle}{2}\mathbf{b}\mathbf{b}^\top\right) + \mathrm{const}.
\end{aligned} \tag{A.16}$$

Once this expectation is calculated, we can determine that the distribution followed by the parameter is given by

$$q^*(\mathbf{b}) = \mathcal{N}\left(\mathbf{b} \mid \langle\mathbf{b}\rangle, \Sigma_{\mathbf{b}}\right), \tag{A.17}$$

where the variance is

$$\Sigma_{\mathbf{b}}^{-1} = (N\langle\tau\rangle + 1)\mathbf{I}_C, \tag{A.18}$$

and the mean is

$$\langle\mathbf{b}\rangle = \langle\tau\rangle \sum_{n=1}^{N} \left(\mathbf{y}_{n,:} - \boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:})\langle\mathbf{W}\rangle\right)\Sigma_{\mathbf{b}}. \tag{A.19}$$

A.4. Distribution of τ

Finally, the approximate distribution of τ is

$$\begin{aligned}\ln\left(q^*(\tau)\right) = \; & \mathbb{E}_{\mathbf{W},\mathbf{b}}\left[\ln\left(p\left(\mathbf{Y} \mid \boldsymbol{\Phi}_{\mathrm{RFF}}, \mathbf{W}, \tau, \mathbf{b}\right)\right)\right] \\ & + \mathbb{E}\left[\ln(p(\tau))\right] + \text{const.}\end{aligned} \tag{A.20}$$

We can calculate the expectation of Eq. (A.1), obtaining

$$\begin{aligned}\mathbb{E}_{\mathbf{W},\mathbf{b}}\left[\ln\left(p\left(\mathbf{Y} \mid \boldsymbol{\Phi}_{\mathrm{RFF}}, \mathbf{W}, \tau, \mathbf{b}\right)\right)\right] &= \frac{NC}{2}\ln(\tau) \\ - \frac{\tau}{2}\Bigg(\sum_{n=1}^{N}\sum_{c=1}^{C} y_{n,c}^2 &- 2\,\mathrm{Tr}\left\{\langle\mathbf{W}^\top\rangle\boldsymbol{\Phi}_{\mathrm{RFF}}^\top\mathbf{Y}\right\} \\ + \mathrm{Tr}\left\{\langle\mathbf{W}\mathbf{W}^\top\rangle\boldsymbol{\Phi}_{\mathrm{RFF}}^\top\boldsymbol{\Phi}_{\mathrm{RFF}}\right\} &- 2\sum_{n=1}^{N}\mathbf{y}_{n,:}\langle\mathbf{b}^\top\rangle \\ +2\sum_{n=1}^{N}\boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:})\langle\mathbf{W}\rangle\langle\mathbf{b}^\top\rangle &+ N\langle\mathbf{b}\mathbf{b}^\top\rangle\Bigg),\end{aligned} \tag{A.21}$$

and then the second term is

$$\mathbb{E}\left[\ln(p(\tau))\right] = \ln(p(\tau)) = -\beta_0^\tau\tau + \left(\alpha_0^\tau - 1\right)\ln(\tau) + \text{const.} \tag{A.22}$$

So, if we join both expectation elements and identify distribution terms, we see that the new distribution is

$$q^*(\tau) = \Gamma\left(\tau \mid a_\tau, b_\tau\right), \tag{A.23}$$

where the parameter a_τ is

$$a_\tau = \frac{NC}{2} + \mathrm{a}_\tau^0, \tag{A.24}$$

and the parameter b_τ can be expressed as

$$\begin{aligned}b_\tau = \; & \mathrm{b}_\tau^0 + \frac{1}{2}\sum_{n=1}^{N}\sum_{c=1}^{C} y_{n,c}^2 + \frac{1}{2}Tr\{\langle\mathbf{W}^\top\mathbf{W}\rangle\boldsymbol{\Phi}\boldsymbol{\Phi}_{\mathrm{RFF}}^\top\} \\ & - Tr\{\mathbf{Y}\langle\mathbf{W}^\top\rangle\boldsymbol{\Phi}_{\mathrm{RFF}}^\top\} - \sum_{n=1}^{N}\mathbf{y}_{n,:}\langle\mathbf{b}^\top\rangle \\ & + \sum_{n=1}^{N}\boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:})\langle\mathbf{W}\rangle\langle\mathbf{b}^\top\rangle + \frac{N}{2}\langle\mathbf{b}\mathbf{b}^\top\rangle.\end{aligned} \tag{A.25}$$

Appendix B. RFF-BLR lower bound

Here, we present the complete derivation of the lower bound of the model. We can calculate the changes in the lower bound as follows:

$$\begin{aligned}LB = \; & -\int q(\Theta)\ln\left(\frac{q(\Theta)}{p\left(\mathbf{Y}, \Theta \mid \boldsymbol{\Phi}_{\mathrm{RFF}}\right)}\right)d\Theta \\ = \; & \int q(\Theta)\ln\left(p\left(\mathbf{Y}, \Theta \mid \boldsymbol{\Phi}_{\mathrm{RFF}}\right)\right)d\Theta - \int q(\Theta)\ln(q(\Theta))\,d\Theta \\ = \; & \mathbb{E}_q\left[\ln\left(p\left(\mathbf{Y}, \Theta \mid \boldsymbol{\Phi}_{\mathrm{RFF}}\right)\right)\right] - \mathbb{E}_q\left[\ln(q(\Theta))\right].\end{aligned} \tag{B.1}$$

We will separately calculate the terms related to $\mathbb{E}_q\left[\ln\left(p\left(\mathbf{Y}, \Theta \mid \boldsymbol{\Phi}_{\mathrm{RFF}}\right)\right)\right]$ and the entropy of $q(\Theta)$.

B.1. Terms associated to $\mathbb{E}_q\left[\ln\left(p\left(\boldsymbol{Y}, \Theta \mid \boldsymbol{\Phi}_{\mathrm{RFF}}\right)\right)\right]$

This first term of the lower bound would be composed by the following terms:

$$\begin{aligned}&\mathbb{E}_q\left[\ln\left(p\left(\mathbf{Y}, \Theta \mid \boldsymbol{\Phi}_{\mathrm{RFF}}\right)\right)\right] = \\ &\quad \mathbb{E}_q\left[\ln(p(\mathbf{W} \mid \boldsymbol{\alpha}))\right] + \mathbb{E}_q\left[\ln(p(\boldsymbol{\alpha}))\right] \\ &\quad + \mathbb{E}_q\left[\ln\left(p\left(\mathbf{Y} \mid \mathbf{W}, \boldsymbol{\Phi}_{\mathrm{RFF}}, \mathbf{b}, \tau\right)\right)\right] + \mathbb{E}_q\left[\ln(p(\tau))\right] + \mathbb{E}_q\left[\ln(p(\mathbf{b}))\right].\end{aligned} \tag{B.2}$$

These are calculated as

$$\begin{aligned}\mathbb{E}_q\left[\ln(p(\mathbf{W} \mid \boldsymbol{\alpha}))\right] &= -\frac{MC}{2}\ln(2\pi) - \sum_{m=1}^{M}\left(a_{\alpha_m}\right) \\ &+ \frac{C}{2}\sum_{m=1}^{M}\left(\psi\left(a_{\alpha_m}\right) - \ln\left(b_{\alpha_m}\right)\right) + \beta_0\sum_{m=1}^{M}\left(\frac{a_{\alpha_m}}{b_{\alpha_m}}\right),\end{aligned} \tag{B.3}$$

where $\psi(x)$ represents the cumulative distribution function.

$$\begin{aligned}\mathbb{E}_q\left[\ln(p(\boldsymbol{\alpha}))\right] &= C\left(\alpha_0\ln\left(\beta_0\right) - \ln\left(\Gamma\left(\alpha_0\right)\right)\right) \\ &+ \sum_{m=1}^{M}\left(-\beta_0\frac{a_{\alpha_m}}{b_{\alpha_m}} + \left(\alpha_0 - 1\right)\left(\psi\left(a_{\alpha_m}\right) - \ln\left(b_{\alpha_m}\right)\right)\right)\end{aligned} \tag{B.4}$$

$$\begin{aligned}\mathbb{E}_q\left[\ln(p(\mathbf{W}, \boldsymbol{\alpha}))\right] &= \left(\frac{C}{2} + \alpha_0 - 1\right)\sum_{m=1}^{M}\left(\psi\left(a_{\alpha_m}\right) - \ln\left(b_{\alpha_m}\right)\right) \\ &- \frac{MC}{2}\ln(2\pi) + C\Big(\alpha_0\ln\left(\beta_0\right) - \ln\left(\Gamma\left(\alpha_0\right)\right)\Big) - \sum_{m=1}^{M}\left(a_{\alpha_m}\right),\end{aligned} \tag{B.5}$$

$$\begin{aligned}&\mathbb{E}_q\left[\ln\left(p\left(\mathbf{Y} \mid \mathbf{W}, \boldsymbol{\Phi}_{\mathrm{RFF}}, \mathbf{b}, \tau\right)\right)\right] \\ &= \; -\frac{NC}{2}\ln(2\pi) + \frac{C}{2}\left(\mathbb{E}_q\left[\ln(\tau)\right]\right) \\ &\quad - \frac{\langle\tau\rangle}{2}\sum_{n=1}^{N}\sum_{c=1}^{C}\Big(y_{n,c}y_{n,c} + y_{n,c}\langle\mathbf{w}_{:,c}^\top\rangle\boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:})^\top \\ &\quad - \frac{1}{2}\langle\mathbf{w}_{:,c}\mathbf{w}_{:,c}^\top\rangle\boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:})^\top\boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:}) - y_{n,c}\langle\mathrm{b_c}\rangle \\ &\quad - \boldsymbol{\phi}_{\mathrm{RFF}}(\mathbf{x}_{n,:})\langle\mathbf{w}_{:,c}\rangle\langle\mathrm{b_c}\rangle + \frac{1}{2}\langle\mathrm{b_c}\mathrm{b_c}\rangle\Big),\end{aligned} \tag{B.6}$$

$$\begin{aligned}\mathbb{E}_q\left[\ln(p(\tau))\right] &= \alpha_0^\tau\ln\left(\beta_0^\tau\right) - \ln\left(\Gamma\left(\alpha_0^\tau\right)\right) - \beta_0^\tau\frac{a_\tau}{b_\tau} \\ &+ \left(\alpha_0^\tau - 1\right)\Big(\psi\left(a_\tau\right) - \ln\left(b_\tau\right)\Big),\end{aligned} \tag{B.7}$$

$$\mathbb{E}_q\left[\ln(p(\mathbf{b}))\right] = -\frac{C}{2}\ln(2\pi) - \frac{1}{2}\langle\mathbf{b}\mathbf{b}^\top\rangle. \tag{B.8}$$

B.2. Terms of entropy of $q(\Theta)$

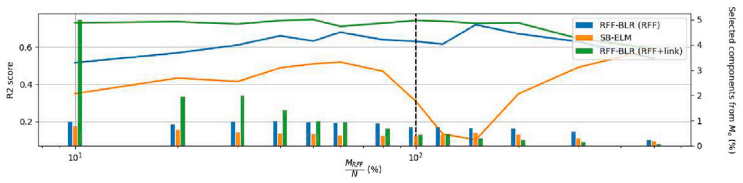
The second term of the lower bound, the entropy, can be calculated as

$$\begin{aligned}\mathbb{E}_q\left[\ln(q(\Theta))\right] = \; & \mathbb{E}_q\left[\ln(q(\mathbf{W}))\right] + \mathbb{E}_q\left[\ln(q(\boldsymbol{\alpha}))\right] \\ = \; & +\mathbb{E}_q\left[\ln(q(\tau))\right] + \mathbb{E}_q\left[\ln(q(\mathbf{b}))\right],\end{aligned} \tag{B.9}$$

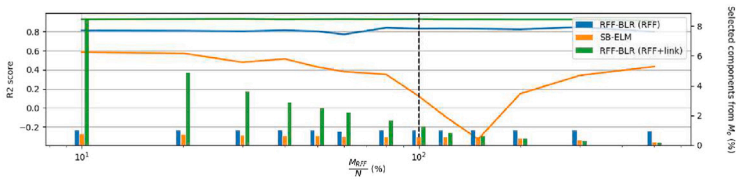
where we can now determine the entropy of each model parameter having

$$\mathbb{E}_q\left[\ln(q(\mathbf{W}))\right] = \frac{MC}{2}\ln(2\pi e) + \frac{M}{2}\ln|\Sigma_{\mathbf{W}}|, \tag{B.10}$$

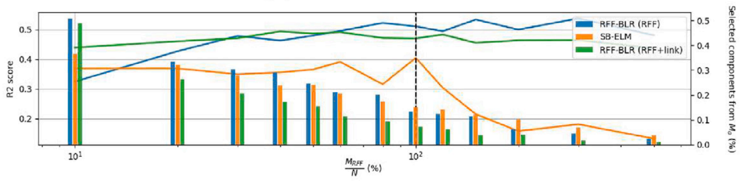
$$\begin{aligned}\mathbb{E}_q\left[\ln(q(\boldsymbol{\alpha}))\right] = \sum_{m=1}^{M}\Big(& a_{\alpha_m} + \ln\left(\Gamma\left(a_{\alpha_m}\right)\right) \\ & - \left(1 - a_{\alpha_m}\right)\psi\left(a_{\alpha_m}\right) - \ln\left(b_{\alpha_m}\right)\Big),\end{aligned} \tag{B.11}$$



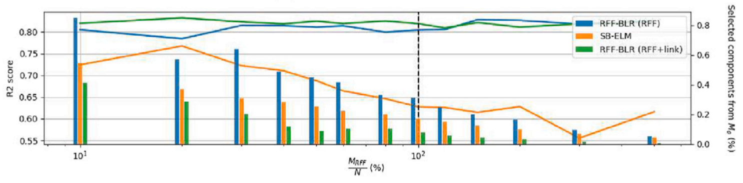
(a) *atp7d* database.



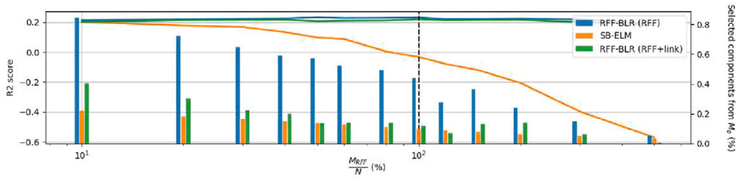
(b) *oes10* database.



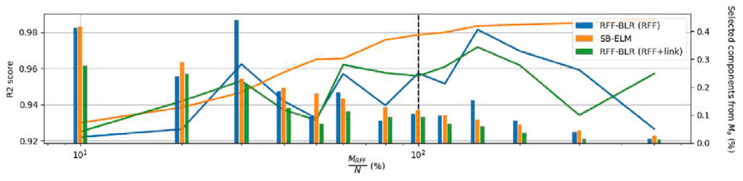
(c) *edm* database.



(d) *jura* database.



(e) *wq* database.



(f) *enb* database.

Fig. C.4. Evaluation of the R^2 score as a function of the initial and final number of RFFs. This experiment compares the performance of SB-ELM (orange) and RFF-BLR with and without links (green and blue) averaged over 10-folds. The vertical dotted line shows the point where the initial number of RFFs is equal to the size of the training set. The bars represent the final number of features selected after pruning for each number of initial RFF features. These bars are measured with the right y-axis while the R^2 score is measured with the left y-axis.

$$\begin{aligned}&\mathbb{E}_q\left[\ln\left(q\left(\tau\right)\right)\right]\\ =\quad & a_\tau + \ln\left(\Gamma\left(\alpha_0^\tau\right)\right) - \left(1-\alpha_0^\tau\right)\left(\psi\left(\alpha_0^\tau\right) - \ln\left(\beta_0^\tau\right)\right),\end{aligned} \tag{B.12}$$

$$\mathbb{E}_q\left[\ln\left(q\left(\mathbf{b}\right)\right)\right] = \frac{C}{2}\ln\left(2\pi e\right) + \frac{1}{2}\ln\mid \Sigma_{\mathbf{b}}\mid. \tag{B.13}$$

B.3. Complete lower bound

Finally, if we combine both terms, Eq. (B.2) and (B.9), we get that the complete lower bound is

$$\begin{aligned}LB = &-\left(\frac{C}{2}+\alpha_0-1\right)\sum_{m=1}^{M}\left(\ln\left(b_{\alpha_m}\right)\right)\\ &-\left(\alpha_0^\tau-1\right)\ln\left(b_\tau\right)-\frac{1}{2}\langle\mathbf{b}\mathbf{b}^\top\rangle-\beta_0^\tau\frac{a_\tau}{b_\tau}+\frac{C}{2}\left(\mathbb{E}_q\left[\ln\left(\tau\right)\right]\right)\\ &-\frac{\langle\tau\rangle}{2}\sum_{n=1}^{N}\sum_{c=1}^{C}\Big(\frac{1}{2}\langle \mathrm{b_c b_c}\rangle + y_{n,c}\langle\mathbf{w}_{:,c}^\top\rangle\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}_{n,:}^\top)-y_{n,c}\langle\mathrm{b_c}\rangle\\ &-\frac{1}{2}\langle\mathbf{w}_{:,c}\mathbf{w}_{:,c}^\top\rangle\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}_{n,:}^\top)\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}_{n,:})-\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}_{n,:})\langle\mathbf{w}_{:,c}\rangle\langle\mathrm{b_c}\rangle\Big)\\ &-\frac{C}{2}\ln\mid\Sigma_{\mathbf{W}}\mid-\frac{1}{2}\ln\mid\Sigma_{\mathbf{b}}\mid+\sum_{m=1}^{M}\ln\left(b_{\alpha_m}\right)+\ln\left(b_\tau\right)+\text{const},\end{aligned} \tag{B.14}$$

where we can use Eq. (A.25) to simplify the lower bound

$$\begin{aligned}LB = &-\left(\frac{C}{2}+\alpha_0-1\right)\sum_{m=1}^{M}\left(\ln\left(b_{\alpha_m}\right)\right)\\ &-\left(\frac{C}{2}+\alpha_0^\tau-1\right)\ln\left(b_\tau\right)-\frac{1}{2}\langle\mathbf{b}\mathbf{b}^\top\rangle\\ &-\frac{C}{2}\ln\mid\Sigma_{\mathbf{W}}\mid-\frac{1}{2}\ln\mid\Sigma_{\mathbf{b}}\mid+\sum_{m=1}^{M}\ln\left(b_{\alpha_m}\right)+\ln\left(b_\tau\right)+\text{const}\end{aligned} \tag{B.15}$$

B.4. Lower bound dependent on $\boldsymbol{\Phi}_{RFF}$

To maximise the lower bound to obtain the optimum γ value we need the terms dependent on $\boldsymbol{\Phi}_{\text{RFF}}$ from Eq. (B.14), obtaining

$$\begin{aligned}LB = \quad&\langle\tau\rangle\sum_{n=1}^{N}\sum_{c=1}^{C}\Big(y_{n,c}\langle\mathbf{w}_{:,c}^\top\rangle\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}_{n,:})^\top-\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}_{n,:})\langle\mathbf{w}_{:,c}\rangle\langle\mathrm{b_c}\rangle\\ &-\frac{1}{2}\langle\mathbf{w}_{:,c}\mathbf{w}_{:,c}^\top\rangle\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}_{n,:})^\top\boldsymbol{\phi}_{\text{RFF}}(\mathbf{x}_{n,:})\Big).\end{aligned} \tag{B.16}$$

Appendix C. Sparsity analysis

In this section, we present the graphs obtained for the 6 databases that were not included in the Results section. Fig. C.4 includes the results obtained when running both our proposal and SB-ELM for different numbers of initial RFF.

References

Aho, T., Ženko, B., Džzeroski, S., Elomaa, T., & Brodley, C. (2012). Multi-target regression with rule ensembles. *Journal of Machine Learning Research*, *13*(8), 2367–2407.

Argyriou, A., Evgeniou, T., & Pontil, M. (2006). Multi-task feature learning. In B. Schölkopf, J. Platt, & T. Hoffman (Eds.), *Advances in neural information processing systems: vol. 19*, MIT Press.

Bishop, C. (2006). Pattern recognition and machine learning. *Information science and statistics*, Springer, ISBN: 9780387310732, URL https://books.google.es/books?id=kTNoQgAACAAJ.

Borchani, H., Varando, G., Bielza, C., & Larranaga, P. (2015). A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *5*(5), 216–233. http://dx.doi.org/10.1002/widm.1157.

Brouard, C., Szafranski, M., & d'Alché-Buc, F. (2016). Input output kernel regression: Supervised and semi-supervised structured output prediction with operator-valued kernels. *Journal of Machine Learning Research*, *17*.

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*(2), 121–167. http://dx.doi.org/10.1023/A:1009715923555.

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*(1), 41–75. http://dx.doi.org/10.1023/A:1007379606734.

da Silva, B. L. S., Inaba, F. K., Salles, E. O. T., & Ciarelli, P. M. (2020). Outlier robust extreme machine learning for multi-target regression. *Expert Systems with Applications*, *140*, Article 112877. http://dx.doi.org/10.1016/j.eswa.2019.112877.

Dinuzzo, F., Ong, C. S., Pillonetto, G., & Gehler, P. V. (2011). Learning output kernels with block coordinate descent. In *8th international conference on machine learning* (pp. 49–56).

Džeroski, S., Demšar, D., & Grbović, J. (2000). Predicting chemical parameters of river water quality from bioindicator data. *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies*, *13*(1), 7–17. http://dx.doi.org/10.1023/A:1008323212047.

Emambakhsh, M., Bay, A., & Vazquez, E. (2019). Convolutional recurrent predictor: Implicit representation for multi-target filtering and tracking. *IEEE Transactions on Signal Processing*, *67*(17), 4545–4555. http://dx.doi.org/10.1109/TSP.2019.2931170.

Farlessyost, W., Grant, K.-R., Davis, S. R., Feil-Seifer, D., & Hand, E. M. (2021). The effectiveness of multi-label classification and multi-output regression in social trait recognition. *Sensors*, *21*(12), http://dx.doi.org/10.3390/s21124127.

Gao, R., Li, R., Hu, M., Suganthan, P. N., & Yuen, K. F. (2023). Online dynamic ensemble deep random vector functional link neural network for forecasting. *Neural Networks*, *166*, 51–69.

Goncalves, A., Ray, P., Soper, B., Widemann, D., Nygård, M., Nygård, J. F., et al. (2019). Bayesian multitask learning regression for heterogeneous patient cohorts. *Journal of Biomedical Informatics*, *100*, Article 100059. http://dx.doi.org/10.1016/j.yjbinx.2019.100059.

Hartman, E. J., Keeler, J. D., & Kowalski, J. M. (1990). Layered Neural Networks with Gaussian Hidden Units as Universal Approximations. *Neural Computation*, *2*(2), 210–215. http://dx.doi.org/10.1162/neco.1990.2.2.210.

He, J., Li, X., Liu, P., Wang, L., Zhou, H., Wang, J., et al. (2023). Ensemble deep random vector functional link for self-supervised direction-of-arrival estimation. *Engineering Applications of Artificial Intelligence*, *120*, Article 105831.

Huang, G.-B. (2015). What are extreme learning machines? Filling the gap between Frank Rosenblatt's Dream and John von Neumann's puzzle. *Cognitive Computation*, *7*(3), 263–278. http://dx.doi.org/10.1007/s12559-015-9333-0.

Huang, G.-B., Bai, Z., Kasun, L. L. C., & Vong, C. M. (2015). Local receptive fields based extreme learning machine. *IEEE Computational Intelligence Magazine*, *10*(2), 18–29. http://dx.doi.org/10.1109/MCI.2015.2405316.

Huang, G.-B., Chen, L., Siew, C. K., et al. (2006). Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Transactions on Neural Networks*, *17*(4), 879–892. http://dx.doi.org/10.1109/TNN.2006.875977.

Huang, G.-B., Zhu, Q.-Y., & Siew, C.-K. (2004). Extreme learning machine: a new learning scheme of feedforward neural networks. In *IEEE international joint conference on neural networks, vol. 2* (IJCNN 2004), (pp. 985–990). http://dx.doi.org/10.1109/IJCNN.2004.1380068.

Husmeier, D. (1999). Random vector functional link (RVFL) networks. In *Neural networks for conditional probability estimation: forecasting beyond point predictions* (pp. 87–97). Springer London, http://dx.doi.org/10.1007/978-1-4471-0847-4_6.

Inaba, F. K., Teatini Salles, E. O., Perron, S., & Caporossi, G. (2018). DGR-ELM–distributed generalized regularized ELM for classification. *Neurocomputing*, *275*, 1522–1530. http://dx.doi.org/10.1016/j.neucom.2017.09.090.

Jacob, L., Vert, J.-P., & Bach, F. (2008). Clustered multi-task learning: A convex formulation. In D. Koller, D. Schuurmans, Y. Bengio, & L. Bottou (Eds.), *Advances in neural information processing systems: vol. 21*, (pp. 1–8). Curran Associates, Inc..

Jalali, A., Sanghavi, S., Ruan, C., & Ravikumar, P. (2010). A dirty model for multi-task learning. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, & A. Culotta (Eds.), *Advances in neural information processing systems: vol. 23*, Curran Associates, Inc..

Karalič, A., & Bratko, I. (1997). First order regression. *Machine Learning*, *26*(2–3), 147–176. http://dx.doi.org/10.1023/A:1007365207130.

Ketu, S., & Mishra, P. K. (2021). Enhanced Gaussian process regression-based forecasting model for COVID-19 outbreak and significance of IoT for its detection. *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies*, *51*, 1492–1512. http://dx.doi.org/10.1007/s10489-020-01889-9.

Khan, G. A., Hu, J., Li, T., Diallo, B., & Du, S. (2023). Multi-view subspace clustering for learning joint representation via low-rank sparse representation. *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies*, *53*(19), 22511–22530. http://dx.doi.org/10.1007/s10489-023-04716-z.

Khan, G. A., Hu, J., Li, T., Diallo, B., & Wang, H. (2023). Multi-view clustering for multiple manifold learning via concept factorization. *Digital Signal Processing*, *140*, Article 104118. http://dx.doi.org/10.1016/j.dsp.2023.104118.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd international conference on learning representations*.

Kocev, D., Vens, C., Struyf, J., & Džeroski, S. (2007). Ensembles of multi-objective decision trees. In *18th European conference on machine learning* (pp. 624–631). http://dx.doi.org/10.1007/978-3-540-74958-5_61.

Li, R., Gao, R., Yuan, L., Suganthan, P., Wang, L., & Sourina, O. (2023). An enhanced ensemble deep random vector functional link network for driver fatigue recognition. *Engineering Applications of Artificial Intelligence*, *123*, Article 106237. http://dx.doi.org/10.1016/j.engappai.2023.106237.

Li, C., Wei, F., Dong, W., Wang, X., Liu, Q., & Zhang, X. (2019). Dynamic structure embedded online multiple-output regression for streaming data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *41*(2), 323–336. http://dx.doi.org/10.1109/TPAMI.2018.2794446.

Liu, H., Wang, L., & Zhao, T. (2015). Calibrated multivariate regression with application to neural semantic basis discovery. *Journal of Machine Learning Research*, *16*, 1579–1606.

Lozano, A. C., & Swirszcz, G. (2012). Multi-level lasso for sparse multi-task regression. In *29th international conference on machine learning* (pp. 595–602).

Luo, J., Vong, C.-M., & Wong, P.-K. (2013). Sparse Bayesian extreme learning machine for multi-classification. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(4), 836–843. http://dx.doi.org/10.1109/TNNLS.2013.2281839.

Ma, X., Kundu, S., & for the Alzheimer's Disease Neuroimaging Initiative (2024). Multi-task learning with high-dimensional noisy images. *Journal of the American Statistical Association*, *119*(545), 650–663. http://dx.doi.org/10.1080/01621459.2022.2140052.

Malik, A. K., Gao, R., Ganaie, M., Tanveer, M., & Suganthan, P. N. (2023). Random vector functional link network: recent developments, applications, and future directions. *Applied Soft Computing*, *143*, Article 110377. http://dx.doi.org/10.1016/j.asoc.2023.110377.

Martínez-Martínez, J. M., Escandell-Montero, P., Soria-Olivas, E., Martín-Guerrero, J. D., Magdalena-Benedito, R., & Gómez-Sanchis, J. (2011). Regularized extreme learning machine for regression problems. *Neurocomputing*, *74*(17), 3716–3721. http://dx.doi.org/10.1016/j.neucom.2011.06.013.

Masmoudi, S., Elghazel, H., Taieb, D., Yazar, O., & Kallel, A. (2020). A machine-learning framework for predicting multiple air pollutants' concentrations via multi-target regression and feature selection. *Science of the Total Environment*, *715*, Article 136991. http://dx.doi.org/10.1016/j.scitotenv.2020.136991.

Moyano, J. M., Reyes, O., Fardoun, H. M., & Ventura, S. (2021). Performing multi-target regression via gene expression programming-based ensemble models. *Neurocomputing*, *432*, 275–287. http://dx.doi.org/10.1016/j.neucom.2020.12.060.

Murtagh, F. (1991). Multilayer perceptrons for classification and regression. *Neurocomputing*, *2*(5–6), 183–197. http://dx.doi.org/10.1016/0925-2312(91)90023-5.

Nazabal, A., Olmos, P. M., Ghahramani, Z., & Valera, I. (2020). Handling incomplete heterogeneous data using VAEs. *Pattern Recognition*, *107*, Article 107501. http://dx.doi.org/10.1016/j.patcog.2020.107501.

Neal, R. M. (2002). *Bayesian learning for neural networks*. New York, NY: Springer-Verlag, ISBN: 978-0-387-94724-2.

Pao, Y.-H., & Takefuji, Y. (1992). Functional-link net computing: theory, system architecture, and functionalities. *Computer*, *25*(5), 76–79. http://dx.doi.org/10.1109/2.144401.

Rahimi, A., & Recht, B. (2007). Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer, & S. Roweis (Eds.), *Advances in neural information processing systems: vol. 20*, (pp. 1–8). Curran Associates, Inc..

Rai, P., Kumar, A., & Daume, H. (2012). Simultaneously leveraging output and task structures for multiple-output regression. In F. Pereira, C. Burges, L. Bottou, & K. Weinberger (Eds.), *Advances in neural information processing systems: vol. 25*, (pp. 1–9). Curran Associates, Inc..

Rasmussen, C. E. (2003). Gaussian processes in machine learning. In *Advanced lectures on machine learning: ML summer schools 2003* (pp. 63–71). Springer, http://dx.doi.org/10.1007/978-3-540-28650-9_4.

Rothman, A. J., Levina, E., & Zhu, J. (2010). Sparse multivariate regression with covariance estimation. *Journal of Computational and Graphical Statistics*, *19*(4), 947–962. http://dx.doi.org/10.1198/jcgs.2010.09188.

Ruder, S. (2017). An overview of multi-task learning in deep neural networks. http://dx.doi.org/10.48550/arXiv.1706.05098, arXiv:1706.05098.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). *Learning internal representations by error propagation: Technical report*, California Univ San Diego La Jolla Inst for Cognitive Science.

Sajid, M., Malik, A., Tanveer, M., & Suganthan, P. (2024). Neuro-fuzzy random vector functional link neural network for classification and regression problems. *IEEE Transactions on Fuzzy Systems*, *32*(5), 2738–2749. http://dx.doi.org/10.1109/TFUZZ.2024.3359652.

Sajid, M., Tanveer, M., & Suganthan, P. N. (2024). Ensemble deep random vector functional link neural network based on fuzzy inference system. *IEEE Transactions on Fuzzy Systems*, http://dx.doi.org/10.1109/TFUZZ.2024.3411614.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. Cambridge, MA: MIT Press, ISBN: 9780262256933.

Shalev-Shwartz, S., Singer, Y., Srebro, N., & Cotter, A. (2011). Pegasos: primal estimated sub-gradient solver for SVM. *Mathematical Programming*, *127*, 3–30. http://dx.doi.org/10.1007/s10107-010-0420-4.

Shi, Q., Katuwal, R., Suganthan, P., & Tanveer, M. (2021). Random vector functional link neural network based ensemble deep learning. *Pattern Recognition*, *117*, Article 107978. http://dx.doi.org/10.1016/j.patcog.2021.107978.

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*(3), 199–222. http://dx.doi.org/10.1023/B:STCO.0000035301.49549.88.

Soria-Olivas, E., Gomez-Sanchis, J., Martin, J. D., Vila-Frances, J., Martinez, M., Magdalena, J. R., et al. (2011). BELM: Bayesian extreme learning machine. *IEEE Transactions on Neural Networks*, *22*(3), 505–509. http://dx.doi.org/10.1109/TNN.2010.2103956.

Spyromitros-Xioufis, E., Tsoumakas, G., Groves, W., & Vlahavas, I. (2016). Multi-target regression via input space expansion: treating targets as inputs. *Machine Learning*, *104*(1), 55–98. http://dx.doi.org/10.1007/s10994-016-5546-z.

Suganthan, P. N., & Katuwal, R. (2021). On the origins of randomization-based feedforward neural networks. *Applied Soft Computing*, [ISSN: 1568-4946] *105*, Article 107239. http://dx.doi.org/10.1016/j.asoc.2021.107239.

Tan, C., Chen, S., Ji, G., & Geng, X. (2022). Multilabel distribution learning based on multioutput regression and manifold learning. *IEEE Transactions on Cybernetics*, *52*(6), 5064–5078. http://dx.doi.org/10.1109/TCYB.2020.3026576.

Tu, Y.-H., Du, J., Gao, T., & Lee, C.-H. (2020). A multi-target SNR-progressive learning approach to regression based speech enhancement. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *28*, 1608–1619. http://dx.doi.org/10.1109/TASLP.2020.2996503.

Vovk, V. (2013). Kernel ridge regression. In B. Schölkopf, Z. Luo, & V. Vovk (Eds.), *Empirical inference* (pp. 105–116). Berlin, Heidelberg: Springer, http://dx.doi.org/10.1007/978-3-642-41136-6_11.

Xiong, T., Bao, Y., & Hu, Z. (2014). Multiple-output support vector regression with a firefly algorithm for interval-valued stock price index forecasting. *Knowledge-Based Systems*, *55*, 87–100. http://dx.doi.org/10.1016/j.knosys.2013.10.012.

Yuan, M., & Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society. Series B. Statistical Methodology*, *68*(1), 49–67. http://dx.doi.org/10.1111/j.1467-9868.2005.00532.x.

Zezario, R. E., wei Fu, S., Chen, F., Fuh, C.-S., Wang, H.-M., & Tsao, Y. (2022). MTI-Net: A Multi-Target Speech Intelligibility Prediction Model. In *Interspeech 2022* (pp. 5463–5467). http://dx.doi.org/10.21437/Interspeech.2022-10828.

Zhen, X., Yu, M., He, X., & Li, S. (2018). Multi-target regression via robust low-rank learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(2), 497–504. http://dx.doi.org/10.1109/TPAMI.2017.2688363.

Zhen, X., Yu, M., Zheng, F., Nachum, I. B., Bhaduri, M., Laidley, D., et al. (2018). Multitarget sparse latent regression. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(5), 1575–1586. http://dx.doi.org/10.1109/TNNLS.2017.2651068.

Zhen, X., Zhang, H., Islam, A., Bhaduri, M., Chan, I., & Li, S. (2017). Direct and simultaneous estimation of cardiac four chamber volumes by multioutput sparse regression. *Medical Image Analysis*, *36*, 184–196. http://dx.doi.org/10.1016/j.media.2016.11.008.

Zhou, Q., & Zhao, Q. (2015). Flexible clustered multi-task learning by learning representative tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *38*(2), 266–278. http://dx.doi.org/10.1109/TPAMI.2015.2452911.