Fast multi-output relevance vector regression

Youngmin Ha [a] and Hai Zhang [b]

[a]Adam Smith Business School, University of Glasgow, UK; [b]Department of Accounting & Finance, Strathclyde Business School, UK

(*First draft: April 14, 2017; this draft: April 9, 2019*)

This paper has applied the matrix Gaussian distribution of the likelihood function of the complete data set to reduce time complexity of multi-output relevance vector regression from $O\left(VM^3\right)$ to $O\left(V^3+M^3\right)$, where V and M are the number of output dimensions and basis functions respectively and $V < M$. Our experimental results demonstrate that the proposed method is more competitive and faster than the existing methods like Thayananthan *et al.* (2008). Its computational efficiency and accuracy can be attributed to the different model specifications of the likelihood of the data, as the existing method expresses the likelihood of the training data as the product of Gaussian distributions whereas the proposed method expresses it as the matrix Gaussian distribution.

Keywords: Relevance vector regression; Machine learning; Sparse Bayesian learning

1. Introduction

When it comes to multi-input nonparametric nonlinear regression or classification, normally the following three methods can be considered: support vector machine (SVM), relevance vector machine (RVM), and Gaussian process (GP) regression and classification.

SVM, one of the popular machine learning tools, was by Cortes and Vapnik (1995). SVM uses kernel trick and makes the classification and regression computationally efficient for multidimensional inputs. However, its disadvantage is that a user needs to choose a proper value of the error/margin trade-off parameter 'C' (the proper value can be found by k-fold cross-validation).

RVM, firstly invented by Tipping (2001),[1] avoids estimating the error/margin trade-off parameter 'C' of SVM and the additional insensitivity parameter 'ε' (Vapnik 2000), or 'ν' (Schölkopf *et al.* 2000) in regression, which significantly reduces the computation time. Moreover, RVM allows probabilistic predictions (i.e. both mean and variance of a Gaussian distribution) although SVM predicts only mean values (the error bar estimation of SVM is possible with additional computation (Gao *et al.* 2002, Chu *et al.* 2004)).

Similar to RVM, GP regression or classification, invented by Gibbs (1997), also avoids estimating 'C' and the additional parameter of regression 'ε' or 'ν'. Furthermore, the predictive variance of GP regression or classification changes over an input vector $\mathbf{x}_*$: the predictive variance is smaller at the denser region of training samples, while the predictive variance of RVM is almost constant over $\mathbf{x}_*$.

Support vector regression (SVR), relevance vector regression (RVR), and GP regression (GPR) aforementioned suit only for multi-input but single-output regression, and they have been extended to multi-input and multi-output (MIMO) regression which is more appropriate to model correlated outputs: multi-output SVR (Pérez-Cruz *et al.* 2002, Vazquez and Walter 2003, Tuia *et al.* 2011),

CONTACT Hai Zhang. Email: hai.zhang@strath.ac.uk

[1] For a detailed explanation, please go to `http://static1.squarespace.com/static/58851af9ebbd1a30e98fb283/t/58902f4a6b8f5ba2ed9d3bfe/1485844299331/RVM+Explained.pdf`.

multi-output RVR (Thayananthan 2005, Thayananthan *et al.* 2008), and multi-output GPR (Boyle and Frean 2004, Bonilla *et al.* 2007, Alvarez and Lawrence 2009).

Multi-output regression methods have been pervasively applied in time series prediction: e.g. forecasting an interval-valued stock price index series over short and long horizons by using multi-output SVR (Xiong *et al.* 2014), multi-period-ahead forecasting of agricultural commodity prices by using multi-output RVR (Ticlavilca *et al.* 2010), and by using multi-output GPR (Cheng *et al.* 2018). Another key application is classification, which could be potentially applied to different disciplines from disease classification in health service section to fraud prevention and detection in financial sector.

The existing multi-output relevance vector regression (henceforth MRVR) algorithm proposed by Thayananthan (2005, Chapter 6), Thayananthan *et al.* (2008) uses the Bayes' theorem and the kernel trick to perform MIMO nonparametric nonlinear regression. However, the limitation of its low computational efficiency makes it impracticable for large output dimensions. Therefore, we have proposed a faster and more practicable algorithm which uses matrix normal distribution to model correlated outputs instead of multivariate normal distribution adopted by existing algorithms like (Thayananthan 2005, Thayananthan *et al.* 2008). By doing this, the proposed algorithm significantly reduces time complexity of multi-output relevance vector regression from $O\left(VM^3\right)$ to $O\left(V^3+M^3\right)$, which is one of the main contribution of this paper. Further, our numerical results through Monte Carlo simulation in Section 5 demonstrate the overall superiority of the proposed MRVR algorithm over the existing ones in terms of accuracy and computation time.

The rest of the paper is structured as follows. Section 2 lists related work of nonparametric & nonlinear regression or classification. Section 3 and Section 4 describe the existing and proposed algorithms of MRVR, respectively. Section 5 presents the main experimental results by using MATLAB, and finally Section 6 concludes.

2. Literature review

The following subsections list the existing research which has been conducted to reduce the computation time of single-output support vector machine (SVM), relevance vector machine (RVM), and Gaussian process (GP) regression respectively.

2.1. *Single-output SVM*

Guo and Zhang (2007) reduced SVR training time by reducing the number of training samples. Their method consists of the two steps. Firstly, it extracts samples which are more likely to be support vectors from a full training set befor performing SVR training, based on the heuristic observation that the target value of support vector is usually a local extremum or near extremum. Secondly, the extracted samples are used to train a SVR machine. Their simulation results show its computational efficiency. In particular, the computation time of k-fold cross validation of SVR for a large data set can be reduced greatly.

Catanzaro *et al.* (2008) accelerated SVM computation by using both faster sequential algorithms and parallel computation on a graphics processing unit (GPU). To be specific, a sequential minimal optimisation algorithm is used to solve the quadratic programming problem of SVM, and a GPU, whose role has changed from the manipulation of computer graphics and image processing to general-purpose programming, is employed for high throughput floating-point computation.

Chang and Lin (2011) made fast and efficient C++ software of SVM. They reduced the computation time to minimise SVM quadratic programming problems since the quadratic programming is the most time consuming part of SVM. To be specific, shrinking and caching techniques are used. The shrinking technique tries to identify and remove some bounded elements of the SVM dual problem, and the caching technique reduces the computational time of the decomposition of a dense matrix.

2.2. *Single-output RVM*

Tipping and Faul (2003) proposed a new marginal likelihood maximisation algorithm with efficient addition/deletion of candidate basis functions. The efficiency comes from modifying the marginal likelihood function:

$$\mathcal{L}(\boldsymbol{\alpha}) = -\frac{1}{2}\left(N\log(2\pi) + \log|\mathbf{C}| + \mathbf{t}^{\top}\mathbf{C}^{-1}\mathbf{t}\right), \tag{2.1}$$

where $\boldsymbol{\alpha} = [\alpha_0\ \alpha_1\ \ldots\ \alpha_N]^{\top}$ is a hyperparameter, N is the number of training samples, and $\mathbf{t}$ is a target. In the original algorithm by Tipping (2001), $\mathbf{C}$ is written as $\mathbf{C} = \sigma^2\mathbf{I} + \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^{\top}$, where $\mathbf{\Phi}$ is a design matrix, and $\mathbf{A} = \mathrm{diag}\,(\alpha_0, \alpha_1, \ldots, \alpha_N)$, whereas in the new algorithm by Tipping and Faul (2003), $\mathbf{C}$ is rewritten as $\mathbf{C} = \sigma^2\mathbf{I} + \sum_{m\neq i} \alpha_m^{-1}\boldsymbol{\phi}_m\boldsymbol{\phi}_m^{\top} + \alpha_i^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\top}$, where $\boldsymbol{\phi}$ is a basis vector (the details of the mathematical expressions are in Section 3.3 and Section 4.3).

Ben-Shimon and Shmilovici (2006) partitioned training samples into small chunks to avoid the inverse of a large matrix (the matrix inversion is the most computationally expensive part of RVM). They suggested three methods to accelerate the computation of the basic algorithm by Tipping (2001). Firstly, relevance vectors (RVs) from two partitions are merged together into a new partition recursively (this is called the split and merge RVM). Secondly, RVs from the basic RVM and the previous partition are merged consecutively (this is called the incremental RVM). Thirdly, the incremental RVM is advanced: the merge operation is performed with the partition of the most informative basis functions.

Yang *et al.* (2010) accelerated RVM computation by parallelising the matrix inversion operation on a GPU. To be specific, the Cholesky decomposition for the matrix inversion is implemented on a GPU. RVM uses an expectation–maximization (EM) algorithm, an iterative method, to find maximum likelihood (this is explained in Section 3.4 and Section 4.4), and the Cholesky decomposition is performed iteratively based on the Cholesky decomposition results of the previous EM iteration.

2.3. *Single-output GP regression*

Seeger *et al.* (2003) reduced GP regression training time by approximating the likelihood of training data. The approximation consists of the following three components: i) likelihood approximation for an active set of training samples (the active set is a part of the whole training samples, and the reduced number of training samples decreases the computation time of the GP regression training), ii) how to select the active set (information gain is calculated to score a new point for inclusion to the active set), and iii) marginal likelihood approximation for the active set.

Shen *et al.* (2006) reduced both the training and prediction time of GP regression by using an approximation method, based on a tree-type multiresolution data structure. They assumed that the kernel of GP regression is monotonic isotropic (e.g. the Gaussian radial basis function kernel) and grouped data points together that have similar weights. For the grouping, a k-d tree, a binary tree that recursively partitions a set of data points, was employed.

Srinivasan *et al.* (2010) accelerated linear algebra operations of GP regression on a GPU, and Gramacy *et al.* (2014) made a GPU-accelerated version of GP regression approximation. Srinivasan *et al.* i) implemented the weighted summation of kernel functions by considering GPU memory architecture, ii) accelerated iterative algorithms by having data between iterations stay on a GPU, and iii) constructed kernel matrices in parallel on a GPU. Gramacy *et al.* converted a large problem of GP regression into many small independent ones for a cascade implementation on a GPU.

3. Existing method

In the following subsections, we summarise the existing method of MRVR as in (Thayananthan 2005, Chapter 6), (Thayananthan *et al.* 2008).

3.1. *Model specification*

V-dimensional multi-output regression upon an input $\mathbf{x} \in \mathbb{R}^{U\times 1}$ (U-dimensional column vector), a weight $\mathbf{W} \in \mathbb{R}^{M\times V}$ (M-by-V matrix), and an output function $\mathbf{y}(\mathbf{x};\mathbf{W}) \in \mathbb{R}^{1\times V}$ (V-dimensional row vector) (upright bold lower case letters denote vectors, and upright bold capital letters denote matrices) is expressed as

$$\mathbf{y}(\mathbf{x};\mathbf{W}) = \left(\mathbf{W}^\top \boldsymbol{\phi}(\mathbf{x})\right)^\top, \tag{3.1}$$

where $\mathbf{W}^\top$ is the transpose of the matrix $\mathbf{W}$ (the objective of this chapter is to estimate proper values of $\mathbf{W}$), and $\boldsymbol{\phi}(\mathbf{x}) = [\phi_1(\mathbf{x})\ \phi_2(\mathbf{x})\ \dots\ \phi_M(\mathbf{x})]^\top \in \mathbb{R}^{M\times 1}$ is the transformed input by nonlinear and fixed basis functions. In other words, the output $\mathbf{y}(\mathbf{x};\mathbf{W})$ is a linearly weighted sum of the transformed input $\boldsymbol{\phi}(\mathbf{x})$.

Given a data set of input-target pairs $\left\{\mathbf{x}_i \in \mathbb{R}^{U\times 1}, \mathbf{t}_i \in \mathbb{R}^{1\times V}\right\}_{i=1}^{N}$, where N is the number of training samples, it is assumed that the targets $\mathbf{t}_i$ are samples from the model $\mathbf{y}(\mathbf{x}_i;\mathbf{W})$ with additive noise:

$$\mathbf{t}_i = \mathbf{y}(\mathbf{x}_i;\mathbf{W}) + \boldsymbol{\epsilon}_i, \tag{3.2}$$

where $\mathbf{W} \in \mathbb{R}^{(N+1)\times V}$ is the weight, $\boldsymbol{\epsilon}_i \in \mathbb{R}^{1\times V}$ are independent samples from a Gaussian noise process with mean zero and a covariance matrix $\boldsymbol{\Omega} \in \mathbb{R}^{V\times V}$, and $\boldsymbol{\Omega}$ is decomposed as the diagonal matrix of the variances $\mathbf{D} \in \mathbb{R}^{V\times V}$ and the correlation matrix $\mathbf{R} \in \mathbb{R}^{V\times V}$:

$$\boldsymbol{\Omega} = \mathbf{D}^{\frac{1}{2}}\mathbf{R}\mathbf{D}^{\frac{1}{2}}, \tag{3.3}$$

where $\mathbf{D} = \operatorname{diag}\left(\sigma_1^2, \sigma_2^2, \dots, \sigma_V^2\right)$, and σ_j^2 is the variance of the j-th dimension's noise.

Because of the ignorance of $\mathbf{R}$ by Thayananthan (2005) and the assumption of independent Gaussian noise, the likelihood of the data set can be given by the product of the Gaussian distributions:

$$p(\mathbf{T}|\mathbf{W},\boldsymbol{\sigma}) = \prod_{j=1}^{V}\left(2\pi\sigma_j^2\right)^{-\frac{N}{2}} \exp\left(-\frac{1}{2\sigma_j^2}\left\|\boldsymbol{\tau}_j - \boldsymbol{\Phi}\mathbf{w}_j\right\|^2\right), \tag{3.4}$$

where $\mathbf{T} = \begin{bmatrix}\mathbf{t}_1\\ \mathbf{t}_2\\ \vdots\\ \mathbf{t}_N\end{bmatrix} \in \mathbb{R}^{N\times V}$, $\boldsymbol{\sigma} = [\sigma_1\ \sigma_2\ \dots\ \sigma_V] \in \mathbb{R}_{\geq 0}^{1\times V}$, $\boldsymbol{\tau}_j \in \mathbb{R}^{N\times 1}$ is the j-th column vector of $\mathbf{T}$, $\mathbf{w}_j \in \mathbb{R}^{(N+1)\times 1}$ is the j-th column vector of $\mathbf{W}$, $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_1)\ \boldsymbol{\phi}(\mathbf{x}_2)\ \dots\ \boldsymbol{\phi}(\mathbf{x}_N)]^\top \in \mathbb{R}^{N\times(N+1)}$ is a design matrix, $\boldsymbol{\phi}(\mathbf{x}) = [1\ K(\mathbf{x},\mathbf{x}_1)\ \dots\ K(\mathbf{x},\mathbf{x}_N)]^\top \in \mathbb{R}^{(N+1)\times 1}$, and $K(\mathbf{x},\mathbf{x}')$ is a kernel function. For clarity, the implicit conditioning on the input $\mathbf{x}_i, \forall i$ is omitted in Eq. (3.4) and the subsequent expressions.

An assumption to avoid over-fitting in estimation of $\mathbf{W}$ is

$$p(\mathbf{W}|\boldsymbol{\alpha}) = \prod_{j=1}^{V}\prod_{i=0}^{N}\mathcal{N}\left(0, \alpha_i^{-1}\right). \tag{3.5}$$

This means the prior distribution of $\mathbf{w}_j$ is zero-mean Gaussian with inverse variances $\boldsymbol{\alpha} = [\alpha_0\ \alpha_1\ \dots\ \alpha_N]^\top \in \mathbb{R}_{>0}^{(N+1)\times 1}$, which are $N+1$ hyperparameters (Tipping 2001), and $\mathbf{w}_j$ and $\mathbf{w}_{j'}$ $(j \neq j')$ have the same distribution as $\prod_{i=0}^{N}\mathcal{N}\left(0, \alpha_i^{-1}\right)$.

3.2. *Inference*

By both the Bayes' theorem and the property of $p(\mathbf{T}|\mathbf{W},\boldsymbol{\alpha},\boldsymbol{\sigma}) = p(\mathbf{T}|\mathbf{W},\boldsymbol{\sigma})$,[1] the posterior probability distribution function over $\mathbf{W}$ is decomposed as

$$p(\mathbf{W}|\mathbf{T},\boldsymbol{\alpha},\boldsymbol{\sigma}) = \frac{p(\mathbf{T}|\mathbf{W},\boldsymbol{\sigma})\,p(\mathbf{W}|\boldsymbol{\alpha},\boldsymbol{\sigma})}{p(\mathbf{T}|\boldsymbol{\alpha},\boldsymbol{\sigma})}, \tag{3.6}$$

and it is given by the product of multivariate Gaussian distributions:

$$p(\mathbf{W}|\mathbf{T},\boldsymbol{\alpha},\boldsymbol{\sigma}) = \prod_{j=1}^{V}(2\pi)^{-\frac{N+1}{2}}|\boldsymbol{\Sigma}_j|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(\mathbf{w}_j-\boldsymbol{\mu}_j)^{\top}\boldsymbol{\Sigma}_j^{-1}(\mathbf{w}_j-\boldsymbol{\mu}_j)\right), \tag{3.7}$$

where the j-th posterior covariance and mean are, respectively:

$$\boldsymbol{\Sigma}_j = \left(\sigma_j^{-2}\boldsymbol{\Phi}^{\top}\boldsymbol{\Phi}+\mathbf{A}\right)^{-1}, \tag{3.8}$$

$$\boldsymbol{\mu}_j = \sigma_j^{-2}\boldsymbol{\Sigma}_j\boldsymbol{\Phi}^{\top}\boldsymbol{\tau}_j, \tag{3.9}$$

where $\mathbf{A} = \mathrm{diag}(\alpha_0,\alpha_1,\ldots,\alpha_N) \in \mathbb{R}^{(N+1)\times(N+1)}$.

In the case of uniform hyperpriors $\boldsymbol{\alpha}$ and $\boldsymbol{\sigma}$, maximising a posteriori $p(\boldsymbol{\alpha},\boldsymbol{\sigma}|\mathbf{T}) \propto p(\mathbf{T}|\boldsymbol{\alpha},\boldsymbol{\sigma})\,p(\boldsymbol{\alpha})\,p(\boldsymbol{\sigma})$ is equivalent to maximising the marginal likelihood $p(\mathbf{T}|\boldsymbol{\alpha},\boldsymbol{\sigma})$, which is given by

$$p(\mathbf{T}|\boldsymbol{\alpha},\boldsymbol{\sigma}) = \prod_{j=1}^{V}(2\pi)^{-\frac{N}{2}}\left|\sigma_j^2\mathbf{I}+\boldsymbol{\Phi}\mathbf{A}^{-1}\boldsymbol{\Phi}^{\top}\right|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}\boldsymbol{\tau}_j^{\top}\left(\sigma_j^2\mathbf{I}+\boldsymbol{\Phi}\mathbf{A}^{-1}\boldsymbol{\Phi}^{\top}\right)^{-1}\boldsymbol{\tau}_j\right). \tag{3.10}$$

3.3. *Marginal likelihood maximisation*

Following Tipping and Faul (2003), the log of Eq. (3.10) is an objective function:

$$\mathcal{L}(\boldsymbol{\alpha},\boldsymbol{\sigma}) = -\frac{1}{2}\sum_{j=1}^{V}\left(N\log(2\pi)+\log|\mathbf{C}_j|+\boldsymbol{\tau}_j^{\top}\mathbf{C}_j^{-1}\boldsymbol{\tau}_j\right), \tag{3.11}$$

where $\mathbf{C}_j = \sigma_j^2\mathbf{I}+\boldsymbol{\Phi}\mathbf{A}^{-1}\boldsymbol{\Phi}^{\top} \in \mathbb{R}^{N\times N}$, and by considering the dependence of $\mathcal{L}(\boldsymbol{\alpha},\boldsymbol{\sigma})$ on a single hyperparameter $\alpha_i, i \in \{0,1,\ldots,N\}$, $\mathbf{C}_j$ is decomposed as the following two parts:

$$\begin{aligned}\mathbf{C}_j &= \sigma_j^2\mathbf{I}+\sum_{m\neq i}\alpha_m^{-1}\boldsymbol{\phi}_m\boldsymbol{\phi}_m^{\top}+\alpha_i^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\top}\\ &= \mathbf{C}_{-i,j}+\alpha_i^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\top},\end{aligned} \tag{3.12}$$

where $\mathbf{C}_{-i,j} \in \mathbb{R}^{N\times N}$ is $\mathbf{C}_j$ with the contribution of a basis vector $\boldsymbol{\phi}_i \in \mathbb{R}^{N\times 1}$ removed, and

$$\boldsymbol{\phi}_i = \begin{cases} [1\ 1\ \ldots\ 1]^{\top}, & \text{if } i = 0\\ [K(\mathbf{x}_i,\mathbf{x}_1)\ K(\mathbf{x}_i,\mathbf{x}_2)\ \ldots\ K(\mathbf{x}_i,\mathbf{x}_N)]^{\top}, & \text{otherwise}\end{cases}. \tag{3.13}$$

[1] In the case that the weight $\mathbf{W}$ is given, its inverse variances $\boldsymbol{\alpha}$ are redundant in the calculation of the conditional probability of the target $\mathbf{T}$.

The determinant and inverse matrix of $\mathbf{C}_j$ are, respectively:

$$|\mathbf{C}_j| = |\mathbf{C}_{-i,j}|\left(1 + \alpha_i^{-1}\boldsymbol{\phi}_i^\top \mathbf{C}_{-i,j}^{-1}\boldsymbol{\phi}_i\right), \tag{3.14}$$

by Sylvester's determinant theorem, and

$$\mathbf{C}_j^{-1} = \mathbf{C}_{-i,j}^{-1} - \frac{\mathbf{C}_{-i,j}^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top\mathbf{C}_{-i,j}^{-1}}{\alpha_i + \boldsymbol{\phi}_i^\top\mathbf{C}_{-i,j}^{-1}\boldsymbol{\phi}_i}, \tag{3.15}$$

by Woodbury matrix identity. From these, $\mathcal{L}(\boldsymbol{\alpha},\boldsymbol{\sigma})$ in Eq. (3.11) can be decomposed into $\mathcal{L}(\boldsymbol{\alpha}_{-i},\boldsymbol{\sigma})$, the marginal likelihood with $\boldsymbol{\phi}_i$ excluded, and $\ell(\alpha_i,\boldsymbol{\sigma})$, the isolated marginal likelihood of $\boldsymbol{\phi}_i$:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\alpha},\boldsymbol{\sigma}) = &-\frac{1}{2}\sum_{j=1}^{V}\left(N\log(2\pi) + \log|\mathbf{C}_{-i,j}| + \boldsymbol{\tau}_j^\top\mathbf{C}_{-i,j}^{-1}\boldsymbol{\tau}_j\right)\\ &-\frac{1}{2}\sum_{j=1}^{V}\left(-\log\alpha_i + \log\left(\alpha_i + \boldsymbol{\phi}_i^\top\mathbf{C}_{-i,j}^{-1}\boldsymbol{\phi}_i\right) - \frac{\left(\boldsymbol{\phi}_i^\top\mathbf{C}_{-i,j}^{-1}\boldsymbol{\tau}_j\right)^2}{\alpha_i + \boldsymbol{\phi}_i^\top\mathbf{C}_{-i,j}^{-1}\boldsymbol{\phi}_i}\right)\\ =&\mathcal{L}(\boldsymbol{\alpha}_{-i},\boldsymbol{\sigma}) + \frac{1}{2}\sum_{j=1}^{V}\left(\log\alpha_i - \log\left(\alpha_i + s_{i,j}\right) + \frac{q_{i,j}^2}{\alpha_i + s_{i,j}}\right)\\ =&\mathcal{L}(\boldsymbol{\alpha}_{-i},\boldsymbol{\sigma}) + \ell(\alpha_i,\boldsymbol{\sigma}),\end{aligned} \tag{3.16}$$

where $s_{i,j}$ and $q_{i,j}$ are defined as, respectively:

$$s_{i,j} \overset{\text{def}}{=} \boldsymbol{\phi}_i^\top\mathbf{C}_{-i,j}^{-1}\boldsymbol{\phi}_i, \tag{3.17a}$$

$$q_{i,j} \overset{\text{def}}{=} \boldsymbol{\phi}_i^\top\mathbf{C}_{-i,j}^{-1}\boldsymbol{\tau}_j. \tag{3.17b}$$

To avoid the matrix inversion of $\mathbf{C}_{-i}$ in Eq. (3.17), which requires the time complexity of $O\left(N^3\right)$, $s'_{i,j}$ and $q'_{i,j}$ are computed as, respectively (by the Woodbury matrix identity):[1]

$$\begin{aligned}s'_{i,j} &= \boldsymbol{\phi}_i^\top\mathbf{C}_j^{-1}\boldsymbol{\phi}_i\\ &= \sigma_j^{-2}\boldsymbol{\phi}_i^\top\boldsymbol{\phi}_i - \sigma_j^{-4}\boldsymbol{\phi}_i^\top\boldsymbol{\Phi}\boldsymbol{\Sigma}_j\boldsymbol{\Phi}^\top\boldsymbol{\phi}_i,\end{aligned} \tag{3.18a}$$

$$\begin{aligned}q'_{i,j} &= \boldsymbol{\phi}_i^\top\mathbf{C}_j^{-1}\boldsymbol{\tau}_j\\ &= \sigma_j^{-2}\boldsymbol{\phi}_i^\top\boldsymbol{\tau}_j - \sigma_j^{-4}\boldsymbol{\phi}_i^\top\boldsymbol{\Phi}\boldsymbol{\Sigma}_j\boldsymbol{\Phi}^\top\boldsymbol{\tau}_j,\end{aligned} \tag{3.18b}$$

and then $s_{i,j}$ and $q_{i,j}$ in Eq. (3.17) are computed as, respectively:[2]

$$s_{i,j} = \frac{\alpha_i s'_{i,j}}{\alpha_i - s'_{i,j}}, \tag{3.19a}$$

[1] $s'_{i,j} = \sigma_j^{-2}\boldsymbol{\phi}_i^\top\boldsymbol{\phi}_i$ and $q'_{i,j} = \sigma_j^{-2}\boldsymbol{\phi}_i^\top\boldsymbol{\tau}_j$ when $\alpha_i = \infty, \forall i$.
[2] $s_{i,j} = s'_{i,j}$ and $q_{i,j} = q'_{i,j}$ when $\alpha_i = \infty$.

$$q_{i,j} = \frac{\alpha_i q'_{i,j}}{\alpha_i - s'_{i,j}}. \tag{3.19b}$$

$\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\sigma})$ has a unique maximum with respect to α_i when the following equation is satisfied:

$$\frac{\partial \ell(\alpha_i, \boldsymbol{\sigma})}{\partial \alpha_i} = \frac{1}{2}\sum_{j=1}^{V}\left(\frac{1}{\alpha_i} - \frac{1}{\alpha_i + s_{i,j}} - \frac{q_{i,j}^2}{(\alpha_i + s_{i,j})^2}\right) = 0, \tag{3.20}$$

which is a $(2V-1)$-th order polynomial equation of α_i. This implies that:

- If $\boldsymbol{\phi}_i$ is "in the model" (i.e. $\alpha_i < \infty$) and α_i in Eq. (3.20) has at least one positive real root ($\alpha_i > 0$ as α_i is inverse variance); then, α_i is re-estimated,
- If $\boldsymbol{\phi}_i$ is "in the model" (i.e. $\alpha_i < \infty$) yet α_i in Eq. (3.20) does not have any positive real root; then, $\boldsymbol{\phi}_i$ may be deleted (i.e. α_i is set to be ∞),
- If $\boldsymbol{\phi}_i$ is "out of the model" (i.e. $\alpha_i = \infty$) yet α_i in Eq. (3.20) has at least one positive real root; then, $\boldsymbol{\phi}_i$ may be added (i.e. α_i is set to be a finite value).

In addition, $\dfrac{\partial \mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\sigma})}{\partial \sigma_j^2} = 0$ leads to that $\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\sigma})$ has a unique maximum with respect to σ_j^2 when:

$$\sigma_j^2 = \frac{\|\boldsymbol{\tau}_j - \boldsymbol{\Phi}\boldsymbol{\mu}_j\|^2}{N - \sum_{i=1}^{N+1}\gamma_{i,j}}, \tag{3.21}$$

where $\gamma_{i,j} \overset{\text{def}}{=} 1 - \alpha_{(i-1)}\Sigma_{j,ii}$, and $\Sigma_{j,ii}$ is the i-th diagonal element of $\boldsymbol{\Sigma}_j \in \mathbb{R}^{(N+1)\times(N+1)}$.

3.4. *Expectation–maximisation (EM) algorithm*

Algorithm 1, an EM algorithm to maximise the marginal likelihood, starts without any basis vector (i.e. $M = 0$) and selects the basis vector $\boldsymbol{\phi}_i$ which gives the maximum change of the marginal likelihood $\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\sigma})$ of Eq. (3.11) at every iteration.

For efficient computation of the EM algorithm, quantities $\boldsymbol{\Phi} \in \mathbb{R}^{N\times M}$, $\boldsymbol{\Sigma}_j \in \mathbb{R}^{M\times M}$, and $\boldsymbol{\mu}_j \in \mathbb{R}^{M\times 1}$ contain only M ($M \le N+1$ is always satisfied) basis functions that are currently included in the model (i.e. $\boldsymbol{\phi}_i$ satisfying $\alpha_i < \infty$), and the diagonal matrix $\mathbf{A}$ consists of M hyperparameters of α_i that are currently included in the model (i.e. α_i satisfying $\alpha_i < \infty$). Additionally, Eq. (3.21) is rewritten as

$$\sigma_j^2 = \frac{\|\boldsymbol{\tau}_j - \boldsymbol{\Phi}\boldsymbol{\mu}_j\|^2}{N - \sum_{i=1}^{M}\gamma'_{i,j}}, \tag{3.22}$$

where $\gamma'_{i,j} \overset{\text{def}}{=} 1 - \alpha'_i\Sigma_{j,ii}$, α'_i is the i-th non-infinity value of $\boldsymbol{\alpha}$, and $\Sigma_{j,ii}$ is the i-th diagonal element of $\boldsymbol{\Sigma}_j \in \mathbb{R}^{M\times M}$.

From Eq. (3.11), the change in the marginal likelihood can be written as

$$\begin{aligned} 2\Delta\mathcal{L} &= 2\left(\mathcal{L}(\tilde{\boldsymbol{\alpha}}, \boldsymbol{\sigma}) - \mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\sigma})\right) \\ &= \sum_{j=1}^{V}\left(\log\frac{|\mathbf{C}_j|}{|\tilde{\mathbf{C}}_j|} + \boldsymbol{\tau}_j^{\mathsf{T}}\left(\mathbf{C}_j^{-1} - \tilde{\mathbf{C}}_j^{-1}\right)\boldsymbol{\tau}_j\right), \end{aligned} \tag{3.23}$$

where updated quantities are denoted by a tilde (e.g., $\tilde{\boldsymbol{\alpha}}$ and $\tilde{\mathbf{C}}_j$). Eq. (3.23) differs according to whether α_i is re-estimated, added, or deleted:

Algorithm 1: Existing EM algorithm of MRVR.

```
Input: $\mathbf{T} \in \mathbb{R}^{N\times V}$, and $\boldsymbol{\phi}_i \in \mathbb{R}^{N\times 1}, \forall i = \{0, 1, \ldots, N\}$, where $N$ is the number of training samples, and $V$ is the number
       of output dimensions
Output: $\boldsymbol{\Sigma}_j \in \mathbb{R}^{M\times M}, \boldsymbol{\mu}_j \in \mathbb{R}^{M\times 1}$, and $\sigma_j, \forall j = \{1, 2, \ldots, V\}$, where $M$ is the number of basis functions in the model
// Initialisation
$\alpha_i \leftarrow \infty, \forall i = \{0, 1, \ldots, N\}$
for $j \leftarrow 1$ to $V$ do
    $\bar{t}_j \leftarrow \frac{1}{N}\sum_{i=1}^{N} t_{i,j}$
    $\sigma_j^2 \leftarrow \frac{0.1}{N-1}\sum_{i=1}^{N}(t_{i,j} - \bar{t}_j)^2$
end
convergence←false, $n \leftarrow 1$, $M \leftarrow 0$, where $n$ is the iteration number, and $M$ is the number of basis functions.
while convergence=false do
    // maximisation step
    for $i \leftarrow 0$ to $N$ do
        for $j \leftarrow 1$ to $V$ do
            Update $s'_{i,j}$ and $q'_{i,j}$ using Eq. (3.18), and Update $s_{i,j}$ and $q_{i,j}$ using Eq. (3.19).
        end
        switch the number of positive real roots of Eq. (3.20) do
            case 0 do
                $\tilde{\alpha}_i \leftarrow \infty$
            case 1 do
                $\tilde{\alpha}_i \leftarrow$ the positive real root of Eq. (3.20)
            otherwise do
                $\tilde{\alpha}_i \leftarrow$ one of the positive real roots of Eq. (3.20), which maximises $2\Delta\mathcal{L}_i$ of either i) Eq. (3.24) if
                  $\alpha_i < \infty$ or ii) Eq. (3.25) if $\alpha_i = \infty$
            end
        end
        if $\tilde{\alpha}_i < \infty$ then
            if $\alpha_i < \infty$ then $z_i \leftarrow$'re-estimation'
                Update $2\Delta\mathcal{L}_i$ using Eq. (3.24).
            else $z_i \leftarrow$'addition'
                Update $2\Delta\mathcal{L}_i$ using Eq. (3.25).
            end
        else if $\alpha_i < \infty$ then$z_i \leftarrow$'deletion'
            Update $2\Delta\mathcal{L}_i$ using Eq. (3.26).
        else
            $2\Delta\mathcal{L}_i \leftarrow -\infty$
        end
    end
    $i \leftarrow \arg\max_i 2\Delta\mathcal{L}_i$ // Select i which gives the greatest increase of the marginal likelihood
    if $n \neq 1$ then
        Update $\sigma_j, \forall j$ using Eq. (3.22).
    end
    switch $z_i$ do
        case 're-estimation' do
            // Check convergence
            $\Delta\log\alpha \leftarrow \log\frac{\alpha_i}{\tilde{\alpha}_i}$
            if $|\Delta\log\alpha| < 0.1$ then
                convergence←true
                for $i \leftarrow 0$ to $N$ do
                    if $\alpha_i = \infty$ then // if $\boldsymbol{\phi}_i$ is "out of the model"
                        if $\tilde{\alpha}_i < \infty$ then // if $\boldsymbol{\phi}_i$ may be added "in the model"
                            convergence←false
                            break for loop
                        end
                    end
                end
            end
            $\alpha_i \leftarrow \tilde{\alpha}_i$
        case 'addition' do
            $\alpha_i \leftarrow \tilde{\alpha}_i$, $M \leftarrow M + 1$
        case 'deletion' do
            $\alpha_i \leftarrow \infty$, $M \leftarrow M - 1$
    end
    // Expectation step
    Sequentially update i) $\boldsymbol{\Phi} \in \mathbb{R}^{N\times M}$, $\mathbf{A} \in \mathbb{R}^{M\times M}$, ii) $\boldsymbol{\Sigma}_j \in \mathbb{R}^{M\times M}, \forall j$, and iii) $\boldsymbol{\mu}_j \in \mathbb{R}^{M}, \forall j$ using Eq. (3.8) and
      Eq. (3.9), where $\boldsymbol{\Phi}$, $\boldsymbol{\Sigma}_j$, and $\boldsymbol{\mu}_j$ contain only $M$ basis functions that are currently included in the model, and the
      diagonal matrix $\mathbf{A}$ consists of $M$ hyperparameters of $\alpha_i$ that are currently included in the model.
    $n \leftarrow n + 1$
end
```

Re-estimation. as $\mathbf{C}_j = \mathbf{C}_{-i,j} + \alpha_i^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top$ and $\tilde{\mathbf{C}} = \mathbf{C}_{-i,j} + \tilde{\alpha}_i^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top$,

$$2\Delta\mathcal{L}_i = \sum_{j=1}^{V}\left(\frac{q_{i,j}'^2}{s_{i,j}' + \left(\tilde{\alpha}_i^{-1} - \alpha_i^{-1}\right)^{-1}} - \log\left(1 + s_{i,j}'\left(\tilde{\alpha}_i^{-1} - \alpha_i^{-1}\right)\right)\right), \tag{3.24}$$

where $\tilde{\alpha}_i$ is re-estimated α_i,

Addition. as $\mathbf{C}_j = \mathbf{C}_{-i,j}$ and $\tilde{\mathbf{C}}_j = \mathbf{C}_{-i,j} + \tilde{\alpha}_i^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top$,

$$2\Delta\mathcal{L}_i = \sum_{j=1}^{V}\left(\frac{q_{i,j}^2}{\tilde{\alpha}_i + s_{i,j}} + \log\frac{\tilde{\alpha}_i}{\tilde{\alpha}_i + s_{i,j}}\right), \tag{3.25}$$

Deletion. as $\mathbf{C}_j = \mathbf{C}_{-i,j} + \alpha_i^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top$ and $\tilde{\mathbf{C}}_j = \mathbf{C}_{-i,j}$,

$$2\Delta\mathcal{L}_i = \sum_{j=1}^{V}\left(\frac{q_{i,j}'^2}{s_{i,j}' - \alpha_i} - \log\left(1 - \frac{s_{i,j}'}{\alpha_i}\right)\right). \tag{3.26}$$

The convergence criteria of the EM algorithm are presented between the 39-th line and the 50-th line of Algorithm 1. The logarithmic change $|\Delta \log \alpha|$ should be less than a tolerance, and any basis vector $\boldsymbol{\phi}_i$ "out of the model" for the current iteration should not be added "in the model" for the next iteration, which are the same as those in Tipping and Faul (2003).

3.5. *Making predictions*

We can predict both the mean of j-th output dimension $y_{*,j}$ and its variance $\sigma_{*,j}^2$ from a new input vector $\mathbf{x}_*$ based on both i) Eq. (3.2), the model specification, and ii) Eq. (3.7), the posterior distribution over the weights, conditioned on the most probable (MP) hyperparameters: $\boldsymbol{\alpha}_{\text{MP}} \in \mathbb{R}_{>0}^{M\times 1}$ and $\boldsymbol{\sigma}_{\text{MP}} \in \mathbb{R}_{\geq 0}^{1\times V}$, obtained from Algorithm 1. Predictive distribution of $t_{*,j}$ is normally distributed as

$$p(t_{*,j}|\mathbf{T}, \boldsymbol{\alpha}_{\text{MP}}, \boldsymbol{\sigma}_{\text{MP}}) = \mathcal{N}\left(t_{*,j}|y_{*,j}, \sigma_{*,j}^2\right), \tag{3.27}$$

with

$$y_{*,j} = \boldsymbol{\phi}(\mathbf{x}_*)^\top\boldsymbol{\mu}_j, \tag{3.28}$$

and

$$\sigma_{*,j}^2 = \sigma_{\text{MP},j}^2 + \boldsymbol{\phi}(\mathbf{x}_*)^\top\boldsymbol{\Sigma}_j\boldsymbol{\phi}(\mathbf{x}_*), \tag{3.29}$$

where $\boldsymbol{\phi}(\mathbf{x}_*) \in \mathbb{R}^{M\times 1}$ comes from only M basis functions that are included in the model after the EM algorithm, and subscript j refers to the j-th output dimension. The predictive variance $\sigma_{*,j}^2$ comprises the sum of two variance components: the estimated noise on the training data $\sigma_{\text{MP},j}^2$ and that due to the uncertainty in the prediction of the weights $\boldsymbol{\phi}(\mathbf{x}_*)^\top\boldsymbol{\Sigma}_j\boldsymbol{\phi}(\mathbf{x}_*)$.

3.6. *Algorithm complexity*

Matrix inversion of $\boldsymbol{\Sigma}_j \in \mathbb{R}^{M\times M}$ in Eq. (3.8) for all $j \in \{1, 2, \ldots, V\}$ determines i) the time complexity of the existing algorithm as $O\left(VM^3\right)$ and ii) the memory complexity as $O\left(VM^2\right)$, where V is the number of output dimensions, and M is the number of basis functions.[1]

4. Proposed method

4.1. *Model specification*

Given a data set of input-target pairs $\left\{\mathbf{x}_i \in \mathbb{R}^{U\times 1}, \mathbf{t}_i \in \mathbb{R}^{1\times V}\right\}_{i=1}^{N}$, where N is the number of training samples, it is assumed that the targets $\mathbf{t}_i$ are samples from the model $\mathbf{y}(\mathbf{x}_i;\mathbf{W})$ with additive noise:

$$\mathbf{t}_i = \mathbf{y}(\mathbf{x}_i;\mathbf{W}) + \boldsymbol{\epsilon}_i, \tag{4.1}$$

where $\mathbf{W} \in \mathbb{R}^{(N+1)\times V}$ is the weight and $\boldsymbol{\epsilon}_i \in \mathbb{R}^{1\times V}$ are independent samples from a Gaussian noise process with mean zero and a covariance matrix $\boldsymbol{\Omega} \in \mathbb{R}^{V\times V}$.

Eq. (4.1) can be rewritten, using matrix algebra, as

$$\mathbf{T} = \boldsymbol{\Phi}\mathbf{W} + \mathbf{E}, \tag{4.2}$$

where $\mathbf{T} = \begin{bmatrix}\mathbf{t}_1\\ \mathbf{t}_2\\ \vdots\\ \mathbf{t}_N\end{bmatrix} \in \mathbb{R}^{N\times V}$ is the target, $\mathbf{E} = \begin{bmatrix}\boldsymbol{\epsilon}_1\\ \boldsymbol{\epsilon}_2\\ \vdots\\ \boldsymbol{\epsilon}_N\end{bmatrix} \in \mathbb{R}^{N\times V}$ is the noise, $\boldsymbol{\Phi} = \left[\boldsymbol{\phi}(\mathbf{x}_1)\ \boldsymbol{\phi}(\mathbf{x}_2)\ \dots\ \boldsymbol{\phi}(\mathbf{x}_N)\right]^\top \in \mathbb{R}^{N\times(N+1)}$ is a design matrix, $\boldsymbol{\phi}(\mathbf{x}) = \left[1\ K(\mathbf{x},\mathbf{x}_1)\ K(\mathbf{x},\mathbf{x}_2)\ \dots\ K(\mathbf{x},\mathbf{x}_N)\right]^\top \in \mathbb{R}^{(N+1)\times 1}$, and $K(\mathbf{x},\mathbf{x}')$ is a kernel function.

Unlike the existing method of Thayananthan *et al.* (2008), the likelihood of the data set is given by the matrix Gaussian distribution:

$$p(\mathbf{T}|\mathbf{W},\boldsymbol{\Omega}) = (2\pi)^{-\frac{VN}{2}}|\boldsymbol{\Omega}|^{-\frac{N}{2}}\exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Omega}^{-1}(\mathbf{T}-\boldsymbol{\Phi}\mathbf{W})^\top(\mathbf{T}-\boldsymbol{\Phi}\mathbf{W})\right)\right), \tag{4.3}$$

where $\boldsymbol{\Omega} = \frac{\mathbb{E}[\mathbf{E}^\top\mathbf{E}]}{N}$, and tr denotes trace.[2] As assumed, $\mathbf{I} = \frac{\mathbb{E}[\mathbf{E}\mathbf{E}^\top]}{\mathrm{tr}(\boldsymbol{\Omega})}$, which means the noise is independent among rows with the same variance, where $\mathbf{I}$ is an $N\times N$ identity matrix. For clarity, the implicit conditioning on the input $\mathbf{x}_i, \forall i$ is omitted in Eq. (4.3) and the subsequent expressions.

An assumption to avoid over-fitting in the estimation of $\mathbf{W}$ is

$$p(\mathbf{W}|\boldsymbol{\alpha},\boldsymbol{\Omega}) = (2\pi)^{-\frac{V(N+1)}{2}}|\boldsymbol{\Omega}|^{-\frac{N+1}{2}}|\mathbf{A}|^{\frac{V}{2}}\exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Omega}^{-1}\mathbf{W}^\top\mathbf{A}\mathbf{W}\right)\right), \tag{4.4}$$

where $\mathbf{A}^{-1} = \mathrm{diag}\left(\alpha_0^{-1},\alpha_1^{-1},\dots,\alpha_N^{-1}\right) = \frac{\mathbb{E}[\mathbf{W}\mathbf{W}^\top]}{\mathrm{tr}(\boldsymbol{\Omega})}$. This means the prior distribution of $\mathbf{W}$ is zero-mean Gaussian with among-row inverse variances $\boldsymbol{\alpha} = [\alpha_0\ \alpha_1\ \dots\ \alpha_N]^\top \in \mathbb{R}_{>0}^{(N+1)\times 1}$, which are $N+1$ hyperparameters (Tipping 2001). Eq. (4.4) implies another assumption: $\boldsymbol{\Omega} = \frac{\mathbb{E}[\mathbf{W}^\top\mathbf{W}]}{\mathrm{tr}(\mathbf{A}^{-1})}$. Actually, this is unreasonable because the weight $\mathbf{W}$ has no relationship with the noise $\mathbf{E}$ (i.e. $\mathbf{I} = \frac{\mathbb{E}[\mathbf{W}^\top\mathbf{W}]}{\mathrm{tr}(\mathbf{A}^{-1})}$, which means that the weights of different output dimensions are not correlated, is a reasonable assumption), but it is essential for creating a computationally efficient algorithm.

[1] The matrix multiplication to calculate $s'_{i,j}$ and $q'_{i,j}$ in Eq. (3.18) for all $i \in \{1,2,\dots,N\}$, $j \in \{1,2,\dots,V\}$ at the 11-th line of Algorithm 1 has the same time complexity because the matrix multiplication $\boldsymbol{\Phi}\boldsymbol{\Sigma}_j\boldsymbol{\Phi}^\top$ is pre-calculated. In other words, the time complexity of the matrix multiplication is $O\left(VM^3\right)$, not $O\left(NVM^3\right)$, because $\boldsymbol{\Phi}\boldsymbol{\Sigma}_j\boldsymbol{\Phi}^\top$ is independent of i.

[2] If $\boldsymbol{\Omega} = \mathrm{diag}\left(\sigma_1^2,\sigma_2^2,\dots,\sigma_V^2\right)$, then Eq. (4.3) will be Eq. (3.4).

4.2. *Inference*

By both the Bayes' theorem and the property of $p\left(\mathbf{T}|\mathbf{W},\boldsymbol{\alpha},\boldsymbol{\Omega}\right) = p\left(\mathbf{T}|\mathbf{W},\boldsymbol{\Omega}\right)$,[1] the posterior probability distribution function over $\mathbf{W}$ is decomposed as

$$p\left(\mathbf{W}|\mathbf{T},\boldsymbol{\alpha},\boldsymbol{\Omega}\right) = \frac{p\left(\mathbf{T}|\mathbf{W},\boldsymbol{\Omega}\right)p\left(\mathbf{W}|\boldsymbol{\alpha},\boldsymbol{\Omega}\right)}{p\left(\mathbf{T}|\boldsymbol{\alpha},\boldsymbol{\Omega}\right)}, \tag{4.5}$$

and it is given by the matrix Gaussian distribution:[2]

$$p\left(\mathbf{W}|\mathbf{T},\boldsymbol{\alpha},\boldsymbol{\Omega}\right) = (2\pi)^{-\frac{V(N+1)}{2}}\left|\boldsymbol{\Omega}\right|^{-\frac{N+1}{2}}\left|\boldsymbol{\Sigma}\right|^{-\frac{V}{2}}\exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Omega}^{-1}\left(\mathbf{W}-\mathbf{M}\right)^{\top}\boldsymbol{\Sigma}^{-1}\left(\mathbf{W}-\mathbf{M}\right)\right)\right), \tag{4.6}$$

where the posterior covariance and mean are, respectively:

$$\boldsymbol{\Sigma} = \left(\boldsymbol{\Phi}^{\top}\boldsymbol{\Phi}+\mathbf{A}\right)^{-1}, \tag{4.7}$$

$$\mathbf{M} = \boldsymbol{\Sigma}\boldsymbol{\Phi}^{\top}\mathbf{T}. \tag{4.8}$$

In the case of uniform hyperpriors $\boldsymbol{\alpha}$ and $\boldsymbol{\Omega}$, maximising a posteriori $p\left(\boldsymbol{\alpha},\boldsymbol{\Omega}|\mathbf{T}\right) \propto p\left(\mathbf{T}|\boldsymbol{\alpha},\boldsymbol{\Omega}\right)p\left(\boldsymbol{\alpha}\right)p\left(\boldsymbol{\Omega}\right)$ is equivalent to maximising the marginal likelihood $p\left(\mathbf{T}|\boldsymbol{\alpha},\boldsymbol{\Omega}\right)$, which is given by:

$$p\left(\mathbf{T}|\boldsymbol{\alpha},\boldsymbol{\Omega}\right) = (2\pi)^{-\frac{VN}{2}}\left|\boldsymbol{\Omega}\right|^{-\frac{N}{2}}\left|\mathbf{I}+\boldsymbol{\Phi}\mathbf{A}^{-1}\boldsymbol{\Phi}^{\top}\right|^{-\frac{V}{2}}\exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Omega}^{-1}\mathbf{T}^{\top}\left(\mathbf{I}+\boldsymbol{\Phi}\mathbf{A}^{-1}\boldsymbol{\Phi}^{\top}\right)^{-1}\mathbf{T}\right)\right). \tag{4.9}$$

4.3. *Marginal likelihood maximisation*

We follow Tipping and Faul (2003)'s method via maximising the log likelihood function to accelerate the proposed algorithm.

The log of Eq. (4.9) is an objective function:

$$\mathcal{L}(\boldsymbol{\alpha},\boldsymbol{\Omega}) = -\frac{1}{2}\left(VN\log(2\pi)+N\log\left|\boldsymbol{\Omega}\right|+V\log\left|\mathbf{C}\right|+\mathrm{tr}\left(\boldsymbol{\Omega}^{-1}\mathbf{T}^{\top}\mathbf{C}^{-1}\mathbf{T}\right)\right), \tag{4.10}$$

where $\mathbf{C} = \mathbf{I}+\boldsymbol{\Phi}\mathbf{A}^{-1}\boldsymbol{\Phi}^{\top} \in \mathbb{R}^{N\times N}$, and by considering the dependence of $\mathcal{L}(\boldsymbol{\alpha},\boldsymbol{\Omega})$ on a single hyperparameter $\alpha_i, i \in \{0,1,\ldots,N\}$, $\mathbf{C}$ is decomposed as the following two parts:

$$\begin{aligned}\mathbf{C} &= \mathbf{I}+\sum_{m\neq i}\alpha_m^{-1}\boldsymbol{\phi}_m\boldsymbol{\phi}_m^{\top}+\alpha_i^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\top}\\ &= \mathbf{C}_{-i}+\alpha_i^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\top},\end{aligned} \tag{4.11}$$

where $\mathbf{C}_{-i} \in \mathbb{R}^{N\times N}$ is $\mathbf{C}$ with the contribution of a basis vector $\boldsymbol{\phi}_i \in \mathbb{R}^{N\times 1}$ removed, and

$$\boldsymbol{\phi}_i = \begin{cases} [1\ 1\ \ldots\ 1]^{\top}, & \text{if } i=0\\ [K(\mathbf{x}_i,\mathbf{x}_1)\ K(\mathbf{x}_i,\mathbf{x}_2)\ \ldots\ K(\mathbf{x}_i,\mathbf{x}_N)]^{\top}, & \text{otherwise}\end{cases}. \tag{4.12}$$

[1] In the case that the weight $\mathbf{W}$ is given, its inverse variances $\boldsymbol{\alpha}$ are redundant in the calculation of the conditional probability of the target $\mathbf{T}$.

[2] The proof is in Appendix.

The determinant and inverse matrix of $\mathbf{C}$ are, respectively:

$$|\mathbf{C}| = |\mathbf{C}_{-i}| \left(1 + \alpha_i^{-1} \boldsymbol{\phi}_i^\top \mathbf{C}_{-i}^{-1} \boldsymbol{\phi}_i\right), \tag{4.13}$$

by Sylvester's determinant theorem, and

$$\mathbf{C}^{-1} = \mathbf{C}_{-i}^{-1} - \frac{\mathbf{C}_{-i}^{-1} \boldsymbol{\phi}_i \boldsymbol{\phi}_i^\top \mathbf{C}_{-i}^{-1}}{\alpha_i + \boldsymbol{\phi}_i^\top \mathbf{C}_{-i}^{-1} \boldsymbol{\phi}_i}, \tag{4.14}$$

by Woodbury matrix identity. From these, $\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\Omega})$ in Eq. (4.10) can be decomposed into $\mathcal{L}(\boldsymbol{\alpha}_{-i}, \boldsymbol{\Omega})$, the marginal likelihood with $\boldsymbol{\phi}_i$ excluded, and $\ell(\alpha_i, \boldsymbol{\Omega})$, the isolated marginal likelihood of $\boldsymbol{\phi}_i$:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\Omega}) =& -\frac{1}{2} \left(VN \log(2\pi) + N \log|\boldsymbol{\Omega}| + V \log|\mathbf{C}_{-i}| + \operatorname{tr}\left(\boldsymbol{\Omega}^{-1} \mathbf{T}^\top \mathbf{C}_{-i}^{-1} \mathbf{T}\right)\right) \\
& - \frac{1}{2} \left(-V \log \alpha_i + V \log\left(\alpha_i + \boldsymbol{\phi}_i^\top \mathbf{C}_{-i}^{-1} \boldsymbol{\phi}_i\right) - \operatorname{tr}\left(\boldsymbol{\Omega}^{-1} \mathbf{T}^\top \frac{\mathbf{C}_{-i}^{-1} \boldsymbol{\phi}_i \boldsymbol{\phi}_i^\top \mathbf{C}_{-i}^{-1}}{\alpha_i + \boldsymbol{\phi}_i^\top \mathbf{C}_{-i}^{-1} \boldsymbol{\phi}_i} \mathbf{T}\right)\right) \\
=& \mathcal{L}(\boldsymbol{\alpha}_{-i}, \boldsymbol{\Omega}) + \frac{1}{2} \left(V \log \alpha_i - V \log(\alpha_i + s_i) + \frac{\operatorname{tr}\left(\boldsymbol{\Omega}^{-1} \mathbf{q}_i^\top \mathbf{q}_i\right)}{\alpha_i + s_i}\right) \\
=& \mathcal{L}(\boldsymbol{\alpha}_{-i}, \boldsymbol{\Omega}) + \ell(\alpha_i, \boldsymbol{\Omega}),
\end{aligned} \tag{4.15}$$

where s_i and $\mathbf{q}_i \in \mathbb{R}^{1\times V}$ are defined as, respectively:

$$s_i \overset{\text{def}}{=} \boldsymbol{\phi}_i^\top \mathbf{C}_{-i}^{-1} \boldsymbol{\phi}_i, \tag{4.16a}$$

$$\mathbf{q}_i \overset{\text{def}}{=} \boldsymbol{\phi}_i^\top \mathbf{C}_{-i}^{-1} \mathbf{T}. \tag{4.16b}$$

To avoid the matrix inversion of $\mathbf{C}_{-i}$ in Eq. (4.16), which requires the time complexity of $O\left(N^3\right)$, s_i' and $\mathbf{q}_i' \in \mathbb{R}^{1\times V}$ are computed as (by the Woodbury matrix identity):[1]

$$\begin{aligned} s_i' &= \boldsymbol{\phi}_i^\top \mathbf{C}^{-1} \boldsymbol{\phi}_i \\ &= \boldsymbol{\phi}_i^\top \boldsymbol{\phi}_i - \boldsymbol{\phi}_i^\top \boldsymbol{\Phi} \boldsymbol{\Sigma} \boldsymbol{\Phi}^\top \boldsymbol{\phi}_i, \end{aligned} \tag{4.17a}$$

$$\begin{aligned} \mathbf{q}_i' &= \boldsymbol{\phi}_i^\top \mathbf{C}^{-1} \mathbf{T} \\ &= \boldsymbol{\phi}_i^\top \mathbf{T} - \boldsymbol{\phi}_i^\top \boldsymbol{\Phi} \boldsymbol{\Sigma} \boldsymbol{\Phi}^\top \mathbf{T}, \end{aligned} \tag{4.17b}$$

and then s_i and $\mathbf{q}_i$ in Eq. (4.16) are computed as:[2]

$$s_i = \frac{\alpha_i s_i'}{\alpha_i - s_i'}, \tag{4.18a}$$

$$\mathbf{q}_i = \frac{\alpha_i \mathbf{q}_i'}{\alpha_i - s_i'}. \tag{4.18b}$$

[1] $s_i' = \boldsymbol{\phi}_i^\top \boldsymbol{\phi}_i$ and $\mathbf{q}_i' = \boldsymbol{\phi}_i^\top \mathbf{T}$ when $\alpha_i = \infty, \forall i$.
[2] $s_i = s_i'$ and $\mathbf{q}_i = \mathbf{q}_i'$ when $\alpha_i = \infty$.

$\dfrac{\partial \ell(\alpha_i, \boldsymbol{\Omega})}{\partial \alpha_i} = 0$ leads to that $\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\Omega})$ has a unique maximum with respect to α_i when:

$$\alpha_i = \begin{cases} \dfrac{s_i^2}{\dfrac{\operatorname{tr}\left(\boldsymbol{\Omega}^{-1}\mathbf{q}_i^\top \mathbf{q}_i\right)}{V} - s_i}, & \text{if } \dfrac{\operatorname{tr}\left(\boldsymbol{\Omega}^{-1}\mathbf{q}_i^\top \mathbf{q}_i\right)}{V} > s_i \\ \infty, & \text{if } \dfrac{\operatorname{tr}\left(\boldsymbol{\Omega}^{-1}\mathbf{q}_i^\top \mathbf{q}_i\right)}{V} \le s_i \end{cases}, \tag{4.19}$$

which implies that:

- If $\boldsymbol{\phi}_i$ is "in the model" (i.e. $\alpha_i < \infty$) and $\dfrac{\operatorname{tr}\left(\boldsymbol{\Omega}^{-1}\mathbf{q}_i^\top \mathbf{q}_i\right)}{V} > s_i$; then, α_i is re-estimated,
- If $\boldsymbol{\phi}_i$ is "in the model" (i.e. $\alpha_i < \infty$) yet $\dfrac{\operatorname{tr}\left(\boldsymbol{\Omega}^{-1}\mathbf{q}_i^\top \mathbf{q}_i\right)}{V} \le s_i$; then, $\boldsymbol{\phi}_i$ may be deleted (i.e. α_i is set to be ∞),
- If $\boldsymbol{\phi}_i$ is "out of the model" (i.e. $\alpha_i = \infty$) yet $\dfrac{\operatorname{tr}\left(\boldsymbol{\Omega}^{-1}\mathbf{q}_i^\top \mathbf{q}_i\right)}{V} > s_i$; then, $\boldsymbol{\phi}_i$ may be added (i.e. α_i is set to be a finite value).

In addition, $\dfrac{\partial \mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\Omega})}{\partial \boldsymbol{\Omega}} = \mathbf{0}$, where $\mathbf{0}$ is a $V \times V$ zero matrix, leads to that $\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\Omega})$ has a unique maximum with respect to $\boldsymbol{\Omega}$ when:

$$\boldsymbol{\Omega} = \frac{\mathbf{T}^\top(\mathbf{T} - \boldsymbol{\Phi}\mathbf{M})}{N}. \tag{4.20}$$

4.4. *Expectation–maximisation (EM) algorithm*

Algorithm 2, an EM algorithm to maximise the marginal likelihood, starts without any basis vector (i.e. $M = 0$) and selects the basis vector $\boldsymbol{\phi}_i$ which gives the maximum change of the marginal likelihood $\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\Omega})$ of Eq. (4.10) at every iteration.

For efficient computation of the EM algorithm, quantities $\boldsymbol{\Phi} \in \mathbb{R}^{N\times M}$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{M\times M}$ contain only M ($M \le N+1$ is always satisfied) basis functions that are currently included in the model (i.e. $\boldsymbol{\phi}_i$ which satisfies $\alpha_i < \infty$), and the diagonal matrix $\mathbf{A}$ consists of M hyperparameters of α_i that are currently included in the model (i.e. α_i which satisfies $\alpha_i < \infty$).

From Eq. (4.10), the change in the marginal likelihood can be written as

$$\begin{aligned} 2\Delta\mathcal{L} &= 2\left(\mathcal{L}(\tilde{\boldsymbol{\alpha}}, \boldsymbol{\Omega}) - \mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\Omega})\right) \\ &= V \log \frac{|\mathbf{C}|}{|\tilde{\mathbf{C}}|} + \operatorname{tr}\left(\boldsymbol{\Omega}^{-1}\mathbf{T}^\top\left(\mathbf{C}^{-1} - \tilde{\mathbf{C}}^{-1}\right)\mathbf{T}\right), \end{aligned} \tag{4.21}$$

where updated quantities are denoted by a tilde (e.g., $\tilde{\boldsymbol{\alpha}}$ and $\tilde{\mathbf{C}}$). Eq. (4.21) differs according to whether α_i is re-estimated, added, or deleted:

Re-estimation. as $\mathbf{C} = \mathbf{C}_{-i} + \alpha_i^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top$ and $\tilde{\mathbf{C}} = \mathbf{C}_{-i} + \tilde{\alpha}_i^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top$,

$$2\Delta\mathcal{L}_i = \frac{\operatorname{tr}\left(\boldsymbol{\Omega}^{-1}\mathbf{q}_i'^\top \mathbf{q}_i'\right)}{s_i' + \left(\tilde{\alpha}_i^{-1} - \alpha_i^{-1}\right)^{-1}} - V\log\left(1 + s_i'\left(\tilde{\alpha}_i^{-1} - \alpha_i^{-1}\right)\right), \tag{4.22}$$

where $\tilde{\alpha}_i$ is re-estimated α_i,

Algorithm 2: Proposed EM algorithm of MRVR.

```
Input: $\mathbf{T} \in \mathbb{R}^{N\times V}$ and $\boldsymbol{\phi}_i \in \mathbb{R}^{N\times 1}, \forall i = \{0,1,\dots,N\}$, where $N$ is the number of training samples, and $V$ is the number
       of output dimensions
Output: $\boldsymbol{\Sigma} \in \mathbb{R}^{M\times M}$, $\mathbf{M} \in \mathbb{R}^{M\times V}$, and $\boldsymbol{\Omega} \in \mathbb{R}^{V\times V}$, where $M$ is the number of basis functions in the model
// Initialisation
1  $\alpha_i \leftarrow \infty, \forall i = \{0,1,\dots,N\}$
2  $\bar{\mathbf{t}} \leftarrow \frac{1}{N}\sum_{i=1}^{N}\mathbf{t}_i$, where $\bar{\mathbf{t}} \in \mathbb{R}^{1\times V}$, and $\mathbf{t}_i \in \mathbb{R}^{1\times V}$ is the $i$-th row vector of $\mathbf{T}$.
3  $\boldsymbol{\Omega} \leftarrow \frac{0.1}{N-1}\sum_{i=1}^{N}(\mathbf{t}_i - \bar{\mathbf{t}})^\top(\mathbf{t}_i - \bar{\mathbf{t}})$, where $\boldsymbol{\Omega} \in \mathbb{R}^{V\times V}$ is a covariance matrix.
4  convergence←false
5  $n \leftarrow 1$, where $n$ is the iteration number
6  $M \leftarrow 0$, where $M$ is the number of basis functions.
7  while convergence=false do
       // maximisation step
8      for $i \leftarrow 0$ to $N$ do
9          Update $s_i'$ and $\mathbf{q}_i'$ using Eq. (4.17).
10         Update $s_i$ and $\mathbf{q}_i$ using Eq. (4.18).
11         $\theta_i \leftarrow \frac{\mathrm{tr}\left(\boldsymbol{\Omega}^{-1}\mathbf{q}_i^\top\mathbf{q}_i\right)}{V} - s_i$
12         if $\theta_i > 0$ then
13             if $\alpha_i < \infty$ then $z_i \leftarrow$‘re-estimation’
14                 $\tilde{\alpha}_i \leftarrow \frac{s_i^2}{\theta_i}$
15                 Update $2\Delta\mathcal{L}_i$ using Eq. (4.22).
16             else $z_i \leftarrow$‘addition’
17                 Update $2\Delta\mathcal{L}_i$ using Eq. (4.23).
18             end
19         else if $\alpha_i < \infty$ then$z_i \leftarrow$‘deletion’
20             Update $2\Delta\mathcal{L}_i$ using Eq. (4.24).
21         else
22             $2\Delta\mathcal{L}_i \leftarrow -\infty$
23         end
24     end
25     $i \leftarrow \arg\max_i 2\Delta\mathcal{L}_i$ // Select $i$ which gives the greatest increase of the marginal likelihood
26     switch $z_i$ do
27         case ‘re-estimation’ do
               // Check convergence
28             $\Delta\log\alpha \leftarrow \log\frac{\alpha_i}{\tilde{\alpha}_i}$
29             if $|\Delta\log\alpha| < 0.1$ then
30                 convergence←true
31                 for $i \leftarrow 0$ to $N$ do
32                     if $\alpha_i = \infty$ then // if $\boldsymbol{\phi}_i$ is "out of the model"
33                         if $\theta_i > 0$ then // if $\boldsymbol{\phi}_i$ may be added "in the model"
34                             convergence←false
35                             break for loop
36                         end
37                     end
38                 end
39             end
40             $\alpha_i \leftarrow \tilde{\alpha}_i$
41         case ‘addition’ do
42             $\alpha_i \leftarrow \frac{s_i^2}{\theta_i}$
43             $M \leftarrow M + 1$
44         case ‘deletion’ do
45             $\alpha_i \leftarrow \infty$
46             $M \leftarrow M - 1$
47     end
48     if $n \neq 1$ then
49         Update $\boldsymbol{\Omega}$ using Eq. (4.20).
50     end
       // Expectation step
51     Sequentially update i) $\boldsymbol{\Phi} \in \mathbb{R}^{N\times M}$, $\mathbf{A} \in \mathbb{R}^{M\times M}$, ii) $\boldsymbol{\Sigma} \in \mathbb{R}^{M\times M}$, and iii) $\mathbf{M} \in \mathbb{R}^{M\times V}$ using Eq. (4.7) and Eq. (4.8),
         where $\boldsymbol{\Phi}$, $\boldsymbol{\Sigma}$, and $\mathbf{M}$ contain only $M$ basis functions that are currently included in the model, and the diagonal
         matrix $\mathbf{A}$ consists of $M$ hyperparameters of $\alpha_i$ that are currently included in the model.
52     $n \leftarrow n + 1$
53 end
```

Addition. as $\mathbf{C} = \mathbf{C}_{-i}$ and $\tilde{\mathbf{C}} = \mathbf{C}_{-i} + \tilde{\alpha}_i^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top$, where $\tilde{\alpha}_i = \dfrac{s_i^2}{\dfrac{\operatorname{tr}\left(\boldsymbol{\Omega}^{-1}\mathbf{q}_i^\top\mathbf{q}_i\right)}{V} - s_i}$,

$$2\Delta\mathcal{L}_i = \frac{\operatorname{tr}\left(\boldsymbol{\Omega}^{-1}\mathbf{q}_i'^\top\mathbf{q}_i'\right) - Vs_i'}{s_i'} + V\log\frac{Vs_i'}{\operatorname{tr}\left(\boldsymbol{\Omega}^{-1}\mathbf{q}_i'^\top\mathbf{q}_i'\right)}, \tag{4.23}$$

Deletion. as $\mathbf{C} = \mathbf{C}_{-i} + \alpha_i^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top$ and $\tilde{\mathbf{C}} = \mathbf{C}_{-i}$,

$$2\Delta\mathcal{L}_i = \frac{\operatorname{tr}\left(\boldsymbol{\Omega}^{-1}\mathbf{q}_i'^\top\mathbf{q}_i'\right)}{s_i' - \alpha_i} - V\log\left(1 - \frac{s_i'}{\alpha_i}\right). \tag{4.24}$$

The convergence criteria of the EM algorithm are presented between the 28-th line and the 39-th line of Algorithm 2. The logarithmic change $|\Delta \log\alpha|$ should be less than a tolerance, and any basis vector $\boldsymbol{\phi}_i$ "out of the model" for the current iteration should not be added "in the model" for the next iteration. These are the same as those in Tipping and Faul (2003).

4.5. *Making predictions*

We can predict both a mean vector $\mathbf{y}_* \in \mathbb{R}^{1\times V}$ and a covariance matrix $\boldsymbol{\Omega}_* \in \mathbb{R}^{V\times V}$ from a new input vector $\mathbf{x}_* \in \mathbb{R}^{U\times 1}$ based on both i) Eq. (4.2), the model specification, and ii) Eq. (4.6), the posterior distribution over the weights, conditioned on the most probable (MP) hyperparameters: $\boldsymbol{\alpha}_{\text{MP}} \in \mathbb{R}_{>0}^{M\times 1}$ and $\boldsymbol{\Omega}_{\text{MP}} \in \mathbb{R}^{V\times V}$, obtained from Algorithm 2. Predictive distribution of $\mathbf{t}_*$ is jointly normally distributed as

$$p(\mathbf{t}_*|\mathbf{T}, \boldsymbol{\alpha}_{\text{MP}}, \boldsymbol{\Omega}_{\text{MP}}) = \mathcal{N}(\mathbf{t}_*|\mathbf{y}_*, \boldsymbol{\Omega}_*), \tag{4.25}$$

with

$$\mathbf{y}_* = \boldsymbol{\phi}(\mathbf{x}_*)^\top\mathbf{M}, \tag{4.26}$$

and

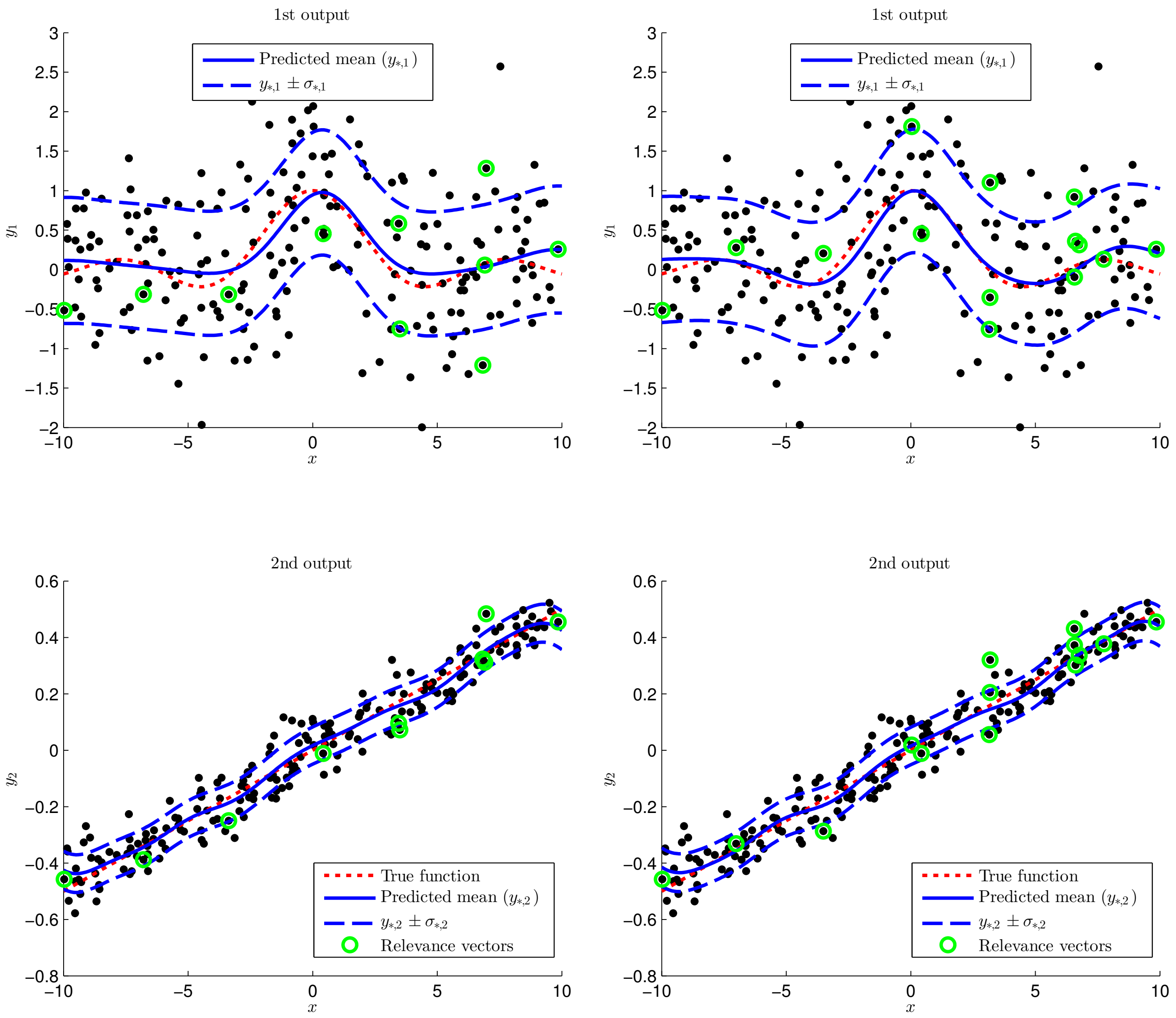
$$\boldsymbol{\Omega}_* = \boldsymbol{\Omega}_{\text{MP}}\left(1 + \boldsymbol{\phi}(\mathbf{x}_*)^\top\boldsymbol{\Sigma}\boldsymbol{\phi}(\mathbf{x}_*)\right), [1] \tag{4.27}$$

where $\boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^{M\times 1}$ comes from only M basis functions that are included in the model after the EM algorithm. The predictive covariance matrix $\boldsymbol{\Omega}_*$ comprises the two components: the estimated noise on the training data $\boldsymbol{\Omega}_{\text{MP}}$ and that due to the uncertainty in the prediction of the weights $\boldsymbol{\Omega}_{\text{MP}}\boldsymbol{\phi}(\mathbf{x}_*)^\top\boldsymbol{\Sigma}\boldsymbol{\phi}(\mathbf{x}_*)$, where $\boldsymbol{\phi}(\mathbf{x}_*)^\top\boldsymbol{\Sigma}\boldsymbol{\phi}(\mathbf{x}_*) \in \mathbb{R}_{\geq 0}$ by the fact that a covariance matrix is always positive semidefinite. They share $\boldsymbol{\Omega}_{\text{MP}}$ by the assumption of $\boldsymbol{\Omega} = \frac{\mathbb{E}[\mathbf{E}^\top\mathbf{E}]}{N} = \frac{\mathbb{E}[\mathbf{W}^\top\mathbf{W}]}{\operatorname{tr}(\mathbf{A}^{-1})}$ in Section 4.1.

4.6. *Algorithm complexity*

Matrix inversion of $\boldsymbol{\Omega} \in \mathbb{R}^{V\times V}$ in Eq. (4.19) and that of the $M \times M$ matrix in Eq. (4.7) determine i) the time complexity of the proposed algorithm as $O\left(V^3 + M^3\right)$ and ii) the memory complexity

[1] Eq. (4.27) is obtained by the property that the covariance between two elements $W_{i,j}$ and $W_{i',j'}$ is the covariance between the rows i and i', i.e. $\boldsymbol{\Sigma}$, multiplied by the covariance between the columns j and j', i.e. $\boldsymbol{\Omega}_{\text{MP}}$ (Arnold 1981, p. 311).



(a) Existing method ($\sigma^2_{*,j}$ is equal to Eq. (3.29))

(b) Proposed method ($\sigma^2_{*,j}$ is equal to the j-th diagonal element of $\mathbf{\Omega}_*$ in Eq. (4.27))

Figure 1. An example of MRVR (when $U = 1, V = 2, N = 200$, and the Gaussian kernel $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\lambda^2}\right)$ with a free parameter $\lambda = 1.6$ is used)

as $O\left(V^2 + M^2\right)$, where V is the number of output dimensions, and M is the number of basis functions.[1]

5. Experimental results

5.1. *An example of MRVR*

Fig. 1 shows an example of the MRVR results obtained using the two methods when the true function of each output dimension is the sinc function and the linear function, respectively. Fig. 1(a) and Fig. 1(b) show slightly different results although the same training samples are used.

[1] The matrix multiplication to calculate s'_i and $\mathbf{q}'_i$ in Eq. (4.17) for all $i \in \{1, 2, \ldots, N\}$ at the 9-th line of Algorithm 2 does not influence the time complexity because the matrix multiplication $\mathbf{\Phi\Sigma\Phi}^\top$ is pre-calculated. In other words, the time complexity of the matrix multiplication is $O\left(M^3\right)$, not $O\left(NM^3\right)$, because $\mathbf{\Phi\Sigma\Phi}^\top$ is independent of i.

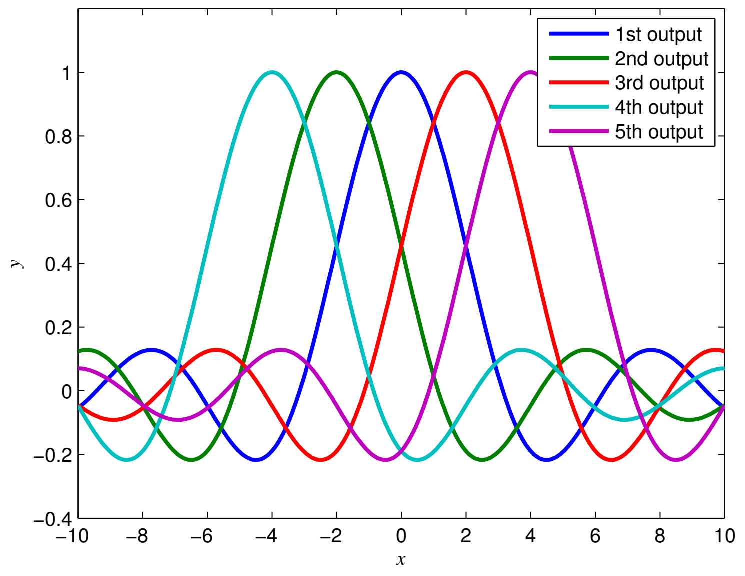


Figure 2. True functions of MC simulations

5.2. *Comparisons of the performance*

The two methods are compared in terms of i) running time (computation time in INTEL® Core™ i5-3470 CPU and MATLAB® R2013b), ii) the estimation accuracy of the noise covariance matrix, iii) root-mean-square error (RMSE) between true functions and predicted mean values, and iv) the number of relevance vectors (RVs), where RVs are those training vectors associated with the remaining non-zero weights.[1]

To measure the estimation accuracy of the noise covariance matrix $\mathbf{\Omega}$, entropy loss $L_1\left(\mathbf{\Omega}, \hat{\mathbf{\Omega}}\right)$ and quadratic loss $L_2\left(\mathbf{\Omega}, \hat{\mathbf{\Omega}}\right)$ are used (each of these is 0 when $\hat{\mathbf{\Omega}} = \mathbf{\Omega}$ and is positive when $\hat{\mathbf{\Omega}} \neq \mathbf{\Omega}$) (Anderson 1984, pp. 273–274):

$$L_1\left(\mathbf{\Omega}, \hat{\mathbf{\Omega}}\right) = \operatorname{tr}\left(\hat{\mathbf{\Omega}}\mathbf{\Omega}^{-1}\right) - \log\left|\hat{\mathbf{\Omega}}\mathbf{\Omega}^{-1}\right| - V, \tag{5.1}$$

$$L_2\left(\mathbf{\Omega}, \hat{\mathbf{\Omega}}\right) = \operatorname{tr}\left(\left(\hat{\mathbf{\Omega}}\mathbf{\Omega}^{-1} - \mathbf{I}\right)^2\right), \tag{5.2}$$

where the estimated $V \times V$ covariance matrix of the noise $\hat{\mathbf{\Omega}}$ is $\mathbf{\Omega}_{\mathrm{MP}}$ in the case of the proposed method. In the case of the existing method, $\hat{\mathbf{\Omega}}$ can be obtained using both i) the estimated standard deviation of the noise $\hat{\mathbf{D}} = \operatorname{diag}(\sigma_{\mathrm{MP},1}, \sigma_{\mathrm{MP},2}, \ldots, \sigma_{\mathrm{MP},V})$ in Section 3.5 and ii) the estimated correlation matrix of the noise $\hat{\mathbf{R}}$:

$$\hat{\mathbf{\Omega}} = \hat{\mathbf{D}}\hat{\mathbf{R}}\hat{\mathbf{D}}, \tag{5.3}$$

where $\hat{\mathbf{R}} = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{\Omega}}\tilde{\mathbf{D}}^{-1}$, $\tilde{\mathbf{\Omega}} = \dfrac{(\mathbf{T} - \mathbf{\Phi}\tilde{\mathbf{M}})^{\top}(\mathbf{T} - \mathbf{\Phi}\tilde{\mathbf{M}})}{N-1}$, $\tilde{\mathbf{M}} = [\boldsymbol{\mu}_1 \ \boldsymbol{\mu}_2 \ \ldots \ \boldsymbol{\mu}_V] \in \mathbb{R}^{M \times V}$, and $\tilde{\mathbf{D}} = \sqrt{\operatorname{diag}(\tilde{\mathbf{\Omega}})}$.

Monte Carlo (MC) simulations with random covariance matrices of the noise were conducted for the performance comparisons. The noise from the random covariance matrix is added to the true

[1] The MATLAB codes of the experiment have been uploaded on http://www.mathworks.com/matlabcentral/fileexchange/49131 to avoid any potential ambiguity of both the methods.

Table 1. The number of rejections of the null hypothesis of the Jarque–Bera test

	Running time	Entropy loss	Quadratic loss	RMSE	The number of RVs
Existing method	30	30	30	4	16
Proposed method	30	28	29	3	14

Table 2. The difference in median values of running time (seconds)

	$N = 50$	$N = 100$	$N = 150$	$N = 200$	$N = 250$	$N = 300$
$V = 1$	**0.04** (3.6×10^{-10})	**0.19** (2.1×10^{-9})	**0.40** (4.9×10^{-9})	**1.02** (3.2×10^{-9})	**3.11** (1.8×10^{-20})	**5.95** (7.1×10^{-18})
$V = 2$	**0.20** (4.8×10^{-25})	**0.76** (1.5×10^{-22})	**2.44** (9.0×10^{-26})	**5.95** (2.1×10^{-25})	**17.20** (1.3×10^{-30})	**26.17** (1.0×10^{-31})
$V = 3$	**0.33** (3.2×10^{-26})	**1.70** (3.8×10^{-32})	**5.27** (5.4×10^{-29})	**10.37** (1.1×10^{-31})	**37.74** (4.7×10^{-34})	**64.99** (2.2×10^{-34})
$V = 4$	**0.74** (2.2×10^{-33})	**3.60** (1.3×10^{-32})	**9.69** (2.1×10^{-33})	**24.51** (2.4×10^{-33})	**57.14** (2.1×10^{-34})	**112.18** (1.2×10^{-34})
$V = 5$	**1.05** (1.1×10^{-33})	**5.25** (5.4×10^{-34})	**13.01** (6.3×10^{-34})	**34.09** (2.2×10^{-34})	**92.96** (1.2×10^{-34})	**176.24** (1.2×10^{-34})

functions in Fig. 2 (each output has a sinc function with a translation in the x-axis), and the two methods of MRVR with the Gaussian kernel were performed using the same training samples for a fair comparison.

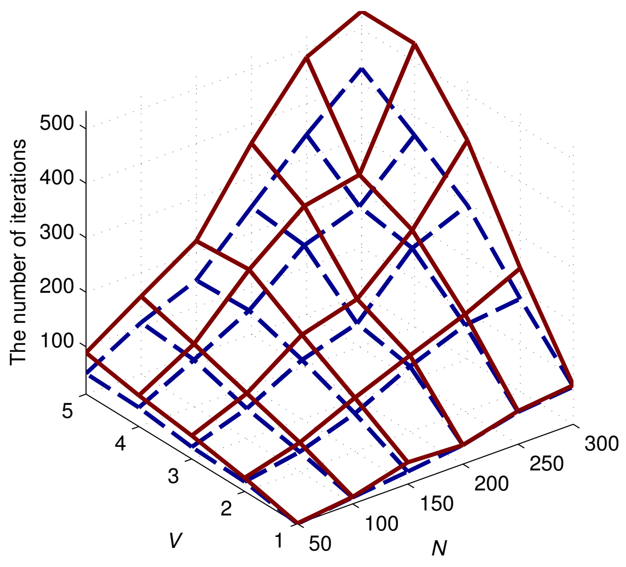
Unpaired two-sample t-tests may be used to compare the two methods to determine whether the performance difference is fundamental or whether it is due to random fluctuations (Simon 2013, pp. 631–635), but the normality assumption of the performance measures (i.e. running time, entropy loss, quadratic loss, RMSE, and the number of RVs) of the two methods must be checked. The Jarque–Bera test with a significance level of 5% for 30 cases ($V = \{1, 2, 3, 4, 5\}$ and $N = \{50, 100, 150, 200, 250, 300\}$) was conducted. Table 1 shows the number of rejections of the null hypothesis of the Jarque–Bera test. Consequently, the t-test can yield misleading results in the case that the null hypothesis of a normal distribution is rejected.

Instead of the t-test, two-sided Wilcoxon rank sum tests, whose null hypothesis is that two populations have equal median values, were used for the comparisons as they have greater efficiency than the t-test on non-normal distributions and are nearly as efficient as the t-test on normal distributions (Montgomery 2013, Chapter 10).

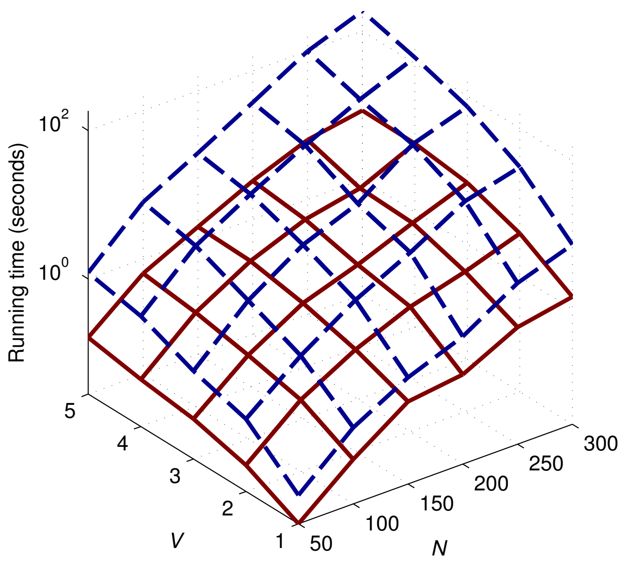
Fig. 3 shows the median values of the performance measures of both methods with various V and N values. Entropy loss, quadratic loss, and RMSE decrease as N increases: the greater the number of training samples, the more accurate the estimation. In contrast, the number of RVs, the number of iterations of each EM algorithm (the same tolerance value of 0.1 for checking the convergence of each EM algorithm was used as in 43th line of Algorithm 1 and 30th line of Algorithm 2), and the running time (only for learning without prediction) of each EM algorithm increase as N increases: the greater the number of training samples, the greater the computational burden.

Tables 2–6 show i) the difference in the median values of the performance measures, where each median value is obtained from 101 MC simulations, and then the median value of the proposed method is subtracted from that of the existing method (i.e. positive difference values mean that the proposed method is better than the existing method, while negative difference values mean the opposite) and ii) the p-values of the Wilcoxon rank sum tests, which appear inside the brackets. The p-value is interpreted as the probability that a difference in the median values would be obtained given that the population medians of two methods are equivalent, i.e. the p-value is not equal to the probability that the population medians are equivalent (Simon 2013, p. 635). Note that statistically significant difference values are marked in bold (p-value < 0.05).

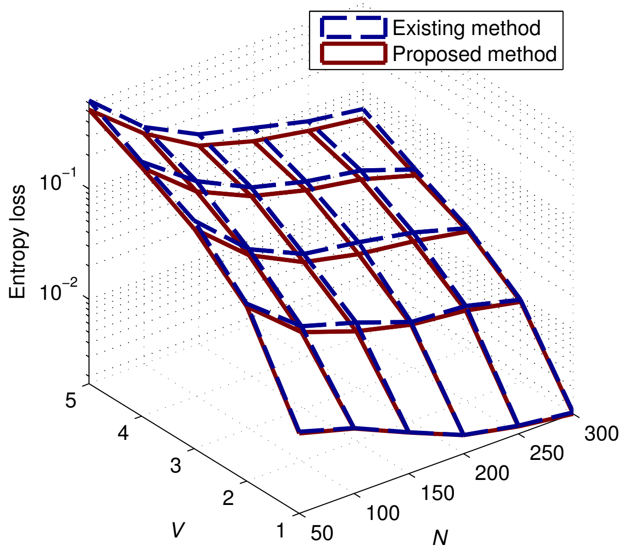
The proposed method is faster than the existing method as shown in Table 2 (all differences are statistically significant). In particular, the time difference is amplified as V or N increases. This is because the time complexity of the proposed method $O\left(V^3 + M^3\right)$ is less than that of the



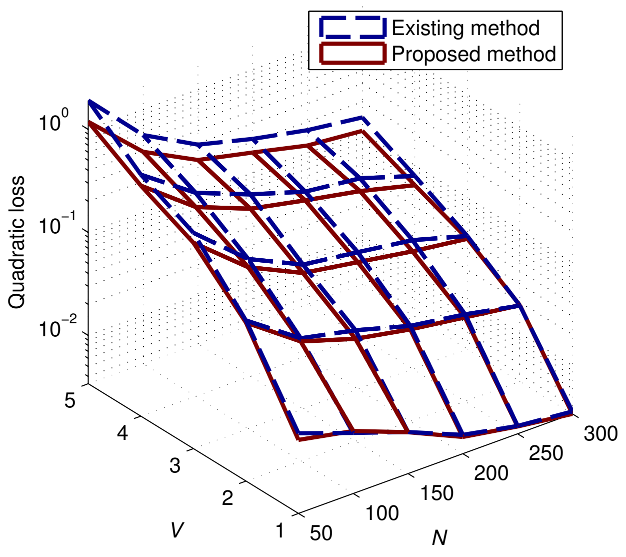
(a) The number of iterations



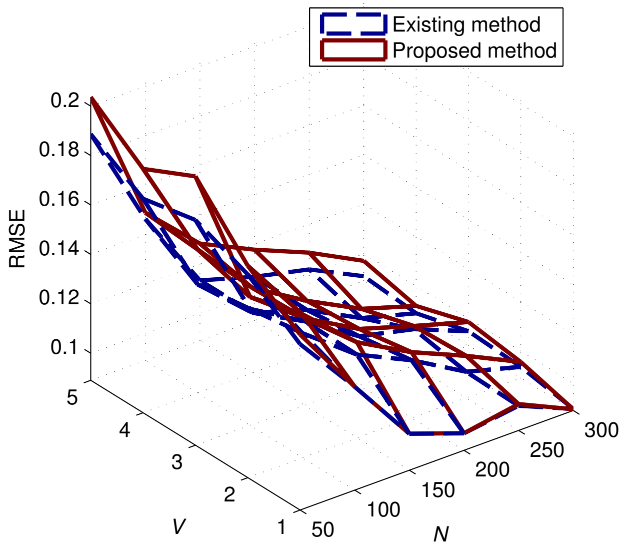
(b) Running time



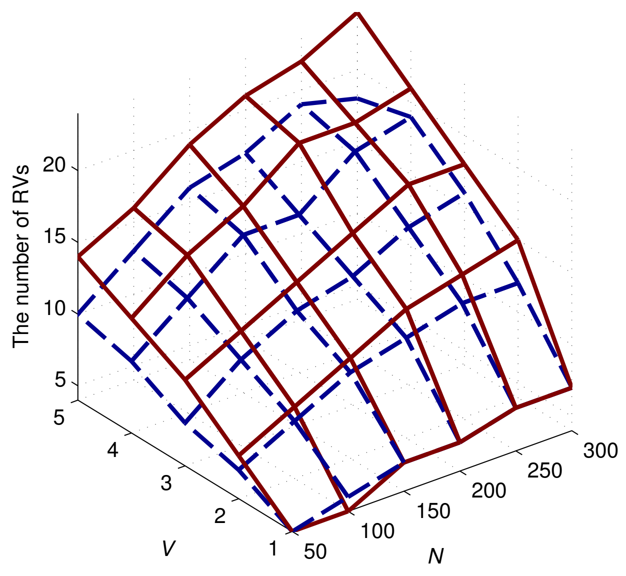
(c) Entropy loss



(d) Quadratic loss



(e) RMSE



(f) The number of RVs

Figure 3. Median values of MC simulations (the number of simulations is 101)

Table 3. The difference in median values of entropy loss

	$N = 50$	$N = 100$	$N = 150$	$N = 200$	$N = 250$	$N = 300$
$V = 1$	3.4×10^{-4}	8.0×10^{-5}	2.2×10^{-5}	1.7×10^{-5}	3.7×10^{-5}	5.7×10^{-5}
	(0.9904)	(0.9176)	(0.9962)	(0.9520)	(0.8454)	(0.9808)
$V = 2$	3.9×10^{-3}	3.3×10^{-3}	3.4×10^{-3}	6.8×10^{-4}	1.1×10^{-3}	4.0×10^{-4}
	(0.4942)	(0.5964)	(0.1132)	(0.3915)	(0.7434)	(0.6234)
$V = 3$	3.2×10^{-2}	8.6×10^{-3}	$\mathbf{6.8 \times 10^{-3}}$	$\mathbf{8.0 \times 10^{-3}}$	$\mathbf{5.1 \times 10^{-3}}$	$\mathbf{1.4 \times 10^{-3}}$
	(0.0883)	(0.2469)	(0.0192)	(0.0069)	(0.0119)	(0.0412)
$V = 4$	3.9×10^{-2}	$\mathbf{3.0 \times 10^{-2}}$	$\mathbf{1.5 \times 10^{-2}}$	$\mathbf{1.1 \times 10^{-2}}$	$\mathbf{9.3 \times 10^{-3}}$	$\mathbf{4.4 \times 10^{-3}}$
	(0.1137)	(0.0030)	(0.0010)	(0.0042)	(0.0063)	(0.0058)
$V = 5$	$\mathbf{1.1 \times 10^{-1}}$	$\mathbf{3.0 \times 10^{-2}}$	$\mathbf{2.9 \times 10^{-2}}$	$\mathbf{2.5 \times 10^{-2}}$	$\mathbf{1.4 \times 10^{-2}}$	$\mathbf{1.2 \times 10^{-2}}$
	(0.0231)	(0.0015)	(0.0000)	(0.0008)	(0.0005)	(0.0018)

Table 4. The difference in median values of quadratic loss

	$N = 50$	$N = 100$	$N = 150$	$N = 200$	$N = 250$	$N = 300$
$V = 1$	2.8×10^{-3}	-5.7×10^{-4}	4.9×10^{-5}	2.3×10^{-4}	4.4×10^{-5}	2.0×10^{-4}
	(0.9981)	(0.9233)	(0.9981)	(0.9405)	(0.8605)	(0.9770)
$V = 2$	5.5×10^{-3}	3.5×10^{-3}	7.0×10^{-3}	2.2×10^{-3}	1.6×10^{-3}	-1.9×10^{-4}
	(0.5080)	(0.4526)	(0.0915)	(0.3616)	(0.6824)	(0.5782)
$V = 3$	$\mathbf{8.5 \times 10^{-2}}$	2.5×10^{-2}	$\mathbf{1.4 \times 10^{-2}}$	$\mathbf{1.4 \times 10^{-2}}$	$\mathbf{1.3 \times 10^{-2}}$	$\mathbf{2.7 \times 10^{-3}}$
	(0.0180)	(0.1154)	(0.0109)	(0.0029)	(0.0051)	(0.0263)
$V = 4$	$\mathbf{1.7 \times 10^{-1}}$	$\mathbf{8.6 \times 10^{-2}}$	$\mathbf{5.4 \times 10^{-2}}$	$\mathbf{2.4 \times 10^{-2}}$	$\mathbf{2.9 \times 10^{-2}}$	$\mathbf{1.5 \times 10^{-2}}$
	(0.0382)	(0.0005)	(0.0001)	(0.0007)	(0.0019)	(0.0013)
$V = 5$	$\mathbf{7.3 \times 10^{-1}}$	$\mathbf{1.8 \times 10^{-1}}$	$\mathbf{8.5 \times 10^{-2}}$	$\mathbf{5.5 \times 10^{-2}}$	$\mathbf{4.8 \times 10^{-2}}$	$\mathbf{3.7 \times 10^{-2}}$
	(0.0058)	(0.0000)	(0.0000)	(0.0001)	(0.0000)	(0.0004)

Table 5. The difference in median values of RMSE

	$N = 50$	$N = 100$	$N = 150$	$N = 200$	$N = 250$	$N = 300$
$V = 1$	-0.0038	-0.0000	0.0000	-0.0001	-0.0007	0.0001
	(0.956)	(0.985)	(0.967)	(1.000)	(0.965)	(0.987)
$V = 2$	-0.0035	-0.0076	-0.0051	-0.0033	-0.0064	-0.0020
	(0.898)	(0.544)	(0.562)	(0.758)	(0.646)	(0.977)
$V = 3$	-0.0177	-0.0049	-0.0012	-0.0030	-0.0013	-0.0037
	(0.182)	(0.299)	(0.408)	(0.661)	(0.546)	(0.565)
$V = 4$	**-0.0121**	**-0.0142**	**-0.0101**	-0.0037	-0.0038	-0.0029
	(0.041)	(0.034)	(0.017)	(0.092)	(0.115)	(0.293)
$V = 5$	**-0.0150**	**-0.0026**	**-0.0149**	**-0.0110**	**-0.0068**	-0.0062
	(0.008)	(0.013)	(0.001)	(0.015)	(0.025)	(0.072)

Table 6. The difference in median values of the number of RVs

	$N = 50$	$N = 100$	$N = 150$	$N = 200$	$N = 250$	$N = 300$
$V = 1$	0	1	0	0	0	0
	(8.8×10^{-1})	(9.0×10^{-1})	(8.4×10^{-1})	(9.6×10^{-1})	(8.9×10^{-1})	(9.9×10^{-1})
$V = 2$	-1	**-1**	**-1**	**-2**	**-2**	**-3**
	(5.2×10^{-2})	(4.8×10^{-4})	(3.2×10^{-3})	(5.1×10^{-4})	(1.5×10^{-3})	(2.2×10^{-3})
$V = 3$	**-3**	**-2**	**-2**	**-3**	**-3**	**-2**
	(3.6×10^{-7})	(3.9×10^{-5})	(1.5×10^{-5})	(3.4×10^{-7})	(3.2×10^{-6})	(2.6×10^{-6})
$V = 4$	**-3**	**-3**	**-2**	**-5**	**-2**	**-2**
	(2.2×10^{-8})	(4.8×10^{-6})	(8.6×10^{-5})	(2.4×10^{-9})	(2.2×10^{-7})	(1.8×10^{-4})
$V = 5$	**-4**	**-3**	**-3**	**-4**	**-3**	**-6**
	(3.9×10^{-15})	(3.9×10^{-10})	(3.9×10^{-8})	(9.0×10^{-10})	(6.9×10^{-7})	(3.2×10^{-7})

existing method $O\left(VM^3\right)$ ($O\left(V^3 + M^3\right) < O\left(VM^3\right)$ is satisfied since $V < M$ is satisfied in most applications). Note that even when the number of input dimensions U changes, the size of the design matrix $\mathbf{\Phi}$ does not change. Hence, U does not influence the time complexity of both the methods.

Furthermore, the proposed method achieves higher accuracy in estimating the covariance matrix of the noise $\mathbf{\Omega}$ than the existing method as shown in Table 3 and Table 4. This is because the proposed method considers the correlation matrix of the noise as Eq. (4.3), but the existing method does not as Eq. (4.3).

However, the proposed method is worse than the existing one in terms of i) the accuracy in predicting the mean values as shown in Table 5 (in particular, the RMSE increases in the region of high V and low N) and ii) the number of RVs as shown in Table 6. This is because the proposed method has the assumption of the weight $\mathbf{\Omega} = \frac{\mathbb{E}[\mathbf{W}^\top \mathbf{W}]}{\mathrm{tr}(\mathbf{A}^{-1})}$, which behaves as the constraint of the weight. Consequently, the mean values tend to deviate from the true functions, and the number of RVs increases.

There is a potential modification to the proposed algorithm to enhance accuracy while maintaining its fast computation. For example, a certain mathematical transformation can be applied to the weight $\mathbf{W}$ before using our method to make the assumption $\mathbf{\Omega} = \frac{\mathbb{E}[\mathbf{W}^\top \mathbf{W}]}{\mathrm{tr}(\mathbf{A}^{-1})} = \frac{\mathbb{E}[\mathbf{E}^\top \mathbf{E}]}{N}$ true, and its inverse transformation should be applied to the weight after conducting fast MRVR.

6. Conclusion

This paper has proposed a new algorithm of MRVR which is more efficient in computing the weight $\mathbf{W}$ and more accurate in estimating the covariance matrix of the noise $\mathbf{\Omega}$ than existing ones. Its computational efficiency and accuracy can be attributed to the different model specifications of the likelihood of the data, as the existing method expresses the likelihood of the training data as the product of the Gaussian distributions in Eq. (3.4) whereas the proposed one expresses it as the matrix Gaussian distribution in Eq. (4.3).

We do not claim that the proposed algorithm is absolutely better than any other existing ones, rather that it is a very computationally efficient regression tool especially with regards to much higher dimensional problems. The proposed method has drawbacks of relatively lower accuracy in estimating the mean of the weight $\mathbf{M}$ in Eq. (4.8) and higher number of RVs than the existing method. This problem is mainly caused by the assumption $\mathbf{\Omega} = \frac{\mathbb{E}[\mathbf{W}^\top \mathbf{W}]}{\mathrm{tr}(\mathbf{A}^{-1})}$ implying a close relation between the weight $\mathbf{W}$ and the noise $\mathbf{E}$ as in Eq. (4.2). One could potentially enhance the accuracy while maintaining high computational efficiency of the proposed method with the help of mathematical transformation of the weight $\mathbf{W}$, which is much challenging and we leave it for further research.

References

Alvarez, M. and Lawrence, N.D., Sparse convolved Gaussian processes for multi-output regression. In *Proceedings of the Advances in Neural Information Processing Systems*, pp. 57--64, 2009.

Anderson, T.W., *An introduction to multivariate statistical analysis* (2 edn), 1984, Wiley.

Arnold, S.F., *The theory of linear models and multivariate analysis*, 1981, Wiley.

Ben-Shimon, D. and Shmilovici, A., Accelerating the relevance vector machine via data partitioning. *Foundations of Computing and Decision Sciences*, 2006, **31**, 27--41.

Bonilla, E.V., Chai, K.M.A. and Williams, C.K.I., Multi-task Gaussian process prediction. In *Proceedings of the Advances in Neural Information Processing Systems*, pp. 153--160, 2007.

Boyle, P. and Frean, M.R., Dependent Gaussian Processes.. In *Proceedings of the Advances in Neural Information Processing Systems*, pp. 217--224, 2004.

Catanzaro, B., Sundaram, N. and Keutzer, K., Fast support vector machine training and classification on graphics processors. In *Proceedings of the 25th international conference on Machine learning*, pp. 104--111, 2008.

Chang, C.C. and Lin, C.J., LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2011, **2**, 27:1--27.

Cheng, L.F., Darnell, G., Chivers, C., Draugelis, M.E., Li, K. and Engelhardt, B.E., Sparse multi-output Gaussian processes for medical time series prediction. Working paper, available at `https://arxiv.org/abs/1703.09112v2`, 2018.

Chu, W., Keerthi, S.S. and Ong, C.J., Bayesian support vector regression using a unified loss function. *IEEE Transactions on Neural Networks*, 2004, **15**, 29--44.

Cortes, C. and Vapnik, V., Support-vector networks. *Machine learning*, 1995, **20**, 273--297.

Gao, J.B., Gunn, S.R., Harris, C.J. and Brown, M., A probabilistic framework for SVM regression and error bar estimation. *Machine Learning*, 2002, **46**, 71--89.

Gibbs, M., Bayesian Gaussian processes for classification and regression. PhD thesis, University of Cambridge, 1997.

Gramacy, R.B., Niemi, J. and Weiss, R.M., Massively parallel approximate Gaussian process regression. *SIAM/ASA Journal on Uncertainty Quantification*, 2014, **2**, 564--584.

Guo, G. and Zhang, J.S., Reducing examples to accelerate support vector regression. *Pattern Recognition Letters*, 2007, **28**, 2173--2183.

Montgomery, D.C., *Applied Statistics and Probability for Engineers* (6 edn), 2013, Wiley.

Pérez-Cruz, F., Camps-Valls, G., Soria-Olivas, E., Pérez-Ruixo, J.J., Figueiras-Vidal, A.R. and Artés-Rodríguez, A., Multi-dimensional function approximation and regression estimation. In *Proceedings of the International Conference on Artificial Neural Networks*, pp. 757--762, 2002.

Schölkopf, B., Smola, A.J., Williamson, R.C. and Bartlett, P.L., New support vector algorithms. *Neural computation*, 2000, **12**, 1207--1245.

Seeger, M., Williams, C. and Lawrence, N., Fast forward selection to speed up sparse Gaussian process regression. In *Proceedings of the Workshop on Artificial Intelligence and Statistics 9*, 2003.

Shen, Y., Ng, A. and Seeger, M., Fast Gaussian process regression using kd-trees. In *Proceedings of the 19th Annual Conference on Neural Information Processing Systems*, 2006.

Simon, D., *Evolutionary optimization algorithms*, 2013, Wiley.

Srinivasan, B.V., Qi, H. and Duraiswami, R., GPUML: Graphical processors for speeding up kernel machines. In *Proceedings of the Workshop on High Performance Analytics - Algorithms, Implementations, and Applications*, 2010.

Thayananthan, A., Template-based pose estimation and tracking of 3D hand motion. PhD thesis, University of Cambridge, 2005.

Thayananthan, A., Navaratnam, R., Stenger, B., Torr, P.H. and Cipolla, R., Pose estimation and tracking using multivariate regression. *Pattern Recognition Letters*, 2008, **29**, 1302--1310.

Ticlavilca, A., Feuz, D.M. and McKee, M., Forecasting agricultural commodity prices using multivariate Bayesian machine learning regression. In *Proceedings of the Conference on Applied Commodity Price Analysis, Forecasting, and Market Risk Management*, 2010.

Tipping, M.E., Sparse Bayesian learning and the relevance vector machine. *The journal of machine learning research*, 2001, **1**, 211--244.

Tipping, M.E. and Faul, A.C., Fast marginal likelihood maximisation for sparse Bayesian models. In *Proceedings of the ninth international workshop on artificial intelligence and statistics*, Vol. 1, 2003.

Tuia, D., Verrelst, J., Alonso, L., Pérez-Cruz, F. and Camps-Valls, G., Multioutput support vector regression for remote sensing biophysical parameter estimation. *IEEE Geoscience and Remote Sensing Letters*, 2011, **8**, 804--808.

Vapnik, V., *The nature of statistical learning theory*, 2000, Springer.

Vazquez, E. and Walter, E., Multi-output support vector regression. In *Proceedings of the 13th IFAC Symposium on System Identification*, pp. 1820--1825, 2003.

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

Yang, D., Liang, G., Jenkins, D.D., Peterson, G.D. and Li, H., High performance relevance vector machine on GPUs. In *Proceedings of the Symposium on Application Accelerators in High-Performance Computing*, 2010.

Appendix: proof of Eq. (4.6) and Eq. (4.9)

$$
\begin{aligned}
& p(\mathbf{W}|\mathbf{T},\boldsymbol{\alpha},\boldsymbol{\Omega})\,p(\mathbf{T}|\boldsymbol{\alpha},\boldsymbol{\Omega}) \\
=& p(\mathbf{T}|\mathbf{W},\boldsymbol{\Omega})\,p(\mathbf{W}|\boldsymbol{\alpha},\boldsymbol{\Omega}) \\
=& \rho\exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Omega}^{-1}(\mathbf{T}-\boldsymbol{\Phi}\mathbf{W})^\top(\mathbf{T}-\boldsymbol{\Phi}\mathbf{W})\right)\right)\exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Omega}^{-1}\mathbf{W}^\top\mathbf{A}\mathbf{W}\right)\right), \\
& \text{where } \rho=(2\pi)^{-\frac{VN}{2}}|\boldsymbol{\Omega}|^{-\frac{N}{2}}(2\pi)^{-\frac{V(N+1)}{2}}|\boldsymbol{\Omega}|^{-\frac{N+1}{2}}|\mathbf{A}|^{\frac{V}{2}} \\
=& \rho\exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Omega}^{-1}\left(\left(\mathbf{T}^\top-\mathbf{W}^\top\boldsymbol{\Phi}^\top\right)(\mathbf{T}-\boldsymbol{\Phi}\mathbf{W})+\mathbf{W}^\top\mathbf{A}\mathbf{W}\right)\right)\right) \\
=& \rho\exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Omega}^{-1}\left(\mathbf{T}^\top\mathbf{T}-\mathbf{T}^\top\boldsymbol{\Phi}\mathbf{W}-\mathbf{W}^\top\boldsymbol{\Phi}^\top\mathbf{T}+\mathbf{W}^\top\boldsymbol{\Sigma}^{-1}\mathbf{W}\right)\right)\right),\ \text{where } \boldsymbol{\Sigma}=\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi}+\mathbf{A}\right)^{-1} \\
=& \rho\exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Omega}^{-1}\left(\mathbf{T}^\top\mathbf{T}-\mathbf{T}^\top\boldsymbol{\Phi}\mathbf{W}+\mathbf{W}^\top\boldsymbol{\Sigma}^{-1}(\mathbf{W}-\mathbf{M})\right)\right)\right),\text{where } \mathbf{M}=\boldsymbol{\Sigma}\boldsymbol{\Phi}^\top\mathbf{T} \\
=& \rho\exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Omega}^{-1}\left(\mathbf{T}^\top\mathbf{T}-\mathbf{T}^\top\boldsymbol{\Phi}\mathbf{W}+\mathbf{W}^\top\boldsymbol{\Sigma}^{-1}(\mathbf{W}-\mathbf{M})-\mathbf{M}^\top\boldsymbol{\Sigma}^{-1}(\mathbf{W}-\mathbf{M})+\mathbf{M}^\top\boldsymbol{\Sigma}^{-1}(\mathbf{W}-\mathbf{M})\right)\right)\right) \\
=& \rho\exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Omega}^{-1}\left((\mathbf{W}-\mathbf{M})^\top\boldsymbol{\Sigma}^{-1}(\mathbf{W}-\mathbf{M})+\mathbf{M}^\top\boldsymbol{\Sigma}^{-1}(\mathbf{W}-\mathbf{M})+\mathbf{T}^\top\mathbf{T}-\mathbf{T}^\top\boldsymbol{\Phi}\mathbf{W}\right)\right)\right) \\
=& \rho\exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Omega}^{-1}\left((\mathbf{W}-\mathbf{M})^\top\boldsymbol{\Sigma}^{-1}(\mathbf{W}-\mathbf{M})+\mathbf{T}^\top\boldsymbol{\Phi}(\mathbf{W}-\mathbf{M})+\mathbf{T}^\top\mathbf{T}-\mathbf{T}^\top\boldsymbol{\Phi}\mathbf{W}\right)\right)\right), \\
& \text{since } \boldsymbol{\Sigma}=\boldsymbol{\Sigma}^\top \\
=& \rho\exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Omega}^{-1}\left((\mathbf{W}-\mathbf{M})^\top\boldsymbol{\Sigma}^{-1}(\mathbf{W}-\mathbf{M})+\mathbf{T}^\top(\mathbf{T}-\boldsymbol{\Phi}\mathbf{M})\right)\right)\right) \\
=& \rho\exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Omega}^{-1}\left((\mathbf{W}-\mathbf{M})^\top\boldsymbol{\Sigma}^{-1}(\mathbf{W}-\mathbf{M})+\mathbf{T}^\top\left(\mathbf{I}-\boldsymbol{\Phi}\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi}+\mathbf{A}\right)^{-1}\boldsymbol{\Phi}^\top\right)\mathbf{T}\right)\right)\right) \\
=& \rho\exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Omega}^{-1}\left((\mathbf{W}-\mathbf{M})^\top\boldsymbol{\Sigma}^{-1}(\mathbf{W}-\mathbf{M})+\mathbf{T}^\top\left(\mathbf{I}+\boldsymbol{\Phi}\mathbf{A}^{-1}\boldsymbol{\Phi}^\top\right)^{-1}\mathbf{T}\right)\right)\right), \\
& \text{by the Woodbury matrix identity} \\
=& (2\pi)^{-\frac{V(N+1)}{2}}|\boldsymbol{\Omega}|^{-\frac{N+1}{2}}|\boldsymbol{\Sigma}|^{-\frac{V}{2}}\exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Omega}^{-1}(\mathbf{W}-\mathbf{M})^\top\boldsymbol{\Sigma}^{-1}(\mathbf{W}-\mathbf{M})\right)\right) \\
& (2\pi)^{-\frac{VN}{2}}|\boldsymbol{\Omega}|^{-\frac{N}{2}}\left|\mathbf{I}+\boldsymbol{\Phi}\mathbf{A}^{-1}\boldsymbol{\Phi}^\top\right|^{-\frac{V}{2}}\exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Omega}^{-1}\mathbf{T}^\top\left(\mathbf{I}+\boldsymbol{\Phi}\mathbf{A}^{-1}\boldsymbol{\Phi}^\top\right)^{-1}\mathbf{T}\right)\right)
\end{aligned}
$$