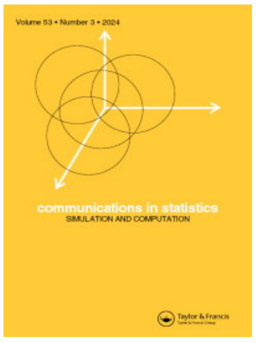

Communications in Statistics - Simulation and Computation

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/lssp20

A Bayesian Lasso based sparse learning model

Ingvild M. Helgøy & Yushu Li

COMMUNICATIONS IN STATISTICS - SIMULATION AND COMPUTATION®
https://doi.org/10.1080/03610918.2023.2272230

OPEN ACCESS

A Bayesian Lasso based sparse learning model

Ingvild M. Helgøy and Yushu Li

Department of Mathematics, University of Bergen, Bergen, Norway

ABSTRACT

The Bayesian Lasso is constructed in the linear regression framework and applies the Gibbs sampling to estimate the regression parameters. This paper develops a new sparse learning model, named the Bayesian Lasso Sparse (BLS) model, that takes the hierarchical model formulation of the Bayesian Lasso. The main difference from the original Bayesian Lasso lies in the estimation procedure; the BLS uses a learning algorithm based on the type-II maximum likelihood procedure. Opposed to the Bayesian Lasso, the BLS provides sparse estimates of the regression parameters. The BLS is also derived for nonlinear supervised learning problems by introducing kernel functions. We compare the BLS model to the well known Relevance Vector Machine, the Fast Laplace, the Bayesian Lasso, and the Lasso, on both simulated and real data. The numerical results show that the BLS is sparse and precise, especially when dealing with noisy and irregular dataset.

ARTICLE HISTORY
Received 9 February 2023
Accepted 12 October 2023

KEYWORDS
Bayesian lasso; Hierarchical models; Kernel functions; Relevance vector machine; Sparse Bayesian learning; Type-II maximum likelihood

1. Introduction

The Lasso of Tibshirani (1996) is a state-of-the-art method for solving linear regression problems. It performs both estimation and variable selection since some of the estimated coefficients can be set to zero during the estimation procedure. It is known that the Lasso estimate corresponds to a Bayesian posterior mode estimate where the priors for the coefficients are identical and independent Laplace distributions (Tibshirani 1996; Hastie et al. 2009). Motivated by the connection between the Lasso and the Laplace prior in the Bayesian framework, Park and Casella (2008) introduced a fully Bayesian model of the Lasso. The Bayesian Lasso by Park and Casella (2008) uses a Laplace prior conditioned on the variance of the random noise, while the approximation of the posterior thereafter is obtained by using the Gibbs sampler. Park and Casella (2008) argued that the prior conditioned on the variance is important because it guarantees a unimodal posterior. On the other hand, using the unconditional Laplace prior from the Lasso, the posterior can easily have more than one mode which may slow down convergence of the Gibbs sampler and produce less concise point estimates. A limitation of the Bayesian Lasso is that it is not a sparse model because none of the estimated coefficient parameters from the Gibbs sampling will be set exactly to zero. Hence, the Bayesian Lasso does not perform variable selection by default. However, Park and Casella (2008) suggest to use the corresponding credible intervals to guide variable selection, but this requires choosing appropriate threshold values. Other similar methods that attempt to overcome the problem of sparsity includes a Bayesian Elastic net method proposed by Li and Lin (2010), and the Bayesian adaptive Lasso by Leng, Tran, and Nott (2014) that uses an adaptive penalty parameter which promotes sparsity. Both these methods utilize the Gibbs sampler to approximate the posterior distribution. Hahn and Carvalho (2015) review the

CONTACT Ingvild M. Helgøy Ingvild.Helgoy@uib.no Department of Mathematics, University of Bergen, Bergen, Norway.

relationship between variable selection priors and shrinkage priors and present an overview of several Bayesian model selection methods.

The last decades, kernel-based machine learning methods have gained increased interest. One of the more popular kernel-based methods is the Support Vector Machine (SVM) (Boser, Guyon, and Vapnik 1992; Vapnik, Golowich, and Smola 1996; Schölkopf et al. 1999), which is a sparse method that has proven to perform well for several different cases, see e.g. Steinwart and Christmann (2008). Despite its popularity, the SVM has also some known limitations. The SVM is purely deterministic because the SVM prediction is a point estimate. Further, Tipping (2001) points out that the SVM model does require that the kernel must satisfy the Mercer conditions (Schölkopf, Smola, and Müller 1998). To overcome this issue, among others, Tipping (2001) presents a new method called the Relevance Vector Machine (RVM), and a faster version was later developed (Tipping and Faul 2003). Opposed to the SVM, the RVM is formulated in the Bayesian framework and the kernel does not have to satisfy the Mercer conditions. Both SVM and RVM can be applied to solve nonlinear regression and classification problems. They can both achieve sparsity in the sample domain and depend only on a subset of the kernel functions and associated training samples. Tipping (2001) demonstrates that the RVM is more sparse than the SVM, and the RVM has shown great success in solving many machine learning and pattern recognition problems (see, e.g. Liu et al. 2015; Kaltwang, Todorovic, and Pantic 2016; Kong et al. 2019; Qiao et al. 2019; Liu et al. 2020). Several extensions of the RVM model can be found in the literature (see, e.g. Schmolck and Everson 2007; Ji, Xue, and Carin 2008; Mohsenzadeh, Sheikhzadeh, and Nazari 2016; Tien Bui et al. 2018; Agrawal, Muchahary, and Tripathi 2019). The RVM has inspired a variety of similar Bayesian methods, including the Fast Laplace (FLAP) by Babacan, Molina, and Katsaggelos (2010), which has mainly been applied in the field of compressive sensing for signal reconstruction. In the derivation of the FLAP, Babacan, Molina, and Katsaggelos (2010) show how the variance can be estimated in the optimization procedure. However, in the numerical results they use a fixed value for the variance because of unstable estimates when the method was used on compressive sensing datasets. In general, sparse Bayesian methods, like the RVM, are frequently used in the field of compressive sensing (see, e.g. Zhou, Liu, and Fang 2015; Huang et al. 2014; Yu et al. 2016; Liu et al. 2018).

This paper presents an adaptation of the Bayesian Lasso by Park and Casella (2008) to the kernel-based framework such that general nonlinear problems can be solved and sparsity in sample domain is achieved. We call this new model the Bayesian Lasso Sparse (BLS) model. Instead of using the Gibbs sampler in Park and Casella (2008) to estimate the parameters, we adapt the marginal likelihood maximization method (also known as type-II maximum likelihood) from the RVM (Tipping and Faul 2003). The BLS assigns individual weight parameters to each input sample and the hierarchical structure from Park and Casella (2008) is utilized in the BLS model which results in an "automatic relevance determination" (ARD) conditional prior (MacKay 1992) for the weight parameters. We will show that this approach leads to a sparse solution in the sample domain. In addition, we provide an analysis of how the variance of the random noise affects the hyperparameter estimation by using the conditional prior for the weight parameters. We conduct a comprehensive simulation study to compare the performance of the BLS with the fast RVM of Tipping and Faul (2003) and the FLAP of Babacan, Molina, and Katsaggelos (2010). We also include the version of the FLAP model that estimate the variance to test its performance on non-linear regression data.

This paper also adjusts the BLS to achieve sparsity in the variable domain in the linear regression framework, where the Bayesian Lasso by Park and Casella (2008) is developed. When applying these modifications to the BLS we obtain a variable selection method that behaves like the original Lasso. The difference between the BLS in the linear regression framework and the Bayesian Lasso by Park and Casella (2008) lies at the estimation procedure. The type-II maximum likelihood estimation results in a sparse model for the BLS where the most important variables have been selected, while the Gibbs sampling in the Bayesian Lasso will in general not produce a sparse model but instead suggest to use credible intervals that can guide variable selection. Simulation studies are used to investigate the sparsity of the BLS in variable domain, and compare it to the Bayesian Lasso and the original Lasso.

The remainder of the paper is divided into the following sections: Sec. 2 contains a detailed description of the BLS, including a fast optimization algorithm and theoretical derivation of the variance related threshold for the hyperparameters. Section 3 presents results from simulation studies and compares the BLS with other kernel-based learning methods where sparsity is achieved in the sample domain. Section 4 shows how the BLS can be used for variable selection, and concluding remarks are given in Sec. 5.

2. The Bayesian Lasso Sparse model

After a general description of supervised learning, this section explains in detail the hierarchical structure of the BLS and its kernel framework. In addition, we specify the type-II maximum likelihood method, as well as an analysis of how the variance of the random noise can affect the sparsity. Inference and prediction by using the BLS model will be briefly described at the end of this section.

2.1. Supervised learning

Supervised learning contains a set of training data $\{\boldsymbol{x}_i, y_i\}_{i=1}^N$, where $\boldsymbol{x}_i \in \mathbb{R}^D$ is a D-dimensional input vector and $y_i \in \mathbb{R}$ is the corresponding scalar target value. Based on the training data, we aim at estimating a function $f(\boldsymbol{x})$ that can model the underlying relationship between the input covariate $\boldsymbol{x}_i$ and the target observation y_i. A common assumption is that $f(\boldsymbol{x})$ can be represented by a set of M linearly independent basis vectors, ϕ_m :

$$f(\boldsymbol{x}) = w_0 + \sum_{m=1}^{M} w_m \phi_m(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{w} = (w_0, w_1, ..., w_M)^\top$ is a vector of weight parameters.

We assume that the observed targets, y_i, are samples of the function $f(\boldsymbol{x})$ with added noise which follows a Gaussian distribution. To relate the targets to the input, we first create a design matrix, Φ, from the basis vectors as

$$\Phi = [\boldsymbol{1}, \phi_1, ..., \phi_M], \qquad \phi_m = (\phi_m(\boldsymbol{x}_1), ..., \phi_m(\boldsymbol{x}_N))^\top, \quad m = 1, ..., M.$$

Let $\boldsymbol{\epsilon} = (\epsilon_1, ..., \epsilon_N)^\top$ be the standard Gaussian distributed noise vector, we then have:

$$\boldsymbol{y} = \Phi \boldsymbol{w} + \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_N), \tag{2}$$

where $\boldsymbol{y} = (y_1, ..., y_N)^\top, \sigma^2$ is the variance of the error terms that are normally distributed and $\boldsymbol{I}_N$ is the $N \times N$ identity matrix.

2.2. Model specification and parameter estimation

The first step to construct the BLS model is to define the basis functions in Equation (1). In the BLS, the basis functions are defined by $M = N$ kernel functions $\phi_m(\boldsymbol{x}) = K(\boldsymbol{x}, \boldsymbol{x}_m);\ m = 1, ..., N$. The kernel function $K(\cdot, \cdot)$ is centered at each of the training input vectors. Thus, each basis function $\phi_m(\boldsymbol{x})$ corresponds to one training input vector $\boldsymbol{x}_m$. The Gaussian and the polynomial kernel functions are the most used kernel functions for sparse learning models.

After the type of kernel function is chosen, the learning task is to estimate the weight parameters, $\boldsymbol{w}$, from the training data. Similar to the Bayesian Lasso, the conditional prior on the weight parameters is given by

$$p(\boldsymbol{w}|\sigma^2,\lambda) = \prod_{i=0}^{N} \frac{\sqrt{\lambda}}{2\sqrt{\sigma^2}} e^{-\sqrt{\lambda}|w_i|/\sqrt{\sigma^2}}, \qquad \lambda \geq 0, \tag{3}$$

which can be recognized as a Laplace prior conditioned on σ^2 and λ. Further, the likelihood function is derived from Equation (2) as

$$p(\boldsymbol{y}|\boldsymbol{w},\sigma^2) = \mathcal{N}(\boldsymbol{y}|\Phi\boldsymbol{w},\sigma^2). \tag{4}$$

Bayesian inference is based on the posterior distribution of $\boldsymbol{w}$ given the data, $p(\boldsymbol{w}|\boldsymbol{y},\sigma^2,\lambda)$, which can be calculated from Equations (3, 4). However, the inclusion of the Laplace prior in Equation (3) makes an analytical solution intractable. We therefore proceed to use a hierarchical representation of the full model, similar to the Bayesian Lasso described by Park and Casella (2008):

$$\text{Likelihood} \quad p(\boldsymbol{y}|\boldsymbol{w},\sigma^2) = \mathcal{N}(\boldsymbol{y}|\Phi\boldsymbol{w},\sigma^2), \tag{5}$$

$$\text{Hierarchicalprior} \quad p(\boldsymbol{w}|\boldsymbol{\tau},\sigma^2) = \prod_{i=0}^{N} \mathcal{N}(w_i|0,\tau_i\sigma^2), \qquad \tau_i \geq 0, \tag{6}$$

$$\text{Hyperprior} \quad p(\boldsymbol{\tau}|\lambda) = \prod_{i=0}^{N} \frac{\lambda}{2} \mathrm{e}^{-\frac{\lambda\tau_i}{2}}, \qquad \lambda \geq 0. \tag{7}$$

The hierarchical prior in Equation (6) can be viewed as an ARD prior where for $i = 0, 1, \ldots, N$, there is an individual hyperparameter τ_i associated independently with each individual weight w_i. Equation (6) shows that when a hyperparameter is estimated to be zero in this ARD prior, it will force the corresponding weight parameter to be zero, and prune the input vector and the related sample.

The priors for λ and σ^2 in Equations (5–7), are defined as the Gamma and inverse Gamma distribution, respectively:

$$p(\lambda) = \frac{b^a}{\Gamma(a)} (\lambda)^{a-1} e^{-b\lambda}, \qquad a, b \geq 0, \tag{8}$$

$$p(\sigma^2) = \frac{d^c}{\Gamma(c)} (\sigma^2)^{-c-1} e^{-d/\sigma^2}, \qquad c, d \geq 0. \tag{9}$$

By combining Equations (5–9) from the above hierarchical Bayesian model, we get the following joint distribution of the dataset, parameters and hyperparameters:

$$p(\boldsymbol{y},\boldsymbol{w},\boldsymbol{\tau},\sigma^2,\lambda) = p(\boldsymbol{y}|\boldsymbol{w},\sigma^2)p(\boldsymbol{w}|\boldsymbol{\tau},\sigma^2)p(\boldsymbol{\tau}|\lambda)p(\lambda)p(\sigma^2).$$

Given the observed data, the following posterior over all unknowns can be found:

$$p(\boldsymbol{w},\boldsymbol{\tau},\sigma^2,\lambda|\boldsymbol{y}) = \frac{p(\boldsymbol{y},\boldsymbol{w},\boldsymbol{\tau},\sigma^2,\lambda)}{p(\boldsymbol{y})}. \tag{10}$$

When the posterior distribution in Equation (10) is available, an expression of the predictive distribution for the output y^* can be obtained as

$$p(y^*|\boldsymbol{y}) = \int p(y^*|\boldsymbol{w}, \sigma^2)p(\boldsymbol{w}, \boldsymbol{\tau}, \sigma^2, \lambda|\boldsymbol{y})\mathrm{d}\boldsymbol{w}\mathrm{d}\boldsymbol{\tau}\mathrm{d}\sigma^2\mathrm{d}\lambda. \tag{11}$$

However, neither the posterior in Equation (10) nor the predictive distribution in Equation (11) can be computed analytically as the normalizing integral $p(\boldsymbol{y}) = \int p(\boldsymbol{y}, \boldsymbol{w}, \boldsymbol{\tau}, \sigma^2, \lambda)\mathrm{d}\boldsymbol{w}\mathrm{d}\boldsymbol{\tau}\mathrm{d}\sigma^2\mathrm{d}\lambda$ is intractable. Instead, the BLS model applies the type-II maximum likelihood estimation from Tipping and Faul (2003) to approximate the predictive distribution in Equation (11).

As the posterior in Equation (10) can not be found directly from Bayes' rule, we use the decomposition:

$$p(\boldsymbol{w}, \boldsymbol{\tau}, \sigma^2, \lambda|\boldsymbol{y}) = p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{\tau}, \sigma^2, \lambda)p(\boldsymbol{\tau}, \sigma^2, \lambda|\boldsymbol{y}). \tag{12}$$

The distribution $p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{\tau}, \sigma^2, \lambda)$ on the right hand side of Equation (12), is a Gaussian distribution with the following mean vector and covariance matrix:

$$\boldsymbol{\mu} = \sigma^{-2}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\mathbf{y},$$
$$\boldsymbol{\Sigma} = [\sigma^{-2}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \Lambda^{-1}]^{-1},$$

where $\Lambda = \mathrm{diag}(\tau_i\sigma^2)$.

To estimate $\boldsymbol{\tau}$, we search for the local maximum of $p(\boldsymbol{\tau}, \sigma^2, \lambda|\boldsymbol{y})$ in Equation (12) with respect to the individual hyperparameters τ_i by using the type-II maximum likelihood procedure. To find the maximum, we use $p(\boldsymbol{\tau}, \sigma^2, \lambda|\boldsymbol{y}) = p(\boldsymbol{y}, \boldsymbol{\tau}, \sigma^2, \lambda)/p(\boldsymbol{y}) \propto p(\boldsymbol{y}, \boldsymbol{\tau}, \sigma^2, \lambda)$, and maximize the following joint distribution $p(\boldsymbol{y}, \boldsymbol{\tau}, \sigma^2, \lambda)$ to get the type-II maximum likelihood estimation of $\boldsymbol{\tau}$. This joint distribution can be obtained by integrating out $\boldsymbol{w}$ as

$$\begin{aligned} p(\boldsymbol{y}, \boldsymbol{\tau}, \sigma^2, \lambda) &= \int p(\boldsymbol{y}|\boldsymbol{w}, \sigma^2)p(\boldsymbol{w}|\boldsymbol{\tau}, \sigma^2)p(\boldsymbol{\tau}|\lambda)p(\lambda)p(\sigma^2)\mathrm{d}\boldsymbol{w} \\ &= \left(\frac{1}{2\pi}\right)^{N/2}|\boldsymbol{C}|^{-\frac{1}{2}}e^{-\frac{1}{2}\boldsymbol{y}^T\boldsymbol{C}^{-1}\boldsymbol{y}}p(\boldsymbol{\tau}|\lambda)p(\lambda)p(\sigma^2), \end{aligned}$$

where $\boldsymbol{C} = (\sigma^2\boldsymbol{I}_N + \boldsymbol{\Phi}\Lambda\boldsymbol{\Phi}^\top)$. The log of $p(\boldsymbol{y}, \boldsymbol{\tau}, \sigma^2, \lambda)$ is given by

$$\begin{aligned} L = &-\frac{1}{2}\log|\boldsymbol{C}| - \frac{1}{2}\boldsymbol{y}^T\boldsymbol{C}^{-1}\boldsymbol{y} + N\log\frac{\lambda}{2} - \frac{\lambda}{2}\sum_i\tau_i + a\log b - \log\Gamma(a) \\ &+(a-1)\log\lambda - b\lambda + c\log d - \log\Gamma(c) - (c+1)\log\sigma^2 - \frac{d}{\sigma^2}. \end{aligned} \tag{13}$$

In the following subsection, an optimization algorithm for Equation (13) is presented, and we prove that the algorithm gives a sparse model because some of the τ_i will be set to zero. Thus, the corresponding weights and input vectors are pruned from the model according to the construction of the ARD prior in Equation (6).

2.3. Fast optimization algorithm

A disadvantage of the original RVM described by Tipping (2001) is that it is computationally slow in the maximization of the type-II likelihood. The original RVM begins with all N basis functions included in the model and updates the hyperparameters iteratively. During the updates, some of the basis functions are pruned. However, the first few iterations require $O(N^3)$ computations. Tipping and Faul (2003) overcome this problem by introducing a fast marginal likelihood maximization algorithm for sparse Bayesian models. Instead of updating the whole hyperparameter vector $\boldsymbol{\tau}$, only a single parameter τ_i is updated at each iteration (step 4 of Algorithm 1). We follow the procedure explained by Tipping and Faul (2003) and choose the τ_i that gives the largest increase of the log marginal likelihood in Equation (13). This fast maximization process is utilized in many sparse learning studies including the work by Babacan, Molina, and Katsaggelos (2010).

To find the new value of τ_i, we first rewrite Equation (13) as:

$$\begin{aligned} L = & -\frac{1}{2}\left[\log|\boldsymbol{C}_{-i}| + \boldsymbol{y}^T\boldsymbol{C}_{-i}^{-1}\boldsymbol{y} + \lambda\sum_{j\neq i}\tau_j\right] + \frac{1}{2}\left[\log\frac{1}{1+\sigma^2\tau_i s_i} + \frac{q_i^2\sigma^2\tau_i}{1+\sigma^2\tau_i s_i} - \lambda\tau_i\right] \\ & + N\log\frac{\lambda}{2} + a\log b - \log\Gamma(a) + (a-1)\log\lambda - b\lambda + c\log d - \log\Gamma(c) \\ & -(c+1)\log\sigma^2 - \frac{d}{\sigma^2}, \end{aligned} \tag{14}$$

where

$$s_i = \phi_i^T\boldsymbol{C}_{-i}^{-1}\phi_i, \quad \text{and} \quad q_i = \phi_i^T\boldsymbol{C}_{-i}^{-1}\boldsymbol{y}. \tag{15}$$

The notation $\boldsymbol{C}_{-i}$ denotes the covariance matrix $\boldsymbol{C}$ without the inclusion of the ith basis function, ϕ_i. The decomposition of the covariance matrix $\boldsymbol{C}$ into $\boldsymbol{C}_{-i}$ and the other components is explained in Appendix A. The log-likelihood function, L, is now decomposed into three parts; the first part is the likelihood where τ_i and the corresponding ϕ_i are excluded, the second part contains the terms that involve τ_i and ϕ_i, while the last part contains all terms not containing $\boldsymbol{\tau}$. From this decomposition, the maximum of L with respect to a single hyperparameter τ_i is found by taking the partial derivative with respect to τ_i and setting it equal to zero, which gives

$$\tau_i = \begin{cases} \dfrac{-s_i - 2\lambda\sigma^{-2} + \sqrt{s_i^2 + 4q\lambda\sigma^{-2}}}{2\lambda s_i}, & q_i^2 - s_i > \lambda\sigma^{-2}, \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

The derivation of Equation (16) can be found in Appendix B.

After τ_i is updated we proceed to find the λ in Equation (16) that maximizes the log likelihood. By taking the derivative of Equation (13) with respect to λ and setting it to zero we obtain

$$\lambda = \frac{2(N+a-1)}{\sum_i\tau_i + 2b}. \tag{17}$$

Similarly, a new estimate of σ^2 is found. When we take the derivative of Equation (13) with respect to σ^2, we notice that σ^2 can be separated from the rest of the components in $\boldsymbol{C}$ such that $\boldsymbol{C} = \sigma^2\tilde{\boldsymbol{C}}$, where $\tilde{\boldsymbol{C}}$ is independent of σ^2. The updated value of σ^2 is then found as

$$\sigma^2 = \frac{\boldsymbol{y}^\top\tilde{\boldsymbol{C}}^{-1}\boldsymbol{y} + 2d}{N + 2c + 2}. \tag{18}$$

In the optimization algorithm, we also have to update the expressions for s_i and q_i. Computing the values of s_i and q_i directly from Equation (15) requires the inversion of the matrix $\boldsymbol{C}_{-i}$. Instead, we follow the approach of Tipping and Faul (2003) and calculate:

$$s_i = \frac{S_i}{1-\tau_i\sigma^2 S_i}, \qquad q_i = \frac{Q_i}{1-\tau_i\sigma^2 S_i}, \tag{19}$$

where,

$$S_i = \sigma^{-2}\phi_i^\top\phi_i - \sigma^{-2}\phi_i^\top\boldsymbol{\Phi\Sigma\Phi}^\top\phi_i\sigma^{-2}, \tag{20}$$

$$Q_i = \sigma^{-2}\phi_i^\top\boldsymbol{y} - \sigma^{-2}\phi_i^\top\boldsymbol{\Phi\Sigma\Phi}^\top\boldsymbol{y}\sigma^{-2}. \tag{21}$$

The matrices $\mathbf{\Sigma}$ and $\mathbf{\Phi}$ contain only the basis functions that are currently included in the model. This computation is therefore much faster than if we had started with all N basis functions. Algorithm 1 summarizes the procedure.

Algorithm 1. The Bayesian Lasso Sparse (BLS) Learning Model

```
1: initialize $\sigma^2$ to some sensible value (e.g. var($\boldsymbol{y}$) $\times$ 0.1)
2: Initialize all $\tau_i = 0$, $\lambda = 0$
3: **while** convergence criteria are not met, **do**
4:    Choose a $\tau_i$
5:    **if** $q_i^2 - s_i > \lambda\sigma^{-2}$ and $\tau_i = 0$ **then**
6:       Add $\tau_i$ to the model with updated $\tau_i$
7:    **else if** $q_i^2 - s_i > \lambda\sigma^{-2}$ and $\tau_i > 0$, **then**
8:       Re-estimate $\tau_i$
9:    **else if** $q_i^2 - s_i < \lambda\sigma^{-2}$, **then**
10:      Set $\tau_i = 0$
11:   **end if**
12:   Update $\Sigma$ and $\boldsymbol{\mu}$
13:   Update $\lambda$ using Equation (17)
14:   Update $\sigma^2$ using Equation (18)
15:   Update $s_i$ and $q_i$ using Equations (19–21)
16: **end while**
```

From Equation (16) and Algorithm 1, we see that the criteria for setting $\tau_i = 0$ depends on both λ and the variance term σ^2. We will now prove that when $\sigma^2 \to \infty$ then the criteria $q_i^2 - s_i \le \lambda\sigma^{-2}$ will be satisfied. We can write $\boldsymbol{C}_{-i}$ as (see Appendix A):

$$\begin{aligned}\boldsymbol{C}_{-i} &= \sigma^2\boldsymbol{I} + \mathbf{\Phi}_{-i}\Lambda_{-i}\mathbf{\Phi}_{-i}^{\top}\\ &= \sigma^2\widetilde{\boldsymbol{C}}_{-i},\end{aligned}$$

where $\mathbf{\Phi}_{-i}$ is the $N \times N$ design matrix where basis function i is removed, Λ_{-i} is the diagonal matrix Λ where the single element τ_i is removed, and $\widetilde{\boldsymbol{C}}_{-i}$ denotes C_{-i} where the component σ^2 is excluded. Inserting this decomposition into Equation (15), s_i and q_i can be written as

$$s_i = \sigma^{-2}\widetilde{s}_i, \qquad q_i = \sigma^{-2}\widetilde{q}_i, \tag{22}$$

where $\widetilde{s}_i = \phi_i^{\top}\widetilde{\boldsymbol{C}}_{-i}^{-1}\phi_i$ and $\widetilde{q}_i = \phi_i^{\top}\widetilde{\boldsymbol{C}}_{-i}\boldsymbol{y}$. The condition in Equation (16) says that τ_i is set to zero when $q_i^2 - s_i \le \lambda\sigma^{-2}$. Inserting Equation (22) into this inequality gives

$$\sigma^{-2}\widetilde{q}_i^{\,2} - \widetilde{s}_i \le \lambda.$$

We see that as $\sigma^2 \to \infty$, the inequality always holds because $\lambda \ge 0$ and $\tilde{s}_i \ge 0$. Thus, the corresponding τ_i is set to zero. Therefore, in the BLS, the information of σ^2 is utilized to adjust the number of zero hyperparameters during the estimation of $\boldsymbol{\tau}$. This feature makes the BLS more robust to noisy information that might be confused with the real signal information.

2.4. Prediction

After the convergence of the learning Algorithm 1, we end up with a set of nonzero τ_i's and each of them corresponds to a "relevance basis function" and a related "relevance input vector" from the training data. For a new input data, $\boldsymbol{x}^*$, we can make predictions based on the posterior of the weights conditioned on $\boldsymbol{\tau}$ and σ^2. The predictive distribution (11) for the output y^* can be approximated by

$$p(y^*|\boldsymbol{y}, \boldsymbol{\tau}, \sigma^2) = \int p(y^*|\boldsymbol{w}, \boldsymbol{\tau}, \sigma^2) p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{\tau}, \sigma^2) \mathrm{d}\boldsymbol{w}.$$

This distribution is Gaussian with the following predictive mean and predictive variance:

$$y^* = \phi(\boldsymbol{x}^*)^\top \boldsymbol{\mu},$$
$$\sigma^{*2} = \sigma^2 + \phi(\boldsymbol{x}^*)^\top \Sigma \phi(\boldsymbol{x}^*).$$

where $\phi(\boldsymbol{x}^*) = (1, \phi_1(\boldsymbol{x}^*), \ldots, \phi_N(\boldsymbol{x}^*))^\top$. Note that only the basis functions corresponding to the nonzero τ_i's contribute to the posterior mean vector and covariance matrix.

In practice, the predictive mean can be used as a point prediction, and the predictive variance can be used to construct the prediction interval.

3. BLS for nonlinear regression

In this section, we compare the BLS with the RVM (Tipping and Faul 2003) and the FLAP (Babacan, Molina, and Katsaggelos 2010). We use simulated datasets to be able to compare the estimated σ^2 to the true value. For all the methods, we use the Gaussian kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-r^2\|\boldsymbol{x}_i - \boldsymbol{x}_j\|)$, where the kernel parameter r is determined by using five-fold cross validation.

For the Gamma priors in Equations (8, 9) a common practice is to set them to zero in order to obtain uniform hyperpriors for λ and σ^2. In our implementation of the BLS model we therefore set $a = b = c = d = 0$. The RVM uses the same procedure to obtain a uniform hyperprior for σ^2, but the model does not include λ. In the numerical results of Babacan, Molina, and Katsaggelos (2010), they use a uniform hyperprior for λ in the FLAP. However, they use a fixed value for σ^2. This is done due to the under-determined nature of the compressive sensing problem that they apply the method to, which makes the estimates of σ^2 unstable in the early iterations. In the results of this paper for the FLAP, we follow the same setup as in Babacan, Molina, and Katsaggelos (2010) and use $\sigma^{-2} = 0.01\|\boldsymbol{y}\|_2^2$. In addition, we include a version of the FLAP where σ^2 is estimated in the same manner as the BLS and the RVM. In order to distinguish this approach from the original FLAP, we denote it as FLAP$_\sigma$ in the sections below. We include the results of both FLAP and FLAP$_\sigma$ because, to our awareness this is the first time the FLAP is tested on general nonlinear regression problems.

3.1. The Sinc function

We first consider the Sinc function, $f(x) = \sin(x)/x$, a benchmark function that is frequently used to evaluate how kernel-based learning methods perform (Vapnik, Golowich, and Smola 1996; Tipping 2001; Schmolck and Everson 2007). We use the same procedure as Tipping (2001) where the model is built based on 100 training data while the error is calculated with respect to the true function by using 1000 test data.

Figure 1 shows the results of the four methods on datasets with standard deviation $\sigma = 0.05$, 0.3 and 0.7. The black dots represent the training data from the model $y = f(x) + \epsilon$ where $x \in [-10, 10]$. The same data set is used for all methods to better compare them. The location of the nonzero weighted input vectors, often referred to as the relevance vectors, are represented by the red circles. The blue lines correspond to the Sinc function, $f(x)$, while the green lines are the prediction of each method applied to the test data. From Figure 1 we observe that when $\sigma = 0.05$, the approximations of all four methods perform well and almost overlap with the true function. When σ is set to 0.3, the methods can still capture the general form of the original function, except at the boundaries where they produce a more rough approximation. However, when σ is set to 0.7, the training data begin to loose the original shape of the Sinc function, and the approximations of all four methods have issues capturing the Sinc function shape.

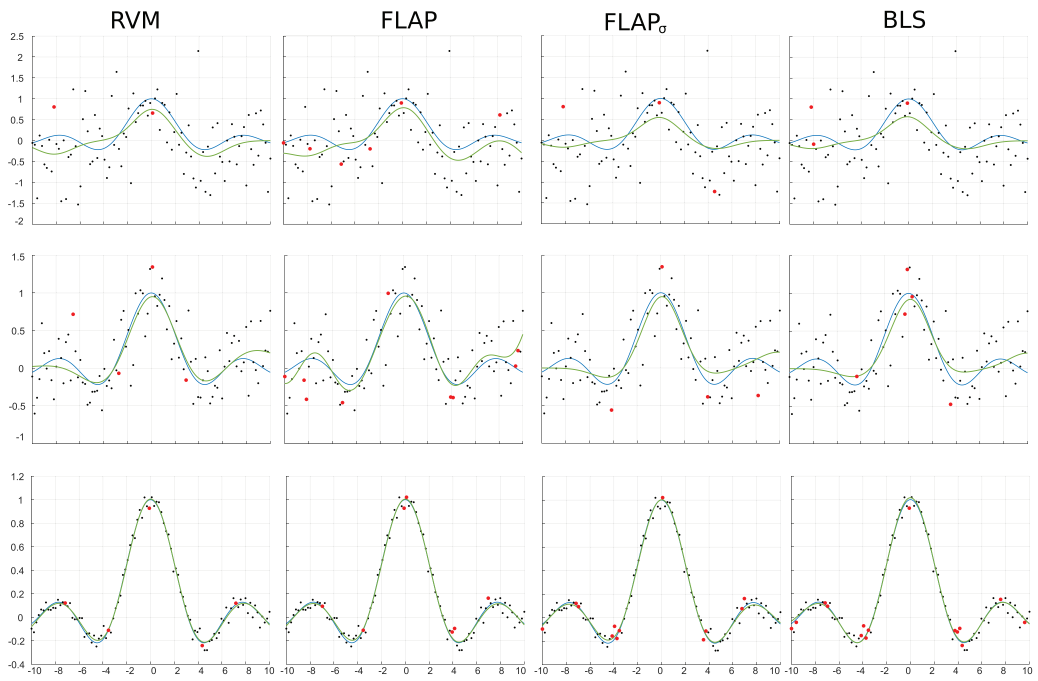

Figure 1. The Sinc function (blue line) and its prediction (green line) from the data generated with $\sigma = 0.05, 0.3$ and 0.7 from bottom to top row. The red dots are the relevance vectors and the black dots are the remaining training data.

Table 1. Results of the simulation study for the Sinc function.

True value of σ	Method	NOV (SE)	RMSE (SE)	$\hat{\sigma}$ (SE)
0.05	RVM	5.31 (0.01)	0.016 (0.0001)	0.049 (0.0001)
	FLAP	6.77 (0.04)	0.015 (0.0001)	–
	FLAP$_\sigma$	12.30 (0.06)	0.017 (0.0001)	0.049 (0.0001)
	BLS	14.01 (0.06)	0.018 (0.0001)	0.049 (0.0001)
0.1	RVM	5.18 (0.02)	0.033 (0.0002)	0.099 (0.0002)
	FLAP	7.34 (0.06)	0.030 (0.0002)	–
	FLAP$_\sigma$	12.79 (0.06)	0.041 (0.0002)	0.101 (0.0002)
	BLS	13.55 (0.06)	0.041 (0.0002)	0.101 (0.0002)
0.3	RVM	4.33 (0.03)	0.091 (0.0005)	0.297 (0.0007)
	FLAP	9.26 (0.08)	0.091 (0.0001)	–
	FLAP$_\sigma$	5.19 (0.05)	0.096 (0.0005)	0.299 (0.0007)
	BLS	5.49 (0.05)	0.098 (0.0005)	0.299 (0.0007)
0.5	RVM	3.80 (0.04)	0.138 (0.0009)	0.489 (0.0011)
	FLAP	9.53 (0.09)	0.154 (0.0012)	–
	FLAP$_\sigma$	3.52 (0.04)	0.132 (0.0008)	0.413 (0.0011)
	BLS	3.70 (0.03)	0.133 (0.0008)	0.491 (0.0011)
0.7	RVM	3.77 (0.04)	0.181 (0.0014)	0.679 (0.0016)
	FLAP	9.66 (0.11)	0.218 (0.0017)	–
	FLAP$_\sigma$	3.47 (0.04)	0.136 (0.0009)	0.516 (0.0024)
	BLS	3.08 (0.02)	0.131 (0.0014)	0.687 (0.0016)

For each value of σ, 1000 datasets were generated. The average number of relevance vectors, NOV, the average root mean square error, RMSE, and the average of estimated noise standard deviation, $\hat{\sigma}$, are reported for each method along with the corresponding standard errors, SE, in the parenthesis.

To measure the prediction accuracy of the methods, we generate 1000 datasets for $\sigma = 0.05$, 0.1, 0.3, 0.5 and 0.7 and run each method on these datasets totaling $1000 \times 5 \times 4 = 20000$ runs. For each value of σ, method and dataset, we calculate the root mean squared error, number of relevance vectors and the estimated standard deviation. Table 1 shows the average values of these quantities, denoted by RMSE, NOV and $\hat{\sigma}$. When the standard deviation is small ($\sigma \leq 0.3$), it is the FLAP that achieves the overall best prediction, but all four methods give good approximations of the true function with low RMSE. For higher standard deviations the RMSE is higher but comparable for all methods. We observe that the NOV of all methods except FLAP is decreasing as the standard deviation increases. Thus, the methods that estimate the standard deviation pick fewer relevance vectors when the noise is large. We also notice that the corresponding standard errors of the NOV for the BLS and FLAP$_\sigma$ show the same decreasing trend when the variance increases. The RVM and the BLS give the most accurate estimates of the standard deviation. In all, Table 1 shows that the BLS performs a bit worse than the rest of the methods when the standard deviation is low, while it performs as well as the established methods when the standard deviation is large.

3.2. The Bump dataset

In this section we investigate how the methods perform on a dataset with high frequency and spikes. The so called Bump function from Donoho and Johnstone (1994) is used to simulate data with sample size $N = 120$ and different signal to noise ratios (SNR).

Figure 2 shows the predictions of the four methods when SNR $= 1, 3$ and 10. In this figure, the true Bump function is represented by the blue lines and the reconstruction from the noisy data is the green lines. The red dots represent the relevance vectors and the black dots represent the remaining data samples. The same data is used for all methods to compare them. When the data is less noisy, with SNR $= 10$, all four methods can capture the bump signal well, however, when the noise is larger, the FLAP tends to overfit the data compared to the other methods. When SNR $= 1$, it is the BLS and FLAP$_\sigma$ that capture the signal best.

As for the Sinc function, we generate 1000 random datasets for SNR = 1, 2, 3, 4, 5 and 10, and utilize RMSE, NOV and $\hat{\sigma}$ to evaluate the four methods. The results are presented in Table 2.

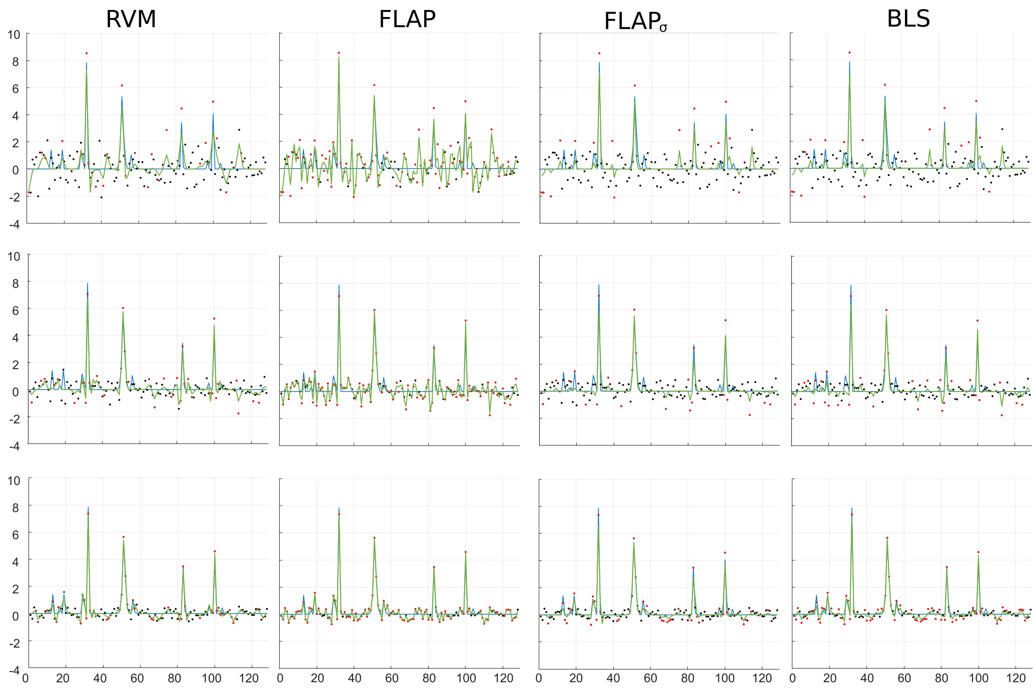

Figure 2. The Bump function (blue line) and its reconstruction (green line) from the data that are generated for different values of SNR = 1, 3 and 10 (from top to bottom row). The red dots are the relevance vectors and the black dots are the remaining data.

Table 2. Results of the simulation study for the Bump function.

SNR	Method	NOV (SE)	RMSE (SE)	$\hat{\sigma}$ (SE)
10	RVM	51.3 (0.20)	0.249 (0.001)	0.251 (0.001)
	FLAP	91.0 (0.19)	0.279 (0.001)	–
	FLAP$_\sigma$	45.9 (0.51)	0.258 (0.001)	0.266 (0.003)
	BLS	62.8 (0.15)	0.217 (0.001)	0.192 (0.001)
5	RVM	49.5 (0.21)	0.342 (0.002)	0.341 (0.002)
	FLAP	85.2 (0.17)	0.415 (0.001)	–
	FLAP$_\sigma$	37.4 (0.64)	0.327 (0.001)	0.397 (0.004)
	BLS	52.0 (0.33)	0.288 (0.001)	0.301 (0.002)
4	RVM	42.8 (0.18)	0.376 (0.001)	0.423 (0.001)
	FLAP	91.5 (0.17)	0.464 (0.001)	–
	FLAP$_\sigma$	36.8 (0.77)	0.349 (0.001)	0.437 (0.004)
	BLS	37.4 (0.31)	0.319 (0.001)	0.414 (0.002)
3	RVM	42.0 (0.18)	0.427 (0.001)	0.481 (0.002)
	FLAP	90.9 (0.18)	0.538 (0.001)	–
	FLAP$_\sigma$	33.2 (0.83)	0.379 (0.002)	0.512 (0.005)
	BLS	35.1 (0.49)	0.346 (0.001)	0.375 (0.003)
2	RVM	40.8 (0.18)	0.511 (0.001)	0.578 (0.002)
	FLAP	89.6 (0.17)	0.662 (0.001)	–
	FLAP$_\sigma$	26.3 (0.72)	0.416 (0.002)	0.653 (0.005)
	BLS	34.1 (0.44)	0.377 (0.001)	0.564 (0.004)
1	RVM	33.8 (0.17)	0.694 (0.002)	0.866 (0.003)
	FLAP	84.7 (0.19)	0.929 (0.019)	–
	FLAP$_\sigma$	24.3 (1.02)	0.518 (0.005)	0.900 (0.009)
	BLS	15.9 (0.15)	0.472 (0.002)	0.966 (0.003)

For each value of SNR, 1000 datasets were generated. The average number of relevance vectors, NOV, the average root mean square error, RMSE, and the average of estimated standard deviation, $\hat{\sigma}$, are reported for each method along with the corresponding standard errors, SE, in the parenthesis.

We observe that the predictions of the BLS have the lowest RMSE of all four methods, especially when the dataset is noisy (SNR ≤ 2). The predictions of the FLAP$_\sigma$ have a lower RMSE than RVM and FLAP, except for SNR $= 10$. Table 2 also shows that the NOV for the FLAP is roughly twice that of the other methods. This is consistent with the indication from Figure 2 that FLAP might overfit the data. The NOV values for the other models are more similar, but we notice that the BLS is the most sparse model when the SNR is low and the dataset is noisy. The NOV of the methods that estimate σ^2 decreases as the noise of the dataset increases, similar to the results in Sec. 3.1.

Compared to the smooth Sinc function where all four methods gave similar results, the BLS shows better result, both in terms of RMSE and NOV, than the other three methods for the irregular and noisy Bump dataset.

4. BLS for variable selection

Sections 2 and 3 derive and test the kernel-based framework where general nonlinear regression problems can be solved. This section describes how the BLS can perform variable selection in multiple linear regression models. Consider the linear regression problem

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}_N),$$

where $\boldsymbol{\beta} = (\beta_1, ..., \beta_P)^\top$ is a vector that holds regression coefficients, $\boldsymbol{y}$ is the response vector, and $\boldsymbol{X}$ is the $N \times P$ matrix of standardized variables. The learning algorithm of the BLS is generally the same as in Sec. 2, and the main difference is that all the Φ matrices will be $\boldsymbol{X}$ matrices and no kernel function is needed.

One of the main features of the Lasso by Tibshirani (1996) is its ability to do variable selection. On the other hand, the Bayesian Lasso by Park and Casella (2008) does not perform variable selection because the regression coefficients in $\boldsymbol{\beta}$ are estimated by the Gibbs sampling. This motivates the use of the BLS where variables related to pruned coefficients will be deleted and sparsity is achieved in variable domain.

For the Gamma priors in Equations (8, 9), we set $a = b = c = d = 0$ for all the results in this section, similar as done in Sec. 3.

4.1. Simulated data

This sub-section compares the BLS for variable selection to the Bayesian Lasso (Park and Casella 2008) and the Lasso (Tibshirani 1996) using simulated data. The data are simulated from

$$y_i = \boldsymbol{x}_i^\top \boldsymbol{\beta} + \epsilon_i, \qquad i = 1, \ldots, N,$$

where $\epsilon_i \sim N(0, \sigma^2)$ and $\boldsymbol{x}_i \in \mathbb{R}^D$. We consider three different simulation studies where we run 1000 simulations for each study. In Simulation 1, the number of observations, N, is much larger than the number of variables, D. In Simulation 2, the number of observations is slightly higher than the number of variables, while in Simulation 3, the number of observations is less than the number of variables.

For all studies we generate a training set of size 50, while the test error is calculated with respect to the true model by using a test set of size 100. For the Bayesian Lasso, the posterior means are calculated based on 15 000 samples after discarding 3000 burn-in samples. Further, the hyperparameters a and b in the Gamma prior for λ^2 is set to 0.1. For the Lasso, the tuning parameter λ is selected by using 10-fold cross validation.

In the first simulation study we set $\boldsymbol{\beta} = (3, 1.5, 0, 0, 2, 0, 0, 0)^\top$ and consider three different scenarios setting $\sigma = 1, 3$ and 5. The pairwise correlation between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j, i \neq j$, is $\rho^{|i-j|}$ with $\rho = 0.5$. A similar example was used in the original paper by Tibshirani (1996) and in several later studies (Leng, Lin, and Wahba 2006; Li and Lin 2010; Alhamzawi and Ali 2018).

The simulation results are reported in Table 3, where the selected frequency of the coefficients from the 1000 simulations are listed. As the true values of β_1, β_2 and β_5 are nonzero, a correct model should select these three as the nonzero coefficients. The table shows that both the Lasso and the BLS selects these coefficients in all 1000 repetitions when $\sigma = 1$, while for $\sigma = 3$ and 5, the Lasso has the highest frequency. The Bayesian Lasso selects all coefficients in every repetition, which is as expected. Maybe more interesting, is that the BLS selects the remaining coefficients, whose true values are zero, less frequently than the Lasso.

Table 3 also reports the average number of nonzero coefficients (NOC) along with the average RMSE and corresponding standard errors. The NOC values from the Bayesian Lasso are 8 in all three cases, which means that none of the estimated coefficients are zero, and the NOC values from the BLS are closer to 3 than the NOC values of the Lasso. The BLS has the lowest RMSE

Table 3. Results from simulation 1.

σ	Method	β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8	NOC (SE)	RMSE (SE)
1	BLS	1	1	0.32	0.28	1	0.30	0.28	0.29	4.46 (0.03)	0.296 (0.004)
	BL	1	1	1	1	1	1	1	1	8 (0)	0.411 (0.004)
	Lasso	1	1	0.47	0.46	1	0.47	0.41	0.47	5.27 (0.05)	0.405 (0.004)
3	BLS	1	0.97	0.26	0.29	0.99	0.25	0.22	0.23	4.20 (0.01)	0.975 (0.013)
	BL	1	1	1	1	1	1	1	1	8 (0)	1.209 (0.012)
	Lasso	1	0.99	0.47	0.47	0.99	0.43	0.42	0.46	5.24 (0.05)	1.182 (0.012)
5	BLS	0.99	0.77	0.23	0.21	0.84	0.19	0.14	0.14	3.51 (0.03)	1.802 (0.023)
	BL	1	1	1	1	1	1	1	1	8 (0)	1.893 (0.021)
	Lasso	0.99	0.87	0.42	0.47	0.93	0.43	0.41	0.42	4.94 (0.05)	1.952 (0.020)

For each value of the true standard deviation σ, 1000 datasets are generated. The table reports the average results of the BLS, the Bayesian Lasso (BL) and the Lasso. The columns $\beta_1, \ldots, \beta_8$ list the selected frequency of each coefficient. The average number of nonzero coefficients, NOC, the average root mean square error, RMSE, and the average of estimated standard deviation, $\hat{\sigma}$, are listed along with the corresponding standard errors SE.

Table 4. The average coefficient estimates and corresponding standard errors based on 1000 datasets for simulation 1.

σ	Method	$\beta_1 = 3$	$\beta_2 = 1.5$	$\beta_3 = 0$	$\beta_4 = 0$	$\beta_5 = 2$	$\beta_6 = 0$	$\beta_7 = 0$	$\beta_8 = 0$
1	BLS	2.987 (0.012)	1.459 (0.014)	0.013 (0.005)	0.011 (0.008)	1.964 (0.008)	0.012 (0.007)	0.002 (0.003)	0.001 (0.004)
	BL	2.985 (0.009)	1.489 (0.010)	0.001 (0.007)	0.006 (0.009)	1.970 (0.009)	0.031 (0.010)	−0.017 (0.009)	0.015 (0.009)
	Lasso	2.940 (0.006)	1.437 (0.006)	0.030 (0.004)	0.026 (0.006)	1.889 (0.005)	0.022 (0.003)	0.003 (0.004)	0.001 (0.003)
3	BLS	2.863 (0.017)	1.238 (0.018)	0.080 (0.008)	0.101 (0.009)	1.653 (0.018)	0.071 (0.008)	0.021 (0.006)	0.006 (0.005)
	BL	2.859 (0.018)	1.210 (0.017)	0.058 (0.011)	0.129 (0.013)	1.566 (0.017)	0.097 (0.013)	0.032 (0.011)	0.012 (0.015)
	Lasso	2.807 (0.017)	1.295 (0.018)	0.095 (0.012)	0.060 (0.011)	1.670 (0.017)	0.051 (0.012)	0.039 (0.012)	0.029 (0.014)
5	BLS	2.584 (0.032)	0.944 (0.027)	0.142 (0.011)	0.141 (0.013)	1.142 (0.028)	0.095 (0.010)	0.019 (0.007)	0.005 (0.007)
	BL	2.511 (0.019)	1.172 (0.025)	0.190 (0.012)	0.235 (0.014)	1.309 (0.025)	0.189 (0.012)	0.051 (0.008)	0.009 (0.014)
	Lasso	2.598 (0.021)	1.187 (0.029)	0.109 (0.013)	0.137 (0.015)	1.441 (0.026)	0.115 (0.010)	0.026 (0.009)	0.030 (0.013)

Table 5. Result for simulation 2 and 3, where average results for 1000 datasets are reported.

	Method	NOC (SE)	RMSE (SE)
Simulation 2	BLS	24.33 (0.094)	0.942 (0.006)
	BL	40 (0)	1.553 (0.005)
	Lasso	23.71 (0.180)	0.944 (0.001)
Simulation 3	BLS	47.17 (0.068)	2.331 (0.013)
	BL	60 (0)	29.16 (0.071)
	Lasso	49.40 (0.092)	2.884 (0.013)

The average number of nonzero coefficients, NOC, and the average root mean square error, RMSE, are listed along with the corresponding standard errors SE.

for all cases of σ. Thus, Table 3 shows that the BLS provides on average the most sparse estimated model with the lowest prediction error.

The average of the estimated values of $\boldsymbol{\beta}$ and the corresponding standard errors are presented in Table 4. All methods give similar values for the coefficients, both for the nonzero and zero coefficients. We observe that while Bayesian Lasso never sets a coefficient to zero, the values of these coefficients are close to zero.

In the second simulation study, we simulate data where the dimension P of the input variables is close to the sample size N. We simulate a dataset with 50 observations for the training set and with 40 predictors. We set

$$\boldsymbol{\beta} = (\underbrace{3, \ldots, 3}_{5}, \underbrace{3, \ldots, 3}_{5}, \underbrace{3, \ldots, 3}_{5}, \underbrace{0, \ldots, 0}_{25})^{\top},$$

and $\sigma = 1$. For the variables in $\boldsymbol{X}$ we first generate Z_1, Z_2 and Z_3 independently from $N(0, 1)$.

Then let

$$\begin{aligned} x_i &= Z_1 + e_i, & \text{for } i &= 1, \ldots, 5, \\ x_i &= Z_2 + e_i, & \text{for } i &= 6, \ldots, 10, \\ x_i &= Z_3 + e_i, & \text{for } i &= 11, \ldots, 15, \end{aligned}$$

where $e_i \sim N(0, 0.01)$ for $i = 1, \ldots, 15$. For the remaining 25 variables, $i = 16, \ldots, 40$, we set $x_i \sim N(0, 1)$. The results from Simulation 2 are presented in Table 5. For Simulation 2 the Lasso model barely obtains the lowest RMSE value before BLS. The Bayesian Lasso RMSE is significantly higher. A correct model should select the 15 nonzero coefficients and set the remaining 25 to zero. Table 5 also shows that the average NOC value from both the Lasso and the BLS overestimate the number of coefficient that are different from zero.

In Simulation 3 we simulate a training dataset with 50 observations and with 60 predictors. We set

$$\boldsymbol{\beta} = (\underbrace{5, \ldots, 5}_{10}, \underbrace{3, \ldots, 3}_{10}, \underbrace{3, \ldots, 3}_{10}, \underbrace{2, \ldots, 2}_{10}, \underbrace{2, \ldots, 2}_{10}, \underbrace{0, \ldots, 0}_{10})^{\top}.$$

Table 6. Results of the diabetes data.

Names	Bayesian Lasso	Bayesian CI	BLS	Bayesian CI	Lasso
Age	−3.080	(-89.91, 83.01)	0	NA	0
Sex	−203.27	(-302.67, −103.68)	−196.87	(-316.87, −76.87)	−188.55
Bmi	523.32	(413.92, 632.28)	533.52	(391.62, 675.45)	521.18
Map	301.08	(192.61, 408.54)	304.81	(182.91, 426.71)	292.36
Tc	−144.05	(-414.08, 66.37)	−100.60	(-214.90, 13.70)	−92.98
Ldl	−18.60	(-215.22, 195.23)	0	NA	0
Hdl	−164.54	(-339.62, 8.82)	−221.77	(-373.67, −69.87)	−220.82
Tch	90.31	(-78.20, 290.18)	0	NA	0
Ltg	506.27	(356.19, 663.89)	529.17	(373.87, 684.47)	508.26
Glu	62.18	(-31.35, 163.78)	20.69	(-30.51, 71.89)	50.20

The estimated coefficients of the variables are listed together with the corresponding 95 % Bayesian credible intervals, CI.

Similar to Simulation 2, for each group of ten variables we generate $Z_i \sim N(0,1)$ and set

$$\begin{aligned} x_i &= Z_1 + e_i, \quad \text{for } i = 1, \ldots, 10, \quad & x_i &= Z_2 + e_i, \quad \text{for } i = 11, \ldots, 20, \\ x_i &= Z_3 + e_i, \quad \text{for } i = 21, \ldots, 30, \quad & x_i &= Z_4 + e_i, \quad \text{for } i = 31, \ldots, 40, \\ x_i &= Z_5 + e_i, \quad \text{for } i = 41, \ldots, 50, \end{aligned}$$

with $e_i \sim N(0, 0.01)$. For the remaining 10 variables we set $x_i \sim N(0,1)$. From Table 5 we see that the BLS has the lowest average RMSE value for Simulation 3, slightly lower than the Lasso. The Bayesian Lasso does not perform well when $P > N$ on this dataset, however, we have observed that the RMSE drastically improves when the number of samples increases.

4.2. The diabetes data

The final study uses the diabetes data presented by Efron et al. (2004). The dataset was also used in a study by Park and Casella (2008) to compare the performance of the Bayesian Lasso with the Lasso and the Ridge Regression. The response is a measure of disease progression of 442 patients measured by 10 variables, one year after baseline. We standardize the predictors to have zero mean and unit variance. Table 6 compares the estimates from the BLS, the Bayesian Lasso and the Lasso. For the BLS, the point estimates in $\boldsymbol{\beta}$ are the mean of the posterior distribution. For the Bayesian Lasso, we use the same settings as Park and Casella (2008) and report the posterior median estimates obtained by using the Gibbs sampler. The Bayesian 95% credible intervals are also given. The BLS estimates are within the credible intervals of the Bayesian Lasso for all coefficients. We also notice that the BLS and the Lasso set the same coefficients to zero.

To compare the prediction performance, the dataset is randomly split in two parts, 70 % of the data is used as a training set and the remaining 30 % is used as a test set. We carry out 1000 repetitions and report the average test RMSE and the average number of selected coefficients (NOC) with corresponding standard errors (SE). For the Lasso, we used 10-fold cross-validation to select the value of λ. The results given in Table 7 shows that for this empirical example, although the RMSE is more or less the same for the three methods, BLS is most sparse with lowest number of selected variables.

5. Conclusion

In this paper a new sparse Bayesian learning method, called the Bayesian Lasso Sparse (BLS) method is presented. The main features of the BLS can be summarized in three points: (I) The developed BLS extends the Bayesian Lasso by Park and Casella (2008) to deal with general nonlinear supervised learning problems, with the help of the kernel-based framework in Tipping (2001) and the fast learning process in Tipping and Faul (2003). The resulting method is a sparse Bayesian method that is shown to achieve sparsity in the sample domain and good predictive

Table 7. Results of the diabetes data.

Models	NOC (SE)	RMSE (SE)
BLS	6.47 (0.02)	55.08 (0.082)
BL	10 (0)	55.06 (0.081)
Lasso	8.15 (0.01)	55.12 (0.083)

The data is split into 70 % training data and 30 % test data for 1000 repetitions, where the average test root mean square error, RMSE, and average number of selected coefficients, NOC, are listed along with the corresponding standard errors, SE.

properties. (II) The prior distribution of BLS is conditioned on the variance of random noise, and the BLS is shown to be robust to the irregular datasets and high variance. We analyze how the posterior estimation of the weight parameters is adjusted by the variance of the noise. (III) We present how the BLS can be used in multiple linear regression. The BLS can here achieve variable selection automatically by a pure data-driven process.

The developed BLS is compared to the well-known methods RVM, FLAP, BL, and Lasso in addition to FLAP$_\sigma$ To investigate the performance of the methods, we carry out a comprehensive study with both simulated and real data. For the simulated datasets the methods are exposed to various extents of noise. For the nonlinear regression test cases included in this paper, the FLAP$_\sigma$ mostly performs better than FLAP. We do not observe that the FLAP$_\sigma$ is unstable at early iterations, which was observed for the compressive sensing reported by Babacan, Molina, and Katsaggelos (2010). Our results show that the performance of the BLS is compatible with the other methods, with low test error, few relevance vectors and low bias estimation of σ^2. The code for the BLS method is available at https://github.com/imhelg/BLScode.git.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Ingvild M. Helgøy http://orcid.org/0000-0003-3231-4111
Yushu Li http://orcid.org/0000-0003-4105-9925

References

Agrawal, R. K., F. Muchahary, and M. M. Tripathi. 2019. Ensemble of relevance vector machines and boosted trees for electricity price forecasting. *Applied Energy* 250:540–8. doi: 10.1016/j.apenergy.2019.05.062.

Alhamzawi, R., and H. T. M. Ali. 2018. The Bayesian adaptive lasso regression. *Mathematical Biosciences* 303:75–82. doi: 10.1016/j.mbs.2018.06.004.

Babacan, S. D., R. Molina, and A. K. Katsaggelos. 2010. Bayesian compressive sensing using Laplace priors. *IEEE Transactions on Image Processing* 19 (1):53–63. doi: 10.1109/TIP.2009.2032894.

Boser, B. E., I. M. Guyon, and V. N. Vapnik. 1992. A training algorithm for optimal margin classifiers. In Proceedings of the fifth annual workshop on Computational learning theory, pages 144–152. ACM. doi: 10.1145/130385.130401.

Donoho, D. L., and J. M. Johnstone. 1994. Ideal spatial adaptation by wavelet shrinkage. *Biometrika* 81 (3):425–55. doi: 10.1093/biomet/81.3.425.

Efron, B., T. Hastie, I. Johnstone, and R. Tibshirani. 2004. Least angle regression. *The Annals of Statistics* 32 (2):407–99. doi: 10.1214/009053604000000067.

Hahn, P. R., and C. M. Carvalho. 2015. Decoupling shrinkage and selection in Bayesian linear models: a posterior summary perspective. *Journal of the American Statistical Association* 110 (509):435–48. doi: 10.1080/01621459.2014.993077.

Hastie, T., R. Tibshirani, J. H. Friedman, and J. H. Friedman. 2009. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. New York: Springer.

Huang, K., Y. Guo, X. Guo, and G. Wang. 2014. Heterogeneous Bayesian compressive sensing for sparse signal recovery. *IET Signal Processing* 8 (9):1009–17. doi: 10.1049/iet-spr.2013.0501.

Ji, S., Y. Xue, and L. Carin. 2008. Bayesian compressive sensing. *IEEE Transactions on Signal Processing* 56 (6): 2346–56. doi: 10.1109/TSP.2007.914345.

Kaltwang, S., S. Todorovic, and M. Pantic. 2016. Doubly sparse relevance vector machine for continuous facial behavior estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38 (9):1748–61. doi: 10.1109/TPAMI.2015.2501824.

Kong, D., Y. Chen, N. Li, C. Duan, L. Lu, and D. Chen. 2019. Relevance vector machine for tool wear prediction. *Mechanical Systems and Signal Processing* 127:573–94. doi: 10.1016/j.ymssp.2019.03.023.

Leng, C., M.-N. Tran, and D. Nott. 2014. Bayesian Adaptive Lasso. *Annals of the Institute of Statistical Mathematics* 66 (2):221–44. doi: 10.1007/s10463-013-0429-6.

Leng, C., Y. Lin, and G. Wahba. 2006. A note on the lasso and related procedures in model selection. *Statistica Sinica* 16 (4):1273–84.

Li, Q., and N. Lin. 2010. The bayesian elastic net. *Bayesian Analysis* 5 (1):151–70. doi: 10.1214/10-BA506.

Liu, D., J. Zhou, D. Pan, Y. Peng, and X. Peng. 2015. Lithium-ion battery remaining useful life estimation with an optimized relevance vector machine algorithm with incremental learning. *Measurement* 63:143–51. doi: 10.1016/j.measurement.2014.11.031.

Liu, S., Y. D. Zhang, T. Shan, and R. Tao. 2018. Structure-aware Bayesian compressive sensing for frequency-hopping spectrum estimation with missing observations. *IEEE Transactions on Signal Processing* 66 (8):2153–66. doi: 10.1109/TSP.2018.2806351.

Liu, X., X. Chen, J. Li, X. Zhou, and Y. Chen. 2020. Facies identification based on multikernel relevance vector machine. *IEEE Transactions on Geoscience and Remote Sensing* 58 (10):7269–82. doi: 10.1109/TGRS.2020.2981687.

MacKay, D. J. 1992. The evidence framework applied to classification networks. *Neural Computation* 4 (5):720–36. doi: 10.1162/neco.1992.4.5.720.

Mohsenzadeh, Y., H. Sheikhzadeh, and S. Nazari. 2016. Incremental relevance sample-feature machine: A fast marginal likelihood maximization approach for joint feature selection and classification. *Pattern Recognition* 60:835–48. doi: 10.1016/j.patcog.2016.06.028.

Park, T., and G. Casella. 2008. The Bayesian Lasso. *Journal of the American Statistical Association* 103 (482):681–6. doi: 10.1198/016214508000000337.

Qiao, W., K. Huang, M. Azimi, and S. Han. 2019. A novel hybrid prediction model for hourly gas consumption in supply side based on improved whale optimization algorithm and relevance vector machine. *IEEE Access*. 7: 88218–30. doi: 10.1109/ACCESS.2019.2918156.

Schmolck, A., and R. Everson. 2007. Smooth relevance vector machine: A smoothness prior extension of the RVM. *Machine Learning* 68 (2):107–35. doi: 10.1007/s10994-007-5012-z.

Schölkopf, B., A. Smola, and K.-R. Müller. 1998. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation* 10 (5):1299–319. doi: 10.1162/089976698300017467.

Schölkopf, B., C. J. Burges, A. J. Smola, et al. 1999. *Advances in kernel methods: Support vector learning*. Cambridge, MA, USA: MIT Press.

Steinwart, I., and A. Christmann. 2008. Support vector machines. In *Information science and statistics*. New York: Springer.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)* 58 (1):267–88.

Tien Bui, D., H. Shahabi, A. Shirzadi, K. Chapi, N.-D. Hoang, B. Pham, Q.-T. Bui, C.-T. Tran, M. Panahi, B. Bin Ahmad, et al. 2018. A novel integrated approach of relevance vector machine optimized by imperialist competitive algorithm for spatial modeling of shallow landslides. *Remote Sensing* 10 (10):1538. doi: 10.3390/rs10101538.

Tipping, M. E. 2001. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research* 1:211–44.

Tipping, M. E., and A. C. Faul. 2003. Fast marginal likelihood maximisation for sparse Bayesian models. In Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics, 276–83.

Vapnik, V., S. E. Golowich, and A. J. Smola. 1996. Support vector method for function approximation, regression estimation and signal processing. In *Advances in neural information processing systems, NIPS'96*, 281–7. MIT Press. https://mitpress.mit.edu/9780262100656/advances-in-neural-informationprocessing-systems-9/

Yu, L., C. Wei, J. Jia, and H. Sun. 2016. Compressive sensing for cluster structured sparse signals: Variational Bayes approach. *IET Signal Processing* 10 (7):770–9. doi: 10.1049/iet-spr.2014.0157.

Zhou, Z., K. Liu, and J. Fang. 2015. Bayesian compressive sensing using normal product priors. *IEEE Signal Processing Letters* 22 (5):583–7. doi: 10.1109/LSP.2014.2364255.

Appendix A

In order to obtain the expression for the log likelihood in Equation (14), we decompose the covariance matrix in the log-likelihood in Equation (13) as:

$$\begin{aligned} \boldsymbol{C} &= \sigma^2 \boldsymbol{I} + \sum_{m \neq i} \sigma^2 \tau_m \phi_m \phi_m^T + \sigma^2 \tau_i \phi_i \phi_i^T \\ &= \boldsymbol{C}_{-i} + \sigma^2 \tau_i \phi_i \phi_i^T, \end{aligned} \tag{23}$$

where $\boldsymbol{C}_{-i}$ denotes $\boldsymbol{C}$ without the inclusion of basis function i. We next use the Woodbury identity on the expression for the covariance matrix in Equation (23), such that the inverse of the covariance matrix is written as:

$$\boldsymbol{C}^{-1} = \boldsymbol{C}_{-i}^{-1} - \frac{\boldsymbol{C}_{-i}^{-1} \phi_i \phi_i^T \boldsymbol{C}_{-i}^{-1}}{\sigma^{-2} \tau_i^{-1} + \phi_i^T \boldsymbol{C}_{-i}^{-1} \phi_i}.$$

Finally we use the determinant identity to obtain the decomposition of the determinant:

$$|\boldsymbol{C}| = |\boldsymbol{C}_{-i}||1 + \sigma^2 \tau_i \phi_i^T \boldsymbol{C}_{-i}^{-1} \phi_i|.$$

These last two expressions can be inserted in Equation (13), which results in Equation (14).

Appendix B

From the decomposition of the log likelihood given in Equation (14), we can find the derivative of L with respect to τ_i, where the other parameters are considered as fixed.

$$\begin{aligned} \frac{\mathrm{d}L}{\mathrm{d}\tau_i} &= \frac{1}{2}\left[-\frac{s_i}{\sigma^{-2} + \tau_i s_i} + \frac{q_i^2 \sigma^{-2}}{(\sigma^{-2} + \tau_i s_i)^2} - \lambda \right] \\ &= -\frac{(\tau_i^2 \kappa_1 + \tau_i \kappa_2 + \kappa_3)}{2(\sigma^{-2} + \tau_i s_i)^2}, \end{aligned}$$

where $\kappa_1 = \lambda s_i^2, \kappa_2 = s_i^2 + 2 s_i \lambda \sigma^{-2}$ and $\kappa_3 = \sigma^{-2}(\lambda \sigma^{-2} + s_i - q_i^2)$. The numerator has a quadratic form while the denominator is always positive so that $dL/d\tau_i = 0$ is satisfied at

$$\tau_i = \frac{-s_i(s_i + 2\lambda\sigma^{-2}) \pm s_i \sqrt{\Theta}}{2\lambda s_i^2}, \tag{24}$$

where $\Theta = (s_i + 2\lambda\sigma^{-2})^2 - 4\lambda\sigma^{-2}(\lambda\sigma^{-2} - (q_i^2 - s_i))$. By analyzing the terms we see that if $q_i^2 - s_i < \lambda\sigma^{-2}$, then $\Theta^2 < s_i + 2\lambda\sigma^{-2}$, and both solutions of Equation (24) are negative. Furthermore, since $\mathrm{d}L/\mathrm{d}\tau_i$ evaluated at $\tau_i = 0$ is negative, the maximum occurs at $\tau_i = 0$. In the other situation, when $q_i^2 - s_i > \lambda\sigma^{-2}$, there are two real solutions of Equation (24), one negative and one positive. The positive solution from Equation (24) maximizes L since $\mathrm{d}L/\mathrm{d}\tau_i$ is positive when evaluated at $\tau_i = 0$ and negative at $\tau_i = \infty$. The maximum of L, when holding the remaining components fixed, is therefore obtained at:

$$\tau_i = \begin{cases} \dfrac{-s_i(s_i + 2\lambda\sigma^{-2}) + s_i\sqrt{\Theta}}{2\lambda s_i^2} & \text{if } q_i^2 - s_i > \lambda\sigma^{-2} \\ 0 & \text{otherwise.} \end{cases}$$

Notice that the expression for Θ can be simplified to $\Theta = s_i^2 + 4q\lambda\sigma^{-2}$ which results in Equation (16).