Neurocomputing 72 (2009) 2952–2963

Contents lists available at ScienceDirect

Neurocomputing

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

An experimental study on methods for the selection of basis functions in regression

Ignacio Barrio [1], Enrique Romero, Lluís Belanche *

Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain

ARTICLE INFO

Article history:
Received 14 May 2008
Received in revised form
21 April 2009
Accepted 26 April 2009
Communicated by T. Heskes
Available online 15 May 2009

Keywords:
Selection of basis functions
Evidence maximization
Regression

ABSTRACT

A comparative study is carried out in the problem of selecting a subset of basis functions in regression tasks. The emphasis is put on practical requirements, such as the *sparsity* of the solution or the *computational effort*. A distinction is made according to the *implicit* or *explicit* nature of the selection process. In explicit selection methods the basis functions are selected from a set of candidates with a search process. In implicit methods a model with all the basis functions is considered and the model parameters are computed in such a way that several of them become zero. The former methods have the advantage that both the sparsity and the computational effort can be controlled. We build on earlier work on Bayesian interpolation to design efficient methods for explicit selection guided by *model evidence*, since there is strong indication that the evidence prefers simple models that generalize fairly well. Our experimental results indicate that very similar results between implicit and explicit methods can be obtained regarding generalization performance. However, they make use of different numbers of basis functions and are obtained at very different computational costs. It is also reported that the models with the highest evidence are not necessarily those with the best generalization performance.

1. Introduction

In regression tasks we are given a data set consisting of input vectors $\{x_n\}_{n=1}^N$ and their corresponding target values $\{t_n\}_{n=1}^N$, where $t_n \in \mathbb{R}$. The goal is to infer a function $y(x)$ that underlies the training data and makes good predictions on unseen vectors. A very common sort of function is obtained as a linear basis expansion with M *fixed* basis functions ϕ_i:

$$y(x; w) = \sum_{i=1}^{M} \omega_i \phi_i(x), \tag{1}$$

where $w = (\omega_1, \omega_2, \ldots, \omega_M)^T$ are the *model parameters*. Since the model is linear with respect to the output of the basis functions, the parameters w_i are typically estimated with maximum likelihood or Bayesian techniques. The main problem lies on the selection of the M basis functions (where M is unknown a priori) from a *dictionary*. For instance, selected from a set of Gaussian radial basis functions (RBF) centered at the input vectors:

$$\phi_i(x) = \exp\left(-\sum_{d=1}^{D} \frac{(x_d - x_{id})^2}{r_d^2}\right),$$

where D is the dimension of the input vectors and there is a width r_d for every dimension.

This problem has been tackled in different ways, according to the *implicit* or *explicit* nature of the selection process. In implicit selection methods, the model with the whole set of basis functions is considered and the parameters are computed in such a way that several of them become zero. An example of such method is the relevance vector machine (RVM) [26], which assumes a particular prior on the parameters of a linear model, so that making inference implicitly produces sparse solutions. Other examples are the support vector machine (SVM) [28], basis pursuit (BP) [5] or the least absolute shrinkage and selection operator (LASSO).

In explicit selection methods a search is carried out guided by the minimization of some cost function. This category includes matching pursuits [13], orthogonal least squares [4], kernel matching pursuit [29] or some Gaussian process approximations [23,17]. Some approaches to the SVM in classification tasks use an explicit selection to control sparsity [8].

Explicit selection methods have two criteria: an *objective function* that conducts the search (e.g., the sum-of-squares error in the training set) and an *evaluation criterion* that checks model performance (e.g., the sum-of-squares error in a validation set). This evaluation criterion can then be used with other information (e.g., current number of basis functions in the expansion) to stop the process. However, minimizing the first criterion does not necessarily entail the minimization of the second. This fact

* Corresponding author.
E-mail addresses: ibarrio@lsi.upc.edu (I. Barrio), eromero@lsi.upc.edu (E. Romero), belanche@lsi.upc.edu (L. Belanche).
[1] In Loving Memory. You have gone to a place where the sun never hides.

0925-2312/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2009.04.015

hinders the use of more sophisticated search strategies, because they could lead to an overfit solution. Actually, all the mentioned explicit methods use *forward selection* as the search strategy.

The objective of this work is to ascertain to what extent explicit search methods are able to obtain competitive results when compared to those of the well-established implicit methods, as the RVM or the SVM. A further goal of the paper is to make practical use of the evidence theory. Within a Bayesian approach, the use of the evidence has been suggested to compare different models [11], given that it penalizes complex models and there is some (anti)correlation between model evidence and generalization error [10]. MacKay did not consider sparse solutions, but showed instead that the evidence prefers simple models that generalize fairly well. The benefits of evidence maximization rely in that it takes into account not only a reasonable fit to the training data, but in that it penalizes over-complex or over-simple models. However, evidence and generalization error are not quite the same thing.

We build on earlier work on Bayesian interpolation to design efficient explicit methods guided by the maximization of *model evidence*. The search strategies are borrowed from the feature selection field, exchanging *features* for *basis functions*. We call such methods as *search strategy guided by the evidence* (SSGE). The developed methods have the added advantage that both the sparsity and the computational effort can be under tighter control by choosing different search strategies, which is very difficult or not possible in the RVM or the standard SVM. Sparsity control may certainly enhance interpretability since the chosen basis functions might be of interest for the application at hand. It also entails reduced storage requirements, which can be an issue for real-time embedded systems.

In order to experimentally assess these goals in a controlled way, our study makes use and includes data sets from the DELVE environment [21]. Our results show that the generalization performance is very similar between implicit and explicit methods. However, they make use of different numbers of basis functions and are obtained at different computational costs. In particular, some SSGEs are able to find very compact models that are competitive with state-of-the-art implicit techniques, such as SVMs and RVMs. More sophisticated SSGEs tend to find even more compact models, at the expense of a slightly worse generalization rate. Parenthetically, it is found that the models with the highest evidence are not necessarily those with the best generalization performance. In addition, in some cases sparsity is shown to be a computational necessity.

The rest of this work is organized as follows. In Section 2 the selection of basis functions for linear models is reviewed. In Section 3 the Bayesian approach for regression with linear models is described. In Section 4 the used search strategies for basis function selection are introduced. Section 5 develops algorithms and a fast implementation for the SSGEs. The experimental study comparing explicit and implicit methods is carried out in Section 6, the results of which are discussed in Section 7.

2. Selection of basis functions for linear models

In machine learning, using a dictionary of basis functions (BFs) centered *at known data* usually gives good results [24]. A line can be drawn between the methods that additionally require the satisfaction of Mercer's condition (viz. kernel methods) and those that do not.

2.1. Implicit methods

Implicit selection methods set the parameters of the unnecessary BFs to zero (or close to), therefore obtaining *sparse* models. This is the case of the RVM, the SVM or BP, among others.

The SVM for regression tasks tries to approximate the data up to some given precision ε (allowing for some errors) while minimizing the Euclidean norm of the model parameters in feature space. This optimization problem can be solved in a dual space with quadratic programming techniques. The solution is sparse because in the dual space only the support vectors, outside the ε-tube, have non-zero parameters (the Lagrange multipliers). The selection of BFs is thus implicitly done: only those associated with the support vectors are present in the final model. The BFs need to be related to a dot product in some feature space (i.e., kernels centered at the input data).

BP fits the data exactly while minimizing the ℓ^1 norm of the model parameters. This requires the solution of a convex, non-quadratic optimization problem which can be reformulated as an equivalent linear program. Basis pursuit de-noising (BPDN) considers data with Gaussian noise (some residual errors are allowed in the fit). The problem can be reformulated as a perturbed linear program and is similar to an SVM minimizing the ℓ^1 norm. BP and BPDN produce sparse solutions in which there are no requirements about the candidate BFs.

Sparse Bayesian learning (SBL) [26] assumes a Gaussian prior probability for the parameters and assigns one hyper-parameter (controlling the scale) for each parameter. The value of each hyper-parameter is optimized in the second level of Bayesian inference by maximizing the marginal likelihood. Most of the hyper-parameters tend to infinity, and their corresponding parameters tend to zero. The RVM is a special case of SBL where the model has identical functional form to the SVM, that is, the BFs are kernels centered at the input data.

2.2. Explicit methods

Explicit selection methods perform a search to select the most relevant BFs, mostly using forward selection. Some popular methods are projection pursuit (PP) [6], matching pursuit (MP) [13], orthogonal least squares (OLS) [4] or some approximations to Gaussian processes (GP approx.) [17].

MP and PP add at each step the BF that best approximates the residual error. The parameter corresponding to this BF is then set while the others are kept fixed. The BFs do not have to satisfy any restriction. Optionally, MP recalculates the parameters of the whole model after several steps.

OLS adds at each step the BF that reduces the most the sum-of-squares error. The BFs are Gaussians centered at the input data. Kernel matching pursuit with pre-fitting (KMPP) [29] minimizes the same cost function than OLS using kernel BFs. Sequential approximation with optimal coefficients and interacting frequencies (SAOCIF) [22] generalizes OLS and KMPP to non-input or non-kernel BFs. To avoid the problem of overfitting the cost function can be adjusted with *regularization*, as in regularized forward selection (RFS) [14] and regularized OLS (ROLS) [3].

Gaussian processes have been approximated with finite generalized linear models [23], where the BFs forming the covariance matrix are kernels centered at the input data. In [17] the subset of BFs is selected with forward selection maximizing the marginal likelihood. This approach has some connection with the present work; however, the use of different assumptions leads to a different marginal likelihood.

All these methods are categorized in Table 1 according to the selection (implicit or explicit) and the requirements for the BFs (kernel or non-kernel).

3. A Bayesian approach for linear models

In the Bayesian framework, three levels of inference can be distinguished: the first considers the posterior distribution over

Table 1
Classification of methods according to the sort of selection and the requirements for the basis functions.

Selection	Basis functions	Method
Implicit	Kernel	SVM, RVM
	Non-kernel	BP, BPDN, SBL
Explicit	Kernel	GP approx., KMPP
	Non-kernel	MP, PP, SAOCIF, OLS, RFS, ROLS

'Non-kernel' means that the basis functions do not need to be kernels.

the parameters, the second adapts the hyper-parameters that control the parameters and the third allows the comparison of different models [11]. For a regression task we consider the targets to be deviated from the real underlying function by independent additive noise:

$$t_n = y(x_n; w) + v_n. \tag{2}$$

We are working with linear models and the target values are assumed to be

$$t = \Phi w + v, \tag{3}$$

where $t = (t_1, t_2, \ldots, t_N)^T$ and Φ is the $N \times M$ design matrix with elements $\Phi_{ij} = \phi_j(x_i)$. If v is assumed to be zero-mean Gaussian noise with variance σ^2, then the probability of the data given the parameters (the *likelihood* of the parameters) is

$$P(t|w,\beta) = \mathcal{N}(\Phi w, \beta^{-1}), \tag{4}$$

where $\beta = 1/\sigma^2$. This probability should be $P(t|w,\beta,x)$, but we omit the conditioning on the input vectors throughout the work for brevity.

3.1. *The first level of inference*

The process of finding the parameters that maximize the likelihood may lead to overfitting [1]. In order to avoid it, the smoothness of $y(x;w)$ is controlled by the definition of a prior distribution. A common choice is a zero-mean Gaussian prior distribution over w,

$$P(w|\alpha) = \mathcal{N}(0, \alpha^{-1}), \tag{5}$$

where α is the precision (inverse variance) of an overall prior from which all parameters are sampled. This hyper-parameter measures how smooth $y(x;w)$ is expected to be. Since the likelihood and the prior are Gaussian, the posterior parameter distribution is also Gaussian and it can be written as

$$\begin{aligned} P(w|t,\alpha,\beta) &= \frac{P(t|w,\beta)P(w|\alpha)}{P(t|\alpha,\beta)} \\ &= (2\pi)^{-M/2}|\Sigma|^{-1/2}\exp\{-\tfrac{1}{2}(w-\mu)^T\Sigma^{-1}(w-\mu)\}, \end{aligned} \tag{6}$$

where

$$\Sigma = (\beta\Phi^T\Phi + \alpha I)^{-1} \quad \text{and} \quad \mu = \beta\Sigma\Phi^T t. \tag{7}$$

The parameters w of the model are set to their most probable value μ, a solution equivalent to the minimization of a weight decay error function (as in ridge regression) [1].

3.2. *The second level of inference*

The marginal likelihood $P(t|\alpha,\beta) = \int P(t|w,\beta)P(w|\alpha)\,dw$ is the convolution of Gaussians, which is also a Gaussian:

$$\begin{aligned} P(t|\alpha,\beta) &= (2\pi)^{-N/2}|\beta^{-1}I + \alpha^{-1}\Phi\Phi^T|^{-1/2} \\ &\quad \times \exp\{-\tfrac{1}{2}t^T(\beta^{-1}I + \alpha^{-1}\Phi\Phi^T)^{-1}t\}. \end{aligned} \tag{8}$$

In order to find the most suitable values for α and β, we make use of Bayes' formula:

$$P(\alpha,\beta|t) = \frac{P(t|\alpha,\beta)P(\alpha,\beta)}{P(t)}. \tag{9}$$

Assuming we have no prior idea of suitable values for α and β, we consider the prior $P(\alpha,\beta)$ to be uniform on a logarithmic scale $\log\alpha$ and $\log\beta$ (improper prior). The most suitable values for α and β are those that maximize $P(t|\alpha,\beta)$. In order to do so, one can make an arbitrary choice such that terms in (8) are independent in the derivatives, differentiate (8) and set the result to zero. This produces the re-estimation formulae as

$$\alpha_{new} = \frac{\gamma}{\|\mu\|^2} \quad \text{and} \quad \beta_{new} = \frac{N-\gamma}{\|t - \Phi\mu\|^2}, \tag{10}$$

where

$$\gamma = M - \alpha\,\mathrm{tr}\,\Sigma \tag{11}$$

is known as the number of well-determined parameters [11]. Eqs. (10) are dependent on the most probable parameters μ, and also on α and β. In order to find suitable values for α, β and μ, the quantities can be reestimated iteratively, starting from an initial guess for α and β, until convergence. The optimization of α and β can also be performed through an expectation–maximization (EM) algorithm, leading to different reestimation formulae that, unlike MacKay's, are guaranteed to increase the marginal likelihood. In practice, MacKay's method converges to the same values faster than its EM counterpart [15].

For prediction purposes, given a new input vector x_*, we seek the probability of the corresponding target t_*, marginalizing α and β as $P(t_*|t) = \int P(t_*|t,\alpha,\beta,w)P(\alpha,\beta,w)\,d\alpha\,d\beta\,dw$. In practice, this integral is commonly approximated by $P(t_*|t,\alpha_{MP},\beta_{MP})$:

$$P(t_*|t,\alpha_{MP},\beta_{MP}) = \int P(t_*|w,\alpha_{MP},\beta_{MP})P(w|t,\alpha_{MP},\beta_{MP})\,dw, \tag{12}$$

where α_{MP} and β_{MP} represent the (most probable) estimates for α and β. Again, this is a Gaussian, with mean and variance (error bars) given by

$$y_* = \mu^T\phi(x_*), \tag{13}$$

$$\sigma_*^2 = \beta^{-1} + \phi(x_*)^T\Sigma\phi(x_*). \tag{14}$$

3.3. *The third level of inference*

Supposing we have a set of models, $\mathcal{H}_i$, containing different subsets of BFs $\{\phi_i\}$, their posterior probability given the data set is

$$P(\mathcal{H}_i|t) = \frac{P(t|\mathcal{H}_i)P(\mathcal{H}_i)}{P(t)}, \tag{15}$$

where $P(\mathcal{H}_i)$ is the prior probability of model $\mathcal{H}_i$. Given a new input vector x_*, a full Bayesian approach would use a committee with all the models to make predictions: $P(t_*|t) = \sum_i P(t_*|t,\mathcal{H}_i)P(\mathcal{H}_i|t)$. As a practical approximation to full Bayesian inference, only the model with the highest $P(\mathcal{H}_i|t)$ is commonly used and then Eq. (12) is used to approximate $P(t_*|t,\mathcal{H}_i)$. Assuming there is no reason to assign different priors to different models, then they can be ranked by evaluating the fully marginalized probability $P(t|\mathcal{H}_i)$, known as the *evidence* for the model. Integrating out α and β from (8):

$$P(t|\mathcal{H}_i) = \int P(t|\alpha,\beta,\mathcal{H}_i)P(\alpha,\beta|\mathcal{H}_i)\,d\alpha\,d\beta, \tag{16}$$

where $P(t|\alpha,\beta,\mathcal{H}_i)$ is the marginal likelihood (8) with the dependency on the model made explicit. This analytically intractable integral has been approximated [11] with a separable

Gaussian around $P(t|\alpha_{MP}, \beta_{MP}, \mathscr{H}_i)$:

$$P(t|\mathscr{H}_i) \simeq P(t|\alpha_{MP}, \beta_{MP}, \mathscr{H}_i)P(\alpha_{MP}, \beta_{MP}|\mathscr{H}_i)2\pi\sqrt{\sigma^2_{\log\alpha}\sigma^2_{\log\beta}}, \tag{17}$$

where $\sigma^2_{\log\beta} = 2/(N-\gamma)$ is the variance of the Gaussian approximation for $\log\beta$ and $\sigma^2_{\log\alpha} = 2/\gamma$ is the variance for $\log\alpha$, and they are both found by differentiating (8) twice. This Gaussian approximation holds good for $\gamma \gg 1$ and $N-\gamma \gg 1$ [12].

Again $P(\alpha, \beta|\mathscr{H}_i)$ is considered, as above, a flat prior over $\log\alpha$ and $\log\beta$, so it cancels out when comparing different models. Dropping constant terms, the evidence (or, for convenience, its logarithm) can be approximated by

$$\begin{aligned}\log P(t|\mathscr{H}_i) &\simeq \log P(t|\alpha_{MP}, \beta_{MP}, \mathscr{H}_i) + \frac{1}{2}\log\left(\frac{2}{\gamma}\right) + \frac{1}{2}\log\left(\frac{2}{N-\gamma}\right)\\ &= -\frac{1}{2}\left[N\log 2\pi + \log|C| + t^TC^{-1}t - \log\left(\frac{2}{\gamma}\right)\right.\\ &\quad\left. - \log\left(\frac{2}{N-\gamma}\right)\right],\end{aligned} \tag{18}$$

where $C = \beta^{-1}_{MP}I + \alpha^{-1}_{MP}\Phi\Phi^T$.

3.4. The relevance vector machine

The RVM is a Bayesian approximation method that considers an individual Gaussian prior for each parameter, instead of using an overall prior for all the parameters (5). It can also be seen as a special case of Sparse Bayesian learning [26]. Training consists on iteratively optimizing the hyper-parameters until a local maximum of the evidence is found. The number of hyper-parameters is very high, which can lead to slow convergence, even when a fast algorithm is used. Following inference on the specific prior produces very sparse solutions. However, this sparsity may come at the expense of worse predictive distributions [20]. An individual Gaussian prior on the parameters has been proposed [26]:

$$P(w|\alpha) = \prod_{i=0}^{M}\mathscr{N}(w_i|0, \alpha_i^{-1}). \tag{19}$$

The posterior for the parameters and the marginal likelihood are still Gaussian. The marginal likelihood can be written as

$$P(t|\alpha, \beta) = \mathscr{N}(0, \Phi A^{-1}\Phi^T + \beta^{-1}I), \tag{20}$$

where $A = diag(\alpha_1, \ldots, \alpha_M)$. Eq. (8) is a particular case of (20), where $A = \alpha I$. In the second level of inference, the hyper-parameters (β and α_i) are optimized by maximizing the marginal likelihood. Initially, the number of BFs equals the number of data ($M = N$); as β and α_i are optimized, most of the α_i tend to infinity, so their corresponding w_i tend to zero, obtaining sparse solutions ($M \ll N$). As in finite linear models, the predictive variances of the RVM are overconfident, which can be healed through augmentation [20].

4. Feature selection search strategies

The main objective of feature selection in inductive learning is selecting the most suitable subset from a set of features [9]. A common approach is a search process with three basic elements: an objective function for subset evaluation, a strategy to decide how to continue exploring and an initial state where the search starts from. Some popular algorithms are

PTA(l, r): Plus l and take away r [9]. At every step, l features are added one at a time and then r features are removed one at a time (always the one that locally optimizes the objective function).

SFFS: Sequential forward floating selection [16]. At every step, a feature is added and then zero or more features are removed one at a time, while the value of the objective function is better than the best value achieved so far with same numbers of features. Note that subset size does not grow constantly with respect to the number of steps, but in a staggered way.

Oscil(c): Oscillating algorithm with parameter c [25]. In simplified form, let s:=1. Add s features, then remove 2s features and add s features (always one at a time). If the objective function has been optimized, let s:=1 and repeat. If not, let s:=$s+1$ and repeat, unless $s = c$. The final solution has as many features as the initial one.

PTA(1,0)—or forward selection—is usually the fastest method, but its solutions tend to be the worse. SFFS usually finds better solutions than PTA [7]. One of the advantages of the PTA(l,r) family over other methods is that the number of steps (and the number of added/removed features) is known in advance. In the next section, we develop specific search strategies inspired on these algorithmic ideas, exchanging *features* for *basis functions*.

5. Search strategies guided by the evidence

A Bayesian approximation suggests the use of the evidence to compare different models. Models with higher evidence will usually generalize better and have a sensible number of BFs. There is, however, the question of which models to compare. One may use heuristics such as using h equally spaced radial BFs over the range of interest and vary h to find alternative models [11]. Alternatively, we develop evidence-guided search algorithms for the selection of good subsets of BFs from a set of candidates.

5.1. Pseudocode for the SSGEs

In order to implement the SSGEs, it is enough to provide pseudocode for the addition (of the best) and elimination (of the worst) BF, as well as for the reestimation of α and β. The pseudocode for these three functions follows. An efficient implementation of these operations is also developed. A *model* is defined by a set of BFs $\{\phi_i\}$ and the posterior distribution over the parameters (w, Σ). We use φ_i to denote the candidate BFs and ϕ_i to denote the BFs present in the current model.

```
AddBestBasisFunction (a model $\mathscr{H}$, $\alpha$, $\beta$, a set of candidate BFs $\{\varphi_i\}$)
1.      for each candidate BF $\varphi_i$ do
2.          set $\mathscr{H}'$ the model obtained by adding $\varphi_i$ to $\mathscr{H}$ and computing
              the initial value for the parameters $w'$ and $\Sigma'$ with equations (7)
3.          $(\alpha', \beta', w', \Sigma')$ :=Reestimate($\alpha, \beta, \mathscr{H}'$)
4.          compute the evidence for $\mathscr{H}'$ with equation (17) using $\alpha'$ and $\beta'$
5.      end for
6.      set $\mathscr{H}$ the model obtained by adding to $\mathscr{H}$ the $\varphi_i$ that
          maximizes the evidence in the previous loop and compute the
          initial value for the parameters $w$ and $\Sigma$ with equations (7)
7.      $(\alpha, \beta, w, \Sigma)$ :=Reestimate($\alpha, \beta, \mathscr{H}$)
8.      return $(\alpha, \beta, \mathscr{H})$
end AddBestBasisFunction

RemoveWorstBasisFunction (a model $\mathscr{H}$, $\alpha$, $\beta$)
9.      for each basis_function $\phi_i$ in $\mathscr{H}$
10.         set $\mathscr{H}'$ the model obtained by removing $\phi_i$ from $\mathscr{H}$ and computing
            the initial value for the parameters $w'$ and $\Sigma'$ with equations (7)
11.         $(\alpha', \beta', w', \Sigma')$ :=Reestimate($\alpha, \beta, \mathscr{H}'$)
12.         compute the evidence for $\mathscr{H}'$ with equation (17) using $\alpha'$ and $\beta'$
13.     end for
14.     set $\mathscr{H}$ the model obtained by removing from $\mathscr{H}$ the$\phi_i$ that
          maximizes the evidence in the previous loop and compute
          the initial value for the parameters $w$ and $\Sigma$ with equations (7)
15.     $(\alpha, \beta, w, \Sigma)$ :=Reestimate($\alpha, \beta, \mathscr{H}'$)
```

```
16.         return $(\alpha, \beta, \mathscr{H})$
end RemoveWorstBasisFunction

Reestimate $(\alpha, \beta$, a model $\mathscr{H})$
17.         while not Convergence$(\alpha, \beta, \mathscr{H})$ do
18.             estimate $\alpha$ and $\beta$ with equations (10)
19.             estimate the parameters $w$ and $\Sigma$ with equations (7)
20.         end while
21.         return$(\alpha, \beta, w, \Sigma)$
end Reestimate
```

5.2. Fast implementation of SSGEs

Some properties of the marginal likelihood have been exploited to devise a fast implementation of the relevance vector machine [27]. Some of these are also useful for model evidence maximization in an incremental way, as explained next.

5.2.1. Addition

Addition of the candidate BF φ_i makes the C matrix become

$$C_{+i} = \beta^{-1} I + \alpha^{-1} \Phi \Phi^T + \alpha^{-1} \varphi_i \varphi_i^T = C + \alpha^{-1} \varphi_i \varphi_i^T, \tag{21}$$

where C_{+i} is C, after the inclusion of BF i. The determinant and inverse are then written as

$$|C_{+i}| = |C||1 + \alpha^{-1} \varphi_i^T C^{-1} \varphi_i|, \tag{22}$$

$$C_{+i}^{-1} = C^{-1} - \frac{C^{-1}\varphi_i \varphi_i^T C^{-1}}{\alpha + \varphi_i^T C^{-1} \varphi_i}. \tag{23}$$

Applying them to the marginal likelihood (8), we get

$$\begin{aligned} \log P(t|\alpha, \beta, \mathscr{H}_{+i}) = {} & \log P(t|\alpha, \beta, \mathscr{H}) \\ & + \frac{1}{2}\left[\log \alpha - \log(\alpha + \varphi_i^T C^{-1} \varphi_i) + \frac{(\varphi_i^T C^{-1} t)^2}{\alpha + \varphi_i^T C^{-1} \varphi_i}\right] \\ = {} & \log P(t|\alpha, \beta, \mathscr{H}) \\ & + \frac{1}{2}\left[\log \alpha - \log(\alpha + S_i) + \frac{Q_i^2}{\alpha + S_i}\right], \end{aligned} \tag{24}$$

where $\mathscr{H}_{+i}$ is the model $\mathscr{H}$ with BF i included and we have defined $S_i \triangleq \varphi_i^T C^{-1} \varphi_i$ and $Q_i \triangleq \varphi_i^T C^{-1} t$. Using the Woodbury identity we can write:

$$S_m = \beta \varphi_m^T \varphi_m - \beta^2 \varphi_m^T \Phi \Sigma \Phi^T \varphi_m, \tag{25}$$

$$Q_m = \beta \varphi_m^T t - \beta^2 \varphi_m^T \Phi \Sigma \Phi^T t. \tag{26}$$

The log-evidence (18) can then be written in an incremental way as

$$\begin{aligned} \log P(t|\mathscr{H}_{+i}) = {} & \log P(t|\mathscr{H}) + \frac{1}{2}\left[\log \alpha - \log(\alpha + S_i) + \frac{Q_i^2}{\alpha + S_i}\right. \\ & + \log\left(\frac{2}{\gamma_{+i}}\right) + \log\left(\frac{2}{N - \gamma_{+i}}\right) - \log\left(\frac{2}{\gamma}\right) \\ & \left. - \log\left(\frac{2}{N - \gamma}\right)\right], \end{aligned} \tag{27}$$

where γ_{+i} is the number of well determined parameters after adding the BF i. To calculate γ_{+i}, the trace of Σ_{+i} should be computed (see Eq. (11)). We have

$$\Sigma_{+i} = \begin{pmatrix} \Sigma + \beta^2 \Sigma_{ii} \Sigma \Phi^T \varphi_i \varphi_i^T \Phi \Sigma & -\beta \Sigma_{ii} \Sigma \Phi^T \varphi_i \\ -\beta \Sigma_{ii} (\Sigma \Phi^T \varphi_i)^T & \Sigma_{ii} \end{pmatrix}, \tag{28}$$

where $\Sigma_{ii} = (\alpha + S_i)^{-1}$. We can write the trace as

$$\operatorname{tr} \Sigma_{+i} = \Sigma_{ii} + \beta^2 \Sigma_{ii} R_i + \operatorname{tr} \Sigma, \tag{29}$$

where we have defined $R_i \triangleq \varphi_i^T \Phi \Sigma \Sigma \Phi^T \varphi_i$. Note that the trace of a product of a column vector by a row vector equals the product of the row vector by the column vector.

To add/remove BFs, it is convenient to maintain and update the values S_m, Q_m and R_m, for each candidate BF φ_m. When selecting the BF to add, we need to select the one that increments the log-evidence (27) the most. This increment can be computed as

$$\begin{aligned} 2(\log P(t|\mathscr{H}_{+i}) - \log P(t|\mathscr{H})) = {} & \log(\alpha) - \log(\alpha + S_i) + \frac{Q_i^2}{\alpha + S_i} \\ & + \log\frac{\gamma}{\gamma_{+i}} + \log\frac{N - \gamma}{N - \gamma_{+i}}, \end{aligned} \tag{30}$$

where

$$\gamma_{+i} = M + 1 - \alpha \operatorname{tr} \Sigma_{+i}. \tag{31}$$

If by adding the BF i, the values of α and β are not modified, S_m, Q_m and R_m can be recomputed incrementally, making use of (28):

$$S_{m_{+i}} = S_m - \Sigma_{ii} (\beta \varphi_m^T e_0)^2, \tag{32}$$

$$Q_{m_{+i}} = Q_m - \omega_i (\beta \varphi_m^T e_0), \tag{33}$$

$$R_{m_{+i}} = R_m + \varphi_i^T e_2 (\Sigma_{ii} \beta \varphi_m^T e_0)^2 + (\varphi_m^T e_1)^2 - (\beta \varphi_m^T e_2)^2, \tag{34}$$

where $\omega_i = \Sigma_{ii} Q_i$ is the value of the i-th parameter after including the BF, and we define

$$e_0 \triangleq \varphi_i - \beta \Phi \Sigma \Phi^T \varphi_i, \tag{35}$$

$$e_1 \triangleq \beta \Sigma_{ii} \Phi \Sigma \Phi^T \varphi_i + \beta \Phi \Sigma \Sigma \Phi^T \varphi_i - \Sigma_{ii} \varphi_i, \tag{36}$$

$$e_2 \triangleq \Phi \Sigma \Sigma \Phi^T \varphi_i. \tag{37}$$

When adding/removing a BF, there is no need to explicitly compute the parameters w (steps 2 and 10 of the pseudocode) in order to compute the evidence.

5.2.2. Removal

We can rewrite (24) for the removal of BF ϕ_i:

$$\begin{aligned} \log P(t|\alpha, \beta, \mathscr{H}) = {} & \log P(t|\alpha, \beta, \mathscr{H}_{-i}) \\ & + \frac{1}{2}\left[\log \alpha - \log(\alpha + \phi_i^T C_{-i}^{-1} \phi_i) + \frac{(\phi_i^T C_{-i}^{-1} t)^2}{\alpha + \phi_i^T C_{-i}^{-1} \phi_i}\right] \\ = {} & \log P(t|\alpha, \beta, \mathscr{H}_{-i}) \\ & + \frac{1}{2}\left[\log \alpha - \log(\alpha + s_i) + \frac{q_i^2}{\alpha + s_i}\right], \end{aligned} \tag{38}$$

where $\mathscr{H}_{-i}$ is the model $\mathscr{H}$ with BF ϕ_i removed and we have defined $s_i \triangleq \phi_i^T C_{-i}^{-1} \phi_i$ and $q_i \triangleq \phi_i^T C_{-i}^{-1} t$. Applying (23) we can write:

$$s_i = \frac{\alpha S_i}{\alpha - S_i} \quad \text{and} \quad q_i = \frac{\alpha Q_i}{\alpha - S_i}. \tag{39}$$

After removing the BF i, the matrix Σ becomes

$$\Sigma_{-i} = \Sigma - \frac{1}{\Sigma_{ii}} \Sigma_i \Sigma_i^T. \tag{40}$$

Notice that this is an abuse of notation, since Σ_{-i} should have one dimension less than Σ. That is, the resulting i-th row and column (which are now zeros) should be removed. The trace can be written as

$$\operatorname{tr} \Sigma_{-i} = \operatorname{tr} \Sigma - \frac{1}{\Sigma_{ii}} \Sigma_i^T \Sigma_i. \tag{41}$$

Replacing s_i and q_i in (38) by (39) and extending to the evidence for the model, we get

$$\begin{aligned} 2(\log P(t|\mathscr{H}_{-i}) - \log P(t|\mathscr{H})) = {} & \frac{Q_i^2}{S_i - \alpha} - \log\left(1 - \frac{S_i}{\alpha}\right) \\ & + \log\frac{\gamma}{\gamma_{-i}} + \log\frac{N - \gamma}{N - \gamma_{-i}}, \end{aligned} \tag{42}$$

where

$$\gamma_{-i} = M - 1 - \alpha \operatorname{tr} \Sigma_{-i}. \tag{43}$$

Immediately after removing the BF i, and if α and β are not reestimated, making use of (40) we can write:

$$S_{m_{-i}} = S_m + \frac{1}{\Sigma_{ii}}(\beta \Sigma_i^T \Phi^T \varphi_m)^2, \tag{44}$$

$$Q_{m_{-i}} = Q_m + \frac{\omega_i}{\Sigma_{ii}}(\beta \Sigma_i^T \Phi^T \varphi_m), \tag{45}$$

$$R_{m_{-i}} = R_m - \frac{2}{\Sigma_{ii}} \varphi_m \Phi \Sigma \Sigma_i \Sigma_i^T \Phi \varphi_m + \frac{\Sigma_i^T \Sigma_i}{\Sigma_{ii}} \varphi_m^T \Phi \Sigma_i \Sigma_i^T \Phi^T \varphi_m, \tag{46}$$

where ω_i denotes the value of the i-th parameter before removing the BF.

5.2.3. *Reestimation of α and β and convergence*

For computational reasons, the selection of the best candidate can be performed with the previous values of α and β (i.e., steps 3 and 11 need not be implemented). However, after adding or removing a BF, α and β should be reestimated (steps 7 and 15).

In order to check *convergence* (step 17), Eqs. (10) can be used for computing trial values α_{new} and β_{new}. Since the evidence is approximated with a Gaussian using $\sigma^2_{\log \alpha}$ and $\sigma^2_{\log \beta}$ (see Eq. (17)), if

$$\frac{|\log \beta_{new} - \log \beta|}{\sigma_{\log \beta}} < \varepsilon \quad \text{and} \quad \frac{|\log \alpha_{new} - \log \alpha|}{\sigma_{\log \alpha}} < \varepsilon \tag{47}$$

then we consider convergence has been achieved. If not, α and β are set to α_{new} and β_{new} (step 18) and the trial is performed again unless condition (47) is satisfied. In the experiments, ε will be set to 0.1. A larger value of ε allows faster computations, while a smaller ε allows a better convergence of α and β. Sometimes convergence is achieved in the first iteration and the *Reestimate* function does not modify the values for α and β. When α and β are not modified, the selection of the next BF can be done more efficiently, as shown in Sections 5.2.1 and 5.2.2. We also found that, as the model grows, α and β tend to stabilize and more fast selections can be done. This is an advantage, since the cost of "slow" selections (immediately after α and β have been changed) is higher for larger models.

5.2.4. *Evaluation of the log-evidence*

After adding or removing a BF, we usually need to evaluate the log-evidence for the model (steps 26 and 30 in the pseudocode). If no recomputation of α and β is necessary, we can update the evidence for the model with the increments given by (30) and (42).

After recomputation of α and β, the evaluation of the log-evidence has to be done from scratch. In that case, evaluating (18) requires the computation of the determinant and inverse of matrix C, sized $N \times N$. However, using the Woodbury identity, both computations can be rewritten as a function of Σ, sized $M \times M$ [11]:

$$\log P(t|\mathscr{H}_i) \simeq -\frac{1}{2}\Bigg[N \log 2\pi - \log|\Sigma| - N \log \beta - M \log \alpha + \beta\|t - \Phi w\|^2 + \alpha\|w\|^2 - \log\left(\frac{2}{\gamma}\right) - \log\left(\frac{2}{N-\gamma}\right)\Bigg]. \tag{48}$$

5.2.5. *Initialization*

Initial values of α and β need to be set beforehand. In our experiments, β was set to $(0.1 \times \mathrm{var}(t))^{-1}$, following [27], and α was set to 0.001, assuming a broad prior distribution that leaves the weight values fairly unconstrained. When adding the first BF, the BF with the largest normalized projection onto the target vector $\|\varphi_i^T t\|^2/\|\varphi_i\|^2$ is selected. After its inclusion, α and β are reestimated, so their initial values are not directly used for any selection of BF. However, the reestimation equations of α and β always depend on previous values and therefore the selection of BFs might vary by changing the initialization.

5.2.6. *Numerical inaccuracies*

The presented methods need the computation of Σ (7)—the inverse of the Hessian matrix—that may become ill-conditioned. This may be the case when a small noise level is assumed (i.e., high β), and there exists near co-linearity between basis vectors ϕ_i. This would lead to numerical inaccuracies due to machine precision. When such inaccuracies arise, inversion techniques like SVD or LU decomposition could help, though they will not always solve the problem. In that case, trying a different set of candidate BFs would be a possible workaround.

5.2.7. *Computational cost and memory requirements in practice*

The cost of selecting the best candidate BF from a set of P candidates is imposed by the computation of all the R_m, Q_m and S_m which, assuming that the candidates φ_i are stored in memory, can be achieved in $O(P \times N \times M)$.

When removing a BF, only the values of R_m, Q_m and S_m corresponding to those BFs present in the model are needed. The cost of removing a BF would then be $O(M^2 \times N)$. However, in that case the other R_m, Q_m and S_m could not be recomputed incrementally. In our implementation we chose to compute all the R_m, Q_m and S_m.

After adding or removing a BF, α and β should be reestimated until convergence. After their reestimation Σ has to be computed, requiring a cost in $O(M^3)$. When the current values of α and β are valid no reestimation is needed. In these cases, Σ, R_m, S_m and Q_m can be recomputed incrementally and the cost of selecting a BF is reduced to $O(P \times N)$. In practice, memory requirements are dominated by the $P \times N$ design matrix for the whole dictionary. In the experiments a dictionary of $P = N$ BFs will be used.

5.3. *An example: the PTA(l,r) family of algorithms*

We illustrate the use of these operations with the pseudocode for the PTA(l, r) family of algorithms:

```
PTA ($l$, $r$, initial model $\mathscr{H}$, set of candidate BFs $\{\varphi_i\}$)
22.   set $\alpha, \beta$ to some sensible value
23.   while not StoppingCriterion($\mathscr{H}$) do
24.     for $l$ times do
25.       $(\alpha, \beta, \mathscr{H})$ = AddBestBasisFunction($\mathscr{H}, \alpha, \beta, \{\varphi_i\}$)
26.       compute the evidence for $\mathscr{H}$
27.     end for
28.     for $r$ times do
29.       $(\alpha, \beta, \mathscr{H})$ = RemoveWorstBasisFunction($\mathscr{H}, \alpha, \beta$)
30.       compute the evidence for $\mathscr{H}$
31.     end for
32.   end while
33.   set $\mathscr{H}$ the model that maximizes the evidence in the previous loop
34.   return ($\mathscr{H}$)
end PTA
```

6. An experimental comparison

6.1. *A toy problem*

We first introduce an artificial data set for illustrative purposes only, by creating a data set with 200 one-dimensional uniformly distributed input data. A target function with 20 BFs $\phi_i = \exp(-0.5(x-c_i)^2)$ was constructed, where the centers of the BFs were randomly chosen from the input data. The 20 parameters of the model were sampled from a Gaussian distribution with inverse variance $\alpha = 0.4$. The targets are generated by adding Gaussian noise with variance $\sigma^2 = 0.05$.

We explored PTA(1,0), PTA(2,1), SFFS and the RVM to model the data. The dictionary of BFs contained 200 radial BFs (i.e., RBFs) centered at the input data with the same width as that generating the model (this way the true model is within the dictionary). The four methods found models with 23, 15, 9 and 8 BFs, respectively. The generated functions and the selected BFs are shown in Fig. 1. We can see that different search strategies achieve comparable accuracy (also comparable to RVM) with a different number of BFs. Interestingly, the true model was never found, and only PTA(1,0) required more BFs than the true model.

6.2. Full experimental comparison

The aim of the following experiments is to *contrast* the described methods regarding generalization performance, model size and computational effort (measured by mean number of added/removed BFs and pure CPU training time). The following SSGEs were used: PTA(1,0), PTA(2,1), SFFS and Oscil(5). For the latter, the initial model is found by PTA(1,0), as suggested elsewhere [25]. We then compared the SSGEs to OLS, the SVM, the RVM and to a model with *all* the candidate BFs included (dubbed ABF), in which the hyper-parameters are adapted to maximize the marginal likelihood. The software LIBSVM [2] was used for the SVM, modified to deal with multiple RBF widths. We used our own implementation of the SSGEs, ABF, OLS and the RVM (for which the fast algorithm described in [27] was implemented). In the first part of the experiments, some data families of moderate size from the DELVE archive are used to contrast the methods. In the second part, a new family of data sets of increasing size is used to test the scalability of the methods.

6.2.1. Candidate BFs and the ABF model

We used N candidate *radial* BFs (RBF) coincident with the training set input vectors c_i:

$$\varphi_i(x) = \exp\left(-\sum_{d=1}^{D} \frac{(x_d - c_{id})^2}{r_d^2}\right),$$

where D is the dimension of the input vectors. In order to choose the RBF widths r_d, a model with all the candidate BFs included ($M = N$) was considered and conjugate gradients was applied to maximize the marginal likelihood. The derivatives of the marginal likelihood with respect to the RBF widths are similar to the those of the η-RVM [26]:

$$\frac{\partial \mathscr{L}}{\partial r_d} = \sum_{n=1}^{N}\sum_{m=1}^{M} \frac{\partial \mathscr{L}}{\partial \phi_{nm}} \frac{\partial \phi_{nm}}{\partial r_d} = \sum_{n=1}^{N}\sum_{m=1}^{M} D_{nm} \frac{\partial \phi_{nm}}{\partial r_d}, \tag{49}$$

where $D = \beta[(t - \Phi w)w^T - \Phi\Sigma]$ and $\phi_{nm} = \phi_m(x_n; r)$. The RBF widths were jointly optimized with α and β. The model obtained from this first stage is the one labeled ABF. Since RBF widths are fixed during SSGE training, there is no need to marginalize them out when computing the evidence for the models.

6.2.2. Additional considerations

A stopping criterion for the SSGEs is advisable since the highest evidence is usually achieved with a rather small subset of BFs (Fig. 2) and much computational effort can be avoided. In the experiments, a stopping criterion was applied as follows: suppose the current model has m BFs and the model with the highest evidence so far has m_h BFs, then if $m_h + k < m$ the process is stopped and the returned model is that with m_h BFs. A value of k too small can make the training stop too early, while a k too high will lead to wasteful computation. In this work we set k dynamically to $max(15, 0.3 \times m_h)$, rounded to the closest integer.

In order to select the number of BFs for OLS, 5-fold cross-validation (5-CV) and the previous stopping criterion were used. Instead of evaluating the models with the evidence, the sum-of-squares error (on the validation set) was used. Appropriate values for the SVM parameters ε and C were also found with 5-CV.

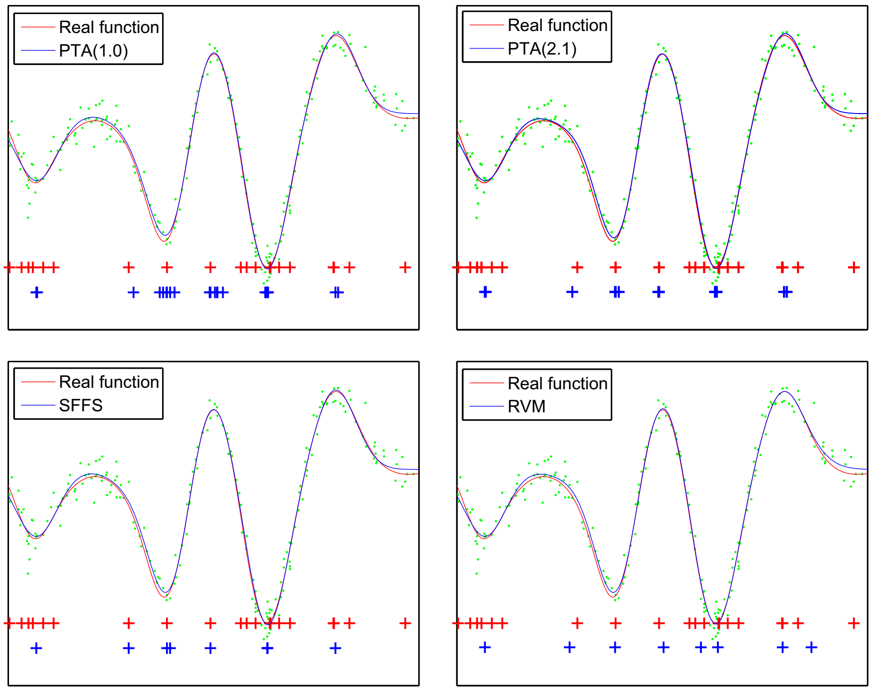


Fig. 1. A toy problem. Top row of crosses: locations of actual BFs. Bottom row: selected BFs. (A) PTA(1,0). (B) PTA(2,1). (C) SFFS. (D) RVM.

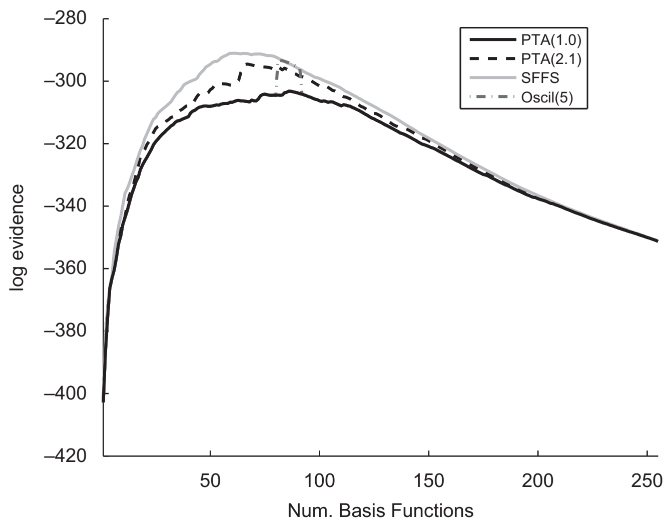


Fig. 2. The best log-evidence obtained by each SSGE method for each model size, measured in number of BFs, for a *kin*-8*nh* task instance. SFFS found the model with the highest evidence. PTA(2,1) and Oscil(5) also found models with higher evidence than PTA(1,0).

6.2.3. *The DELVE environment*

DELVE [21]—data for evaluating learning in valid experiments—contains a collection of families of data sets and an environment to assess the performance of supervised learning methods, allowing for statistically valid comparisons of different methods.

In the *first part* of the experiments, we worked with the *pumadyn*-8, *bank*-8 and *kin*-8 data sets from the DELVE archive. Four versions (*prototasks*) for each data set are provided: one *f* airly linear with *h*igh noise (*fh*), one *f* airly linear with *m*oderate noise (*fm*), one *n*on-linear with *h*igh noise (*nh*) and one *n*on-linear with *m*oderate noise (*nm*).

Each prototask consists of different tasks, where each task specifies the size of the training set. For the *pumadyn*-8, *bank*-8 and *kin*-8 data sets, five tasks are available, for training set sizes 64, 128, 256, 512 and 1024. Finally, each task has several task instances, corresponding to a particular training set and test set for that task, to which the training methods are applied. The particular results of the methods on several task-instances are used to estimate their expected performance on the task. In our case, all the tasks contain eight task-instances, except the 1024 task, that contains four task-instances. The comparisons between different methods are based on the sum-of-squares error on the test sets. Each test set (one for each task instance) consists of 512 examples, except for the tasks with 1024 training data, where the test set consists of 1024 examples. A total of 60 DELVE tasks were run for each method.

In the *second part* of the experiments, the Hwang family of data sets (DELVE version) was used. These data sets give the values of five different test functions over a common two-dimensional domain. The values are provided both without noise and with Gaussian noise added. With the purpose of assessing scalability of the methods, training and test set sizes were set to 1024, 2048, 4096 and 8192, amounting for $5 \times 2 \times 4 = 40$ different tasks. In all cases, target values were translated to $t_i = t'_i - mean(t'_i)$, where t'_i are the targets of the training sets. The input vectors were linearly scaled to $[-1, +1]$.

6.2.4. *Summary of experimental results*

The obtained DELVE diagrams for the first part of the experiments are summarized in Table 2, showing the number of tasks for which a method performed better than another, as follows: a t-test was performed on the test sets averages. Then we considered that a method performed (significantly) better than another if the p-value was lower than 0.05. Each entry shows the number of tasks where the column method is better than the row method; for example, PTA(1,0) performed better than Oscil(5) in five tasks and worse in ten tasks (Table 3).

In addition, Table 4 presents the mean number of BFs of the models found with the different algorithms on the tasks with 1024 training data, for ease of comparison. In a glance, and among the SSGEs, SFFS found the solution with the lowest number of BFs and the highest evidence. The SSGEs and the RVM then had comparable accuracy with a similar number of BFs. In general, models with higher sparsity performed slightly worse. In particular, ABF performed best, followed by Oscil(5) and PTA(1,0), PTA(2,1) and the SVM, the RVM, SFFS and finally OLS. Focusing on the evidence-driven methods, Table 6 presents the number of added/removed BFs of the different methods for the tasks with 1024 training data. The number of steps of SFFS and RVM varies much from one task to another. For those tasks requiring more BFs (i.e., difficult tasks), the number of steps is much larger. When the number of steps and BFs was high,

Table 2
Summary of the results for the *pumadyn*-8, *bank*-8 and *kin*-8 families of data sets.

–	PTA(1,0)	PTA(2,1)	SFFS	Oscil(5)	RVM	ABF	SVM	OLS
PTA(1,0)	–	0	0	10	3	10	6	0
PTA(2,1)	6	–	1	5	2	11	6	0
SFFS	16	10	–	14	4	14	13	0
Oscil(5)	5	0	0	–	1	8	5	0
RVM	13	5	3	13	–	17	7	0
ABF	4	2	1	4	2	–	6	0
SVM	6	3	1	7	2	6	–	0
OLS	23	18	9	22	15	20	14	-

Each cell shows the number of tasks where the column method performed better than the row method (indicated by a p-value lower than 0.05).

Table 3
Summary of the results for the *Hwang* tasks (8192 training data).

–	PTA(1,0)	PTA(2,1)	SFFS	Oscil(5)	RVM	SVM
PTA(1,0)	–	0	0	4	0	4
PTA(2,1)	0	–	0	3	0	4
SFFS	1	0	–	2	0	2
Oscil(5)	1	0	0	–	0	1
RVM	4	4	3	4	–	4
SVM	1	1	2	4	0	–

Each cell shows the number of tasks where the column method performed better than the row method (indicated by a p-value lower than 0.05).

Table 4
Mean number of BFs of the resulting models for all tasks with 1024 training data.

Method	*pumadyn*-8				*kin*-8				*bank*-8			
	fh	fm	nh	nm	fh	fm	nh	nm	fh	fm	nh	nm
PTA(1,0)	39	80	59	68	48	71	120	264	204	52	86	221
PTA(2,1)	11	59	24	34	39	46	103	203	191	35	61	142
SFFS	2.5	4.5	8.5	13	10	22	76	159	123	27	42	100
Oscil(5)	39	80	59	68	48	71	120	264	204	52	86	221
RVM	3.5	6.5	9.0	13	9.2	24	90	185	130	32	43	115
SVM	593	726	668	608	708	667	682	836	801	763	698	854
OLS	11	14	15	26	18	35	49	169	164	23	51	156

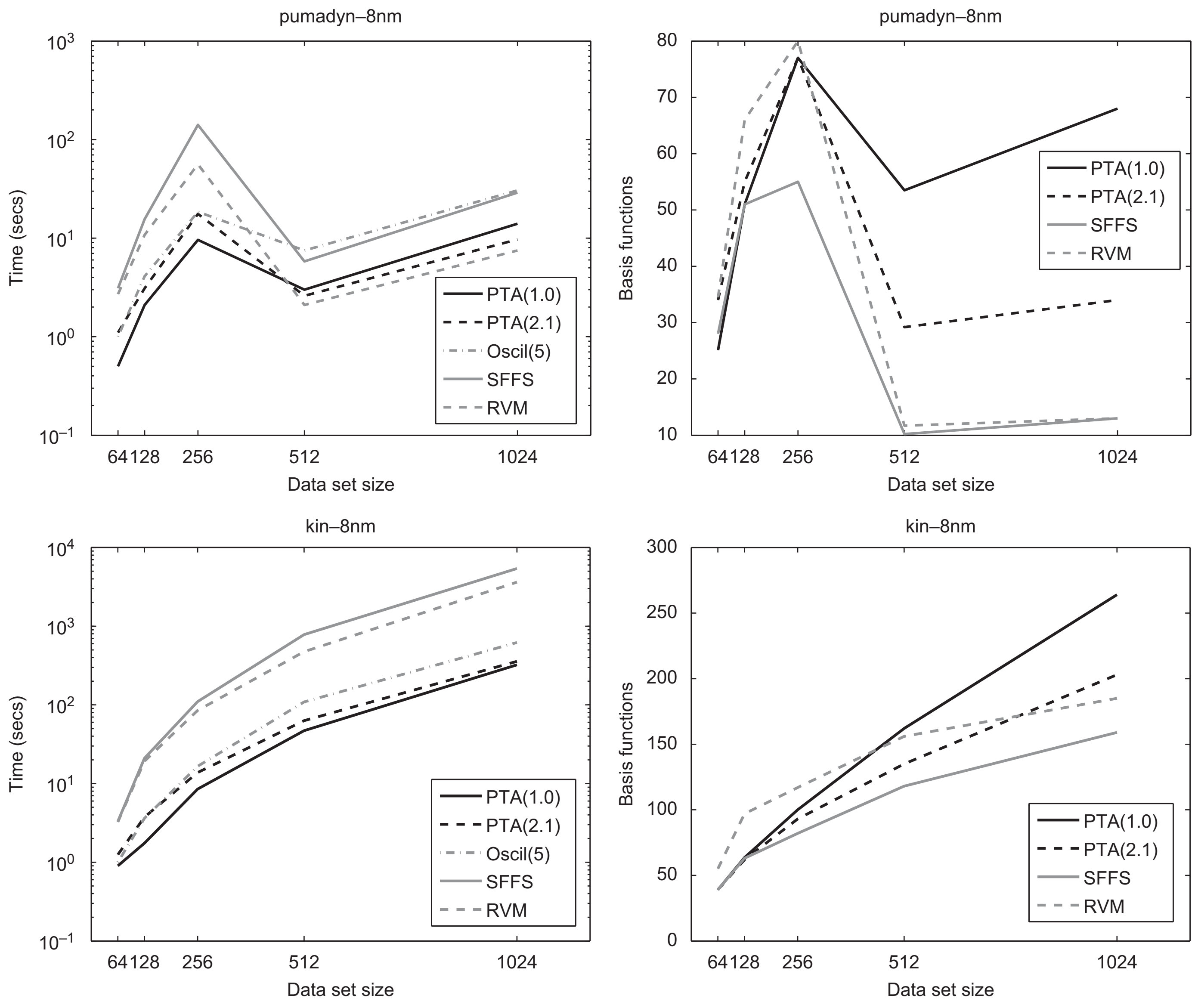


Fig. 3. Training time for all the SSGE methods and the RVM (left) and number of final BFs of the models (right), illustrated on the *pumadyn*-8*nm* and *kin*-8*nm*. These precise values were obtained on a Pentium IV running at 2.80 GHz.

PTA(1,0), PTA(2,1) and Oscil(5) usually required significantly less training time than the RVM and SFFS (Fig. 3). When the number of BFs of the model is high, SFFS and the RVM require a high number of iterations and the training time is higher.

The Hwang problems represent a different source of experimental results. Given its comparatively poor results, OLS was not considered in these experiments. The first thing to note is that ABF cannot be run for training set sizes greater than a couple of thousands, because of excessive training time (over 100 h without finishing). The obtained DELVE diagrams for the second part of the experiments are summarized in Table 3, focusing on the biggest data sets (with 8192 training data). Very noteworthy is the good behavior of Oscil(5) and the SVM. Table 5 presents the mean number of final BFs of the models found as well as the mean number of added/removed BFs (Table 6) (re-estimation iterations in the case of the RVM) again on the tasks with 8192 training data, for ease of comparison. Fig. 4 shows average training times (in seconds) and numbers of BFs across all Hwang tasks as a function of training set size.

Table 5
Mean number of BFs (#BF) and mean number of added/removed BFs (#BFa/r) (re-estimation iterations in the case of the RVM) in the training stage for all the Hwang-8192 tasks.

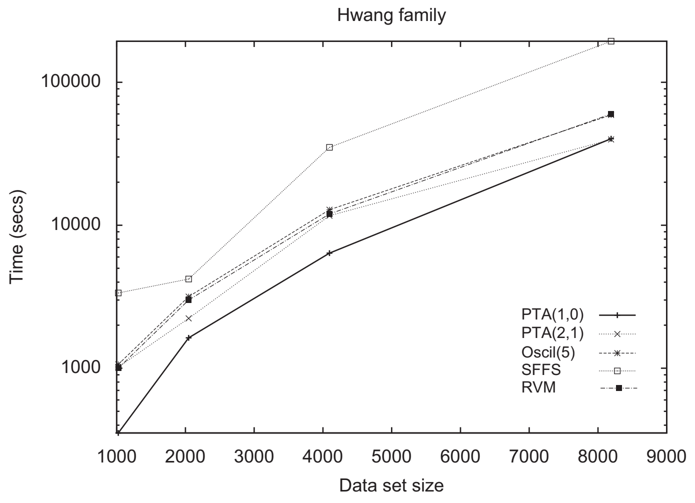
	PTA(1,0)	PTA(2,1)	SFFS	Oscil(5)	RVM	SVM
#BF	42.1	41.7	34.5	42.1	19.2	3407
#BFa/r	57.5	161.1	127.4	247.2	1234.1	–

7. Discussion

From Tables 2 and 3 it can be inferred that ABF, SSGE, the RVM and the SVM obtain in many cases better generalization than OLS. This can be explained by the fact that these methods explicitly consider noise and smoothness in the computation of the parameters, whereas the only measure in OLS to tackle the noise is to control the number of BFs.

Table 6
Mean number of added plus removed BFs in the training stage for all tasks with 1024 training data (re-estimation iterations in the case of the RVM).

Method	*pumadyn*-8				*kin*-8				*bank*-8			
	fh	fm	nh	nm	fh	fm	nh	nm	fh	fm	nh	nm
PTA(1,0)	56	107	79	88	65	92	155	343	244	94	126	261
PTA(2,1)	78	238	118	148	162	183	403	791	692	225	302	545
SFFS	132	344	190	361	330	381	2051	5892	4216	874	1528	3334
Oscil(5)	205	308	256	250	224	311	433	738	529	228	403	552
RVM	180	196	406	230	115	118	6236	13753	302	106	7225	7066



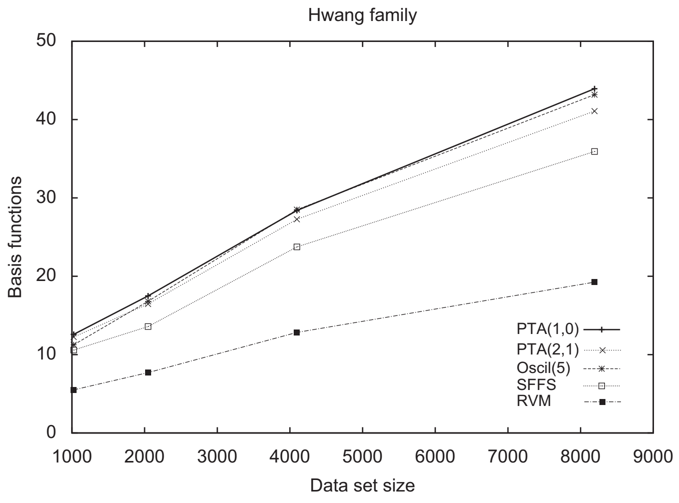


Fig. 4. Training time for all the SSGE methods and the RVM (left) and numbers of final BFs of the models (right), across all Hwang tasks as a function of training set size. These precise values were obtained on a Pentium IV running at 2.80 GHz.

Basis expansion models can produce overconfident predictive variances because the limited number of BFs restricts the prior over functions. This restriction is also responsible for the higher evidence obtained by simpler models, for which this prior is more limited—a model with M BFs only considers M linearly independent functions. Since the evidence is normalized on the space of data sets t (see Fig. 5), the models with the most restrictive priors can have higher evidence than the rest. This effect manifests in our experiments in the ABF method, which displays very good generalization performance (when applicable). The evidence for this model is rather low (Fig. 2) because it is plausible under a wide range of data sets (Fig. 5). The prior over functions for this model is fairly unconstrained (it considers $M = N$ linearly independent functions), suggesting that, contrary to the assumption in Section 3.3, the prior $P(\mathscr{H}_{ABF})$ should be higher than for simpler models, and consequently $P(\mathscr{H}_{ABF}|t)$ should also be high. Therefore, when no computational or memory restrictions are present, and for moderate training sizes, we recommend the use of ABF. If there are, but model size is not an issue, the choice should be the SVM. Otherwise, Oscil(5) represents a suitable trade-off between model size, training time and generalization error and shall be the preferred SSGE. Table 7 presents the methods ordered from top to bottom for the three preferences individually. It is advisable to remind that these issues can be controlled in the SSGEs by selecting the search strategy and that ABF can only cope with problems of moderate size.

On the other hand, the model with the highest evidence (found by SFFS) did not obtain the best generalization rate. Therefore, it could be argued that the evidence is not a good measure to optimize. Under the assumption of all the models having the same prior $P(\mathscr{H}_i)$, the evidence *is* a good measure to optimize, since $P(\mathscr{H}_i|t) \propto P(\mathscr{H}_i)P(t|\mathscr{H}_i)$. However, this assumption is usually inaccurate and the evidence shows a preference for simpler models. In other words, it satisfies a tradeoff between model simplicity and accuracy, which is why the models with the highest evidence, as found by SFFS, did not have the best performance, but were instead much simpler. This is an appealing property for many practical applications, in which time, memory or model size

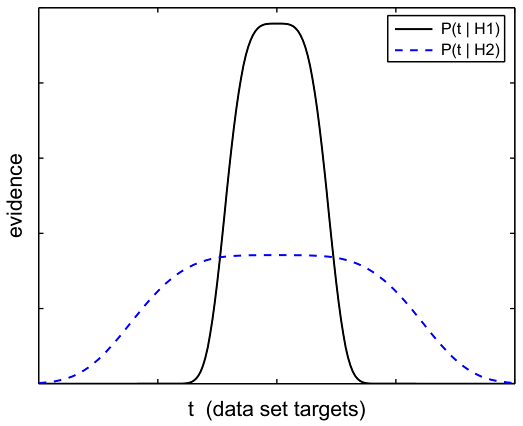


Fig. 5. The horizontal axis represents the space of possible data sets. Model $H1$ has a very low number of BFs and, as opposed to $H2$, it is plausible only under a reduced range of data sets. In other words, the prior over functions is more restrictive for $H1$. The vertical axis shows the evidence for each model. Since the evidence is normalized on t, under certain range of data sets the evidence for $H1$ is much higher than for $H2$.

Table 7
The methods ordered for each preference.

Training time	Model size	Generalization error
SVM	SFFS, RVM	ABF
PTA(1,0)	SFFS, RVM	Oscil(5), SVM
PTA(2,1)	PTA(2,1)	Oscil(5), SVM
Oscil(5)	PTA(1,0), Oscil(5)	PTA(1,0)
ABF	PTA(1,0), Oscil(5)	PTA(2,1)
RVM	SVM	RVM
SFFS	ABF	SFFS

can be critical; in other situations, only pure generalization ability is important. In this sense, Oscil(5) was the most accurate method overall, besides ABF (when applicable). We attribute this fact to the number of BFs being fixed beforehand to a high value—the one obtained with PTA(1,0)—so the simplicity of the model was constrained (i.e., the prior over functions was unconstrained).

8. Conclusions and final remarks

The regression problem has been tackled following an approximate Bayesian treatment as a search guided by the evidence (SSGE methods). In the experiments, some SSGEs—as Oscil(5)—produced compact models very competitive with implicit methods such as the RVM or the SVM. More sophisticated search strategies found more compact models than simpler ones at the expense of some accuracy. It has been found that sparsity usually comes at the expense of some accuracy; therefore, high sparsity is not always desirable. The choice of the proper search strategy will fulfill a tradeoff between computational effort, model size and generalization rate, for which Oscil(5) stands out as a suitable compromise. In this vein, the choice of the parameter in Oscil(5) could be explored to further reduce computational effort.

It has also been reported that a model with all the candidate BFs performed better than the SSGEs and the RVM for moderate training set sizes. This suggests that the assumption of all models having the same prior $P(\mathscr{H}_i)$ is inaccurate. An alternative would then be to approximate the ABF model with a subset of $M<N$ parameters that maximize the posterior (6). The selection of this subset could also be performed with a search process.

Some other approximate Bayesian approaches in the literature such as the RVM or sparse GP approximations are based on counter-intuitive assumptions or on a "convenience prior" that favors sparse solutions [20,18], leading to degenerate Gaussian processes. Since they restrict the prior over functions, the evidence has similar properties than under linear basis expansion models (see Fig. 5): it is biased towards simpler models (with more restricted priors) at the expense of some accuracy. It is conjectured that using complex search strategies to maximize the evidence in sparse GP approximations [17] would produce similar results to those obtained in this work. An interesting discussion about the effect of the prior on the size of the model can be found in [19].

Acknowledgments

This work has been supported by the Ministerio de Educación y Ciencia, under project CGL2004-04702-C02-02 and Grant BES-2003-0200.

References

[1] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, New York, 1995.
[2] C.C. Chang, C.J. Lin, LIBSVM: a library for support vector machines, Software available at ⟨http://www.csie.ntu.edu.tw/~cjlin/libsvm⟩, 2001.
[3] S. Chen, E.S. Chang, K. Alkadhimi, Regularised orthogonal least squares algorithm for constructing radial basis function networks, International Journal of Control 64 (5) (1996) 829–837.
[4] S. Chen, C.F.N. Cowan, P.M. Grant, Orthogonal least squares learning algorithm for radial basis function networks, IEEE Transactions on Neural Networks 2 (2) (1991) 302–309.
[5] S.S. Chen, S.L. Donoho, M.A. Saunders, Atomic decomposition by basis pursuit, SIAM Journal on Scientific Computing 20 (1) (1999) 33–61.
[6] J.H. Friedman, W. Stuetzle, Projection pursuit density estimation, Journal of the American Statistical Association 79 (1984) 599–608.
[7] A. Jain, D. Zongker, Feature selection: evaluation, application, and small sample performance, IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (2) (1997) 153–158.
[8] S. Keerthi, O. Chapelle, D. DeCoste, Building support vector machines with reduced classifier complexity, Journal of Machine Learning Research 8 (2006) 1–22.
[9] J. Kittler, Feature selection and extraction, in: T.Y. Young, K.S. Fu (Eds.), Handbook of Pattern Recognition and Image Processing, Academic Press, New York, 1986.
[10] D.J.C. MacKay, A practical Bayesian framework for backpropagation networks, Neural Computation 4 (3) (1992) 448–472.
[11] D.J.C. MacKay, Bayesian interpolation, Neural Computation 4 (3) (1992) 415–447.
[12] D.J.C. MacKay, Comparison of approximate methods for handling hyperparameters, Neural Computation 11 (5) (1999) 1035–1068.
[13] S.G. Mallat, Z. Zhang, Matching pursuits with time–frequency dictionaries, IEEE Transactions on Signal Processing 41 (12) (1993) 3397–3415.
[14] M.J.L. Orr, Regularisation in the selection of radial basis function centers, Neural Computation 7 (3) (1995) 606–623.
[15] M.J.L. Orr, An EM algorithm for regularised radial basis function networks, in: International Conference on Neural Networks and the Brain, 1998.
[16] P. Pudil, J. Novovičová, J. Kittler, Floating search methods in feature selection, Pattern Recognition Letters 15 (11) (1994) 1119–1125.
[17] J. Quiñonero, Learning with uncertainty—Gaussian processes and relevance vector machines, Ph.D. Thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2004.
[18] J. Quiñonero, C.E. Rasmussen, A unifying view of sparse approximate Gaussian process regression, Journal of Machine Learning Research 6 (2005) 1935–1959.
[19] C. Rasmussen, Z. Ghahramani, Occam's razor, in: Advances in Neural Information Processing Systems, vol. 13, 2001, pp. 294–300.
[20] C.E. Rasmussen, J. Quiñonero, Healing the relevance vector machine through augmentation, in: S. Wrobel, L. De Raedt (Eds.), Proceedings of the 22nd International Conference on Machine Learning, 2005, pp. 689–696.
[21] C.E. Rasmussen, R.M. Neal, G.E. Hinton, D. van Camp, Z. Ghahramani, M. Revow, R. Kustra, R. Tibshirani, The DELVE Manual ⟨www.cs.toronto.edu/~delve/⟩, 1996.
[22] E. Romero, R. Alquézar, A sequential algorithm for feed-forward neural networks with optimal coefficients and interacting frequencies, Neurocomputing 69 (13–15) (2006) 1540–1552.
[23] A.J. Smola, P.L. Bartlett, Sparse greedy Gaussian process regression, in: Advances in Neural Information Processing Systems, vol. 13, 2001, pp. 619–625.
[24] A.J. Smola, B. Schölkopf, On a kernel-based method for pattern recognition, regression, approximation, and operator inversion, Algorithmica 22 (1/2) (1998) 211–231.
[25] P. Somol, P. Pudil, Oscillating search algorithms for feature selection, in: Proceedings of the 15th International Conference on Pattern Recognition, 2000, pp. 2406–2409.
[26] M. Tipping, Sparse Bayesian learning and the relevance vector machine, Journal of Machine Learning Research 1 (2001) 211–244.
[27] M. Tipping, A. Faul, Fast marginal likelihood maximisation for sparse Bayesian models, in: Ninth International Workshop on Artificial Intelligence and Statistics, 2003.
[28] V. Vapnik, The Nature of Statistical Learning Theory, Springer, Berlin, 1995.
[29] P. Vincent, Y. Bengio, Kernel matching pursuit, Machine Learning 48 (1–3) (2002) 165–187.

Ignacio Barrio received the B.Sc. degree in Computer Science from the Universitat Politècnica de Catalunya (UPC) in 2000. He received the M.Sc. degree in Artificial Intelligence from the UPC. His research interests include Neural Networks and Bayesian techniques. He died while he was working towards his Ph.D. degree in Computer Science.

Enrique Romero received the B.Sc. degree in Mathematics in 1989 from the Universitat Autònoma de Barcelona. In 1994, he received the B.Sc. in Computer Science from the Universitat Politècnica de Catalunya (UPC). In 1996, he joined the Software Department at the UPC, as an Assistant Professor. He received the M.Sc. degree in Artificial Intelligence and the Ph.D. degree in Computer Science from the UPC in 2000 and 2004, respectively. His research interests include neural networks, support vector machines and feature selection.

Lluís A. Belanche is an Associate Professor in the Software Department at the Universitat Politènica de Catalunya (UPC). He received a B.Sc. in Computer Science from the UPC in 1990 and an M.Sc. in Artificial Intelligence from the UPC in 1991. He joined the Computer Science Faculty shortly after, where he completed his doctoral dissertation in 2000. He has been doing research in neural networks and support vector machines for pattern recognition and function approximation, as well as in feature selection algorithms and their collective application to workable artificial learning systems.