SPARSEBAYES V1.1: A Baseline Matlab Implementation of "Sparse Bayesian" Model Estimation

Michael E. Tipping
mail@miketipping.com

February 26, 2009

1 About the SparseBayes V1.1 Software

This document briefly summarises "Version 1" of the *SparseBayes* software, designed to run within the MATLAB environment (see `www.mathworks.com`). The latest version of this software, along with other explanatory materials, should be available from:

`http://www.relevancevector.com`

This documentation refers specifically to Version 1.1 of *SparseBayes*. Version 1.1 is a baseline re-implementation of the V1.0 package (dating from 2002) that was originally freely downloadable from Microsoft Research[1].

Please note that *SparseBayes* is *not* intended to be a fully comprehensive library or toolbox. Instead, the software comprises a basic set of representative routines implementing the necessary core functionality for "sparse Bayesian" models, such as the "relevance vector machine". It was originally designed as both a useful demonstration and a base upon which interested users can build more practical code.

This version incorporates simple, "old-style", hyperparameter adaptation, as featured in [2, 3]. As such, it can be relatively slow for larger data sets and is effectively restricted to cases where the entire design (or kernel) matrix can fit in memory. A much more efficient implementation, originally introduced in [4], will be made available in early 2009, as *SparseBayes* V2.0.

[1]The original page at `www.research.microsoft.com/mlp/rvm` appears to be no longer available.

2 File Summary

A brief description of each file in the *SparseBayes* V1.1 distribution is given in the below table. More comprehensive information for most functions can be obtained via the standard MATLAB `help` command. Only the files described under "**Core Functionality**" below are "essential" — other files are for the purposes of diagnostics or demonstration.

	Core Functionality
`SB1_Estimate.m`	The core, general, "sparse Bayesian" hyperparameter re-estimation code for a model with arbitrary basis. Further details are given in Section 4 shortly.
`SB1_RVM.m`	Kernel-based "relevance vector machine" model specialisation of (wrapper around) the above.
`SB1_PosteriorMode.m`	Posterior-mode finding function for the sparse Bayes classification case, called by `SB1_Estimate`.
`SB1_KernelFunction.m`	Definition of several example kernel functions for use by `SB1_RVM`.
	Examples
`SB1_ExampleRegress.m`	An example script to demonstrate regression.
`SB1_ExampleClassify.m`	An example script to demonstrate classification.
	Diagnostics
`SB1_Diagnostic.m`	Flexible diagnostic output.
`getEnvironment.m`	Simple functions to manipulate 'global' settings (not essential, but useful for debugging across multiple files).
`setEnvironment.m`	
	Example Data
`synth.tr`	Ripley's synthetic classification training data from [1].
`synth.te`	Ripley's corresponding test data.
`sb1_manual.pdf`	This document!

3 Quick Start

Copy the package to a directory of choice, and (at the MATLAB prompt) type

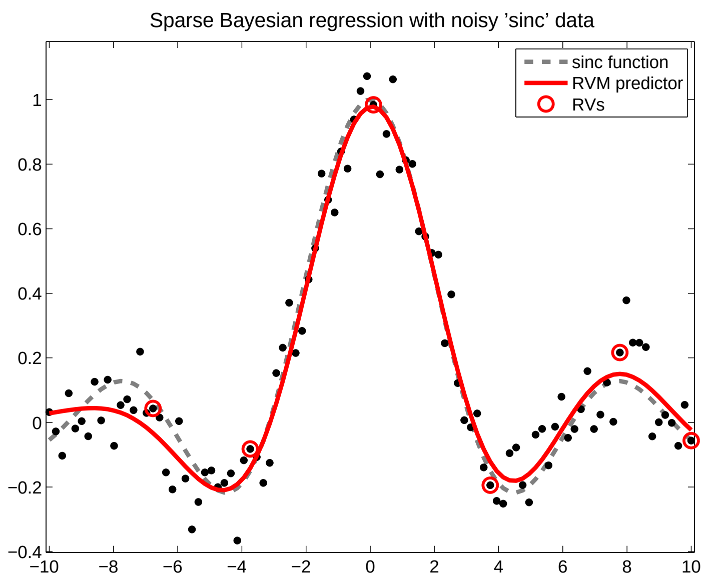
```
>> SB1_ExampleRegress
```

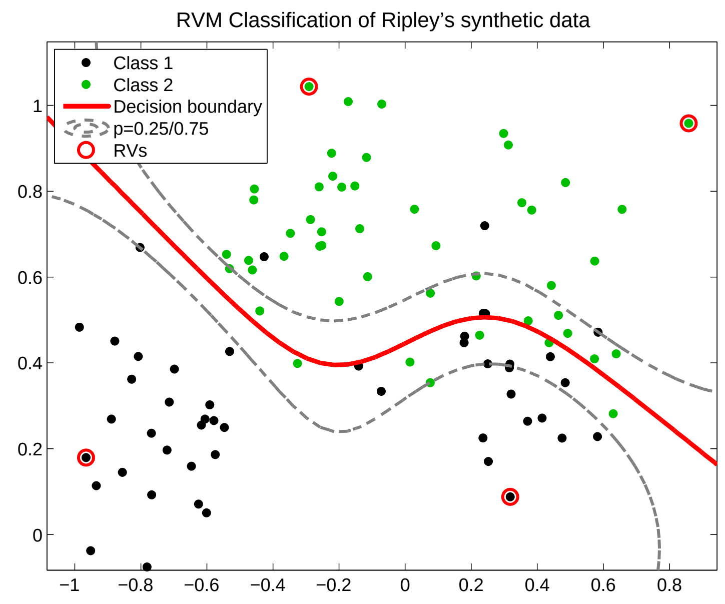
to demonstrate regression with a relevance vector machine model, or

```
>> SB1_ExampleClassify
```

to demonstrate classification. A glance at the content of these two files should hopefully illustrate how the *SparseBayes* software might be used.

If all goes to plan, the output should look as below.





4 The Core Function: `SB1_Estimate`

The ‘core’ functionality of *SparseBayes* is implemented within `SB1_Estimate.m`.

4.1 Basic usage

Assume you have a set of M basis functions evaluated over N data examples within an $N \times M$ matrix “`PHI`” with corresponding “targets” in the $N \times 1$ vector “`t`”. Then in the simplest case, the function can be called as follows:

```
>> [weights, used] = SB1_Estimate(PHI,t,alpha,beta,maxIts,monIts)
```

This will return a sparsified vector of `weights` corresponding to the subset of basis functions (columns of `PHI`) indexed by the vector `used`. Other results can also be returned (see the standard `help` text associated with the function).

Further parameters required by `SB1_Estimate` are:

- `alpha`: An initial common value for the hyperparameters. The algorithm should not be particularly sensitive to this choice (assuming it is non-extreme). The examples supplied use a simple heuristic setting of `alpha`$= 1/M^2$.
- `beta`: The initial value of the noise precision (inverse variance). See below for more detail on the interpretation of this value.
- `maxIts`: The maximum number of iterations to run the likelihood optimisation for.
- `monIts`: Optional parameter which specifies that progress information be output every `monIts` iterations.

4.2 Regression and classification

The function `SB1_Estimate` implements both regression and classification (strictly speaking, it incorporates Gaussian and Bernoulli likelihood functions). In this respect, note the following interpretation of the inverse noise variance `beta` (β) argument:

`beta`> 0	Regression with initial β initialised as specified, but re-estimated within the likelihood maximisation.
`beta`< 0	Regression with inverse noise fixed as $\lvert\beta\rvert$.
`beta`$= 0$	Used to specify classification: β is effectively redundant.

Note that for classification, `SB1_Estimate` will call `SB1_PosteriorMode` at each iteration to find the mode of the posterior distribution as required for the Laplace approximation [3].

4.3 Some implementation details

Hyperparameter updates. During each iteration, all hyperparameters α_i are updated according to the prescription given in [2, 3]. That is: $\alpha_i = \gamma_i/\mu_i^2$.

This is not the most effective approach to hyperparameter optimisation in a "sparse Bayesian" model, but it is the simplest and so makes sense in the context of this "base-line" implementation. The forthcoming *SparseBayes* V2.0 software will incorporate a significantly enhanced mechanism for hyperparameter updates.

Pruning. Basis functions are "pruned" (explicitly, and irreversibly, removed from the model) when the corresponding hyperparameter exceeds `ALPHA_MAX`, set initially to 10^9. This threshold can be reduced if it desired that the algorithm be more "aggressive".

Termination. The algorithm terminates when the largest change in the logarithm of any hyperparameter is less than `MIN_DELTA_LOGALPHA`. Initially set to 10^{-3}, this can be changed if desired.

References

[1] B. D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996.

[2] M. E. Tipping. The Relevance Vector Machine. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 652–658. MIT Press, 2000.

[3] M. E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

[4] M. E. Tipping and A. C. Faul. Fast marginal likelihood maximisation for sparse Bayesian models. In C. M. Bishop and B. J. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics, Key West, FL, Jan 3-6*, 2003.