SPARSE GAUSSIAN PROCESSES: INFERENCE AND MODEL SELECTION

Lehel Csató, Manfred Opper

Neural Computing Research Group, Dept of Information Engineering,
Aston University, Birmingham, U.K.
{csatol,opperm}@aston.ac.uk

Abstract:
Gaussian Process (GP) inference is a probabilistic kernel method where the GP is treated as a latent function. The inference is carried out using the Bayesian online learning and its extension to the more general iterative approach which we call TAP/EP learning (short for TAP (Opper and Winther, 2001) and "expectation-propagation" (EP) (Minka, 2000)).
Sparsity is introduced in this context to make the TAP/EP method applicable to large datasets. We address the prohibitive scaling of the number of parameters by defining a subset of the training data that is used as the support the GP, thus the number of required parameters is independent of the training set, similar to the case of "Support–" or "Relevance–Vectors".
An advantage of the full probabilistic treatment is that allows the computation of the marginal data likelihood or evidence, leading to hyper-parameter estimation within the GP inference.
An EM algorithm to choose the hyper-parameters is proposed. The TAP/EP learning is the E-step and the M-step then updates the hyper-parameters. Due to the sparse E-step the resulting algorithm does not involve manipulation of large matrices. The presented algorithm is applicable to a wide variety of likelihood functions. We present results of applying the algorithm on classification and nonstandard regression problems applied to artificial and real datasets.

Keywords: kernel methods, Bayesian inference, sparse representation

1. INFERENCE WITH GAUSSIAN PROCESSES

Gaussian Processes (GPs) are probabilistic kernel methods which combine the flexibility provided by the generic kernel framework with the advantage of a full probabilistic treatment of the problem, e.g. besides the *most probable latent function* it also allows us to assess the *uncertainties* associated to those values.

To have probabilistic treatment, we encode the data $\mathscr{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^{N}$ using a likelihood function. For independent data we have the likelihood as

$$P(\mathscr{D}|\boldsymbol{f}) = \prod_{n=1}^{N} P(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{f}) = \prod_{n=1}^{N} P(\boldsymbol{y}_n|f_{\boldsymbol{x}_n}). \qquad (1)$$

The likelihood is conditioned on a *latent function* $\boldsymbol{f}$ which we model as a GP, ie. a random function characterised by a joint Gaussian distribution of the function values for any finite collection of inputs. Notice that the general conditioning of the likelihood on the *whole* random function $\boldsymbol{f}$ is simplified to dependence on a single random variable: the GP marginal at location $\boldsymbol{x}_n$ (we use $f_n \doteq f_{\boldsymbol{x}_n}$).

To perform Bayesian inference we need a *GP prior* $p_0(\boldsymbol{f})$ for the function $\boldsymbol{f}$. A GP is fully specified by its prior mean function $\langle f_{\boldsymbol{x}} \rangle_0$ and the prior covariance kernel $K_0(\boldsymbol{x}, \boldsymbol{x}')$. In the following we assume zero prior mean function, thus the choice of the covariance kernel fully encodes our class of functions. The *posterior process* is derived from Bayes' rule and is written as

$$p_{post}(\boldsymbol{f}) = \frac{1}{Z} p(\mathscr{D}|\boldsymbol{f}) p_o(\boldsymbol{f}) \qquad (2)$$

where $Z = P(\mathscr{D}) = \int d\boldsymbol{f}\ p(\mathscr{D}|\boldsymbol{f})p_o(\boldsymbol{f})$ is the normalising constant or the free energy.

The simple expression eq. (2) describing the posterior process can seldom be applied in practise: two fundamental problems need to be addressed. The first problem appears with non-Gaussian likelihoods which leads to non-Gaussian posterior processes. This implies that we need approximations resulting in a tractable posterior, a possible approximation is presented in this section. The second problem is the super-linear increase of the parameters with the size of the dataset, addressed in Section 2.

To have computational tractability we approximate the non-Gaussian posterior process with a Gaussian one by retaining only the posterior mean and covariance kernel functions of the non-tractable posterior. The posterior mean $\langle f_{\boldsymbol{x}}\rangle_{post}$ and covariance kernel $K_{post}(\boldsymbol{x},\boldsymbol{x}')$ functions are given by the following expressions (Csató and Opper, 2002):

$$\begin{aligned}\langle f_{\boldsymbol{x}}\rangle_{post} &= \langle f_{\boldsymbol{x}}\rangle_0 + \textstyle\sum_{n=1}^N \alpha_n K_0(\boldsymbol{x},\boldsymbol{x}_n)\\ K_{post}(\boldsymbol{x},\boldsymbol{x}') &= K_0(\boldsymbol{x},\boldsymbol{x}')\\ &\quad + \textstyle\sum_{m,n=1}^N K_0(\boldsymbol{x},\boldsymbol{x}_m)C_{mn}K_0(\boldsymbol{x}_n,\boldsymbol{x}')\end{aligned} \tag{3}$$

where the vector $\boldsymbol{\alpha} = [\alpha_1,\ldots,\alpha_N]^T$ and matrix $\boldsymbol{C} = \{C_{mn}\}_{m,n=1}^N$ are the scalar parameters given as:

$$\begin{aligned}\alpha_n &= \frac{\partial}{\partial\langle f_n\rangle_0}\ln Z\\ C_{mn} &= \frac{\partial^2}{\partial\langle f_m\rangle_0\partial\langle f_n\rangle_0}\ln Z\end{aligned} \tag{4}$$

where Z is the complete data likelihood or the normalising constant in eq. (2) and the derivatives are with respect to the prior mean $\langle f_n\rangle_0$ of the GP at the training points $\boldsymbol{x}_n$.

It is important that the *functional form* of the posterior approximation does not depend on the particular likelihood. The result for the posterior mean was also obtained in the Kimeldorf-Wahba representer theorem (Kimeldorf and Wahba, 1971). Additionally to that, we have a representation for a *full Gaussian* process by providing the covariance kernel of the posterior approximation. The estimation of both moments of the posterior allows the probabilistic treatment, leading to eg. Bayesian error bars for prediction.

The expression of the posterior moments merely states the *existence* of the parameters $(\boldsymbol{\alpha},\boldsymbol{C})$ since the free energy Z cannot be computed nor the derivatives can be taken. Therefore we have to consider approximations to find them. Exact result exists only for Gaussian likelihood, however efficient approximations are feasible for various other likelihood functions (Csató and Opper, 2002; Minka, 2000; Csató, 2002).

We are focusing on the *joint* approximation of the parameters for both the mean and the covariance kernel functions and mention that the approximation of the parameters of the mean $\boldsymbol{\alpha}$ alone for different likelihoods is an active research area in the kernel community, see eg. in (Schölkopf *et al.*, 1999).

The approximation we use is the extension of the Bayesian online learning, called the TAP/EP algorithm (Minka, 2000; Opper and Winther, 2001). The "expectation-propagation" (EP) algorithm proposed a sequential optimisation procedure which updates the GP coefficients $(\boldsymbol{\alpha},\boldsymbol{C})$ by considering a single likelihood term $P(\boldsymbol{y}_k|f_k)$ from eq. (1) at each iteration. At each step the approximated posterior, the result of the previous step, was viewed as prior in the current step. The update of the parameters were performed using $\ln Z_k$ with single likelihood and the current prior. If we denote the first and second derivative of $\ln Z_k$ with q_k and r_k respectively, then we immediately have from eq. (3) the following updates for the GP mean and covariance functions:

$$\begin{aligned}\langle f_{\boldsymbol{x}}\rangle_{new} &= \langle f_{\boldsymbol{x}}\rangle_{old} + q_k K_{old}(\boldsymbol{x},\boldsymbol{x}_k)\\ K_{new}(\boldsymbol{x},\boldsymbol{x}') &= K_{old}(\boldsymbol{x},\boldsymbol{x}') + r_k K_{old}(\boldsymbol{x},\boldsymbol{x}_k)K_{old}(\boldsymbol{x}_k,\boldsymbol{x}').\end{aligned} \tag{5}$$

In developing the EP algorithm Minka observed that at each individual step the ratio of the prior and the approximated posterior *defines* a Gaussian approximation to each likelihood term $P(\boldsymbol{y}_k|f_k)$, these approximations are however *dependent* on each other, the dependence comes from the use of priors, ie. the approximations to the other likelihood terms. These quadratic approximations allowed to go beyond the conventional online learning and perform iterative updates. In online learning each term in the likelihood can only be used once. In the iterative extension first a "subtraction" of the previous Gaussian approximation to the likelihood is performed, followed by the online learning of the current input (see (Minka, 2000; Opper and Winther, 2001) for details). An other important benefit of the EP algorithm is the approximation to the marginal data likelihood, ie. model evidence and the possibility to perform model selection, presented in Section 3.

An important limiting factor of the framework presented so far is that it cannot be applied to large datasets, the *quadratic* scaling of the parameters with the size of the data is far too prohibitive. Therefore a second approximation is performed, presented next.

2. OBTAINING SPARSE SOLUTION

The kernels we use might not be infinite-dimensional, this implies that often the *representation* of the approximating posterior is *redundant*. Several techniques to exploit the redundancy of the MAP solution, ie. the mean function from eq. (3) have been proposed (Wahba, 1990; Smola and Bartlett, 2001; Williams and Seeger, 2001). These approaches did not take into account the posterior covariance. Our aim in this section, similar in philosophy to the earlier work, is to describe the posterior GP using a *constant and small* number of parameters and to have this number independent from the size of the dataset.

We want to write the posterior mean and covariance considering a *small subset* of the training inputs, called *"basis vector set"* or $\mathscr{B}V$ set (similar to Support Vectors (Vapnik, 1995) or Relevance Vectors (Tipping, 2000)):

$$\begin{aligned}\langle f_{\boldsymbol{x}}\rangle^{*}_{post} &= \sum\nolimits_{n\in\mathscr{B}V}\alpha^{*}_{n}K_0(\boldsymbol{x},\boldsymbol{x}_n)\\ K^{*}_{post}(\boldsymbol{x},\boldsymbol{x}') &= K_0(\boldsymbol{x},\boldsymbol{x}')\\ &\quad+\sum\nolimits_{m,n\in\mathscr{B}V}K_0(\boldsymbol{x},\boldsymbol{x}_n)C^{*}_{nm}K_0(\boldsymbol{x}_m,\boldsymbol{x}')\end{aligned} \tag{6}$$

where we assume that $(\boldsymbol{\alpha}^{*},\boldsymbol{C}^{*})$ are chosen such that the new process is "as close as possible" to the Bayesian posterior whilst $|\mathscr{B}V| = d$ is constant. Since both the original, large GP given by $(\boldsymbol{\alpha},\boldsymbol{C})$ from eq. (3) and its approximation using only the elements from the $\mathscr{B}V$ set are Gaussians, their KL-distance (Cover and Thomas, 1991) is computable. We perform a minimisation to find $(\boldsymbol{\alpha}^{*},\boldsymbol{C}^{*})$ from the original $(\boldsymbol{\alpha},\boldsymbol{C})$ and also evaluate the KL-distance which gives an efficient criterion to select the elements of the $\mathscr{B}V$ set (Csató, 2002; Csató and Opper, 2002). It is important to mention that the computation of the KL-distance *does not* require the computation of $(\boldsymbol{\alpha}^{*},\boldsymbol{C}^{*})$, additionally to the "old" parameters it only requires the inverse of the kernel matrix. In implementation iteratively update the inverse kernel matrix, thus further reducing the computational cost.

An efficient algorithm is obtained by *combining* the TAP/EP iterations with the KL-optimal projections. In the TAP/EP step, when processing the selected input, the $\mathscr{B}V$ set is *increased*. After the TAP/EP update, in the *pruning phase*, we consider the following optimisation problem: find the parameters of a new GP containing *one less* $\mathscr{B}V$ and it is the closest possible in the KL-sense to the GP resulting from the TAP/EP update step. The resulting algorithm is efficient and avoids the inversion of large kernel matrices, a frequent problem in using kernel methods. The algorithm is a greedy one in choosing the elements of the $\mathscr{B}V$ set, i.e. at each time it only looks at the possibility of exchanging a single value.

The approximated posterior is used for prediction of the $\boldsymbol{y}$ at an unseen point $\boldsymbol{x}$. Applying again Bayes' rule, we have the *predictive distribution* of the outputs:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \int d\,f_{\boldsymbol{x}}\,P(\boldsymbol{y}|f_{\boldsymbol{x}})\,p_{post}(f_{\boldsymbol{x}}) \tag{7}$$

where $P(\boldsymbol{y}|f_{\boldsymbol{x}})$ contains the latent function at an unseen $\boldsymbol{x}$. The latent process is a GP, meaning that the integral in eq. (7) is one-dimensional and it is with respect to a Gaussian, which is often doable either exactly or approximately. A second observation is that a Gaussian approximation to the latent variables does not mean a Gaussian predictive distribution, this depends on the choice of the likelihood function.

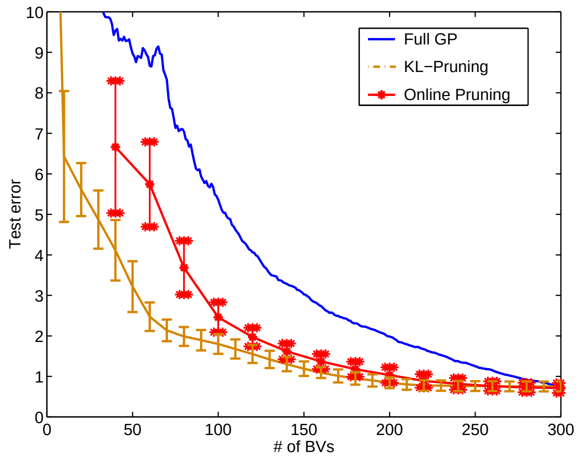

Fig. 1. Results for the Friedman dataset #1 using 300 training and 500 test data. The lines show the average test errors with error bars showing the empirical variance across different training sets.

We show the performance of the sparse algorithm for normal regression using the Friedman dataset in Fig. 1. The continuous line on the top shows the result of the batch GP regression when the inputs were only *partially* used, up to the size of the $\mathscr{B}V$ set. The bottom, dash-dotted line shows the performance when sparsification was applied *to the full GP solution* ie. all inputs were added to the $\mathscr{B}V$ set and then only the specific number of basis vectors has been retained. The middle, continuous line shows the results of combining the online learning with sparsity. We obtain a stable performance for $\mathscr{B}V$ set sizes exceeding 120. The continuous line, labelled "Full GP" shows us that the error for the sparse GP is never worse than the performance of a GP where we stop at the $\mathscr{B}V$ set size. The fact that the two bottom lines overlap means that the for $\mathscr{B}V$ sizes matching the "effective" dimension of the data the sparse GP is optimal.

We compared the sparse GP with the SVM and RVM methods. The SVM used 116 support vectors that lead to a test error of 2.92 and similar test error performance of 2.80 was obtained using 59 relevance vectors (Tipping, 2001), stating that our algorithm compares well to these other ones.

A few differences between these methods need to be mentioned. A first one is that both the SVM and the RVM start with a full solution and then obtain the sparse result without having control over the size of the result. On the contrary, in the sparse GP method we start with an empty $\mathscr{B}V$ set add training inputs up to the capacity of the machine used for the experiment or up to the dimension of the kernel. A second observation is that due to the storage of the covariance, the number of parameter is quadratic with respect to $|\mathscr{B}V|$ and this number is linear for the other two methods.

In this experiment the hyper-parameters were not selected automatically. The next section we introduce a method for hyper-parameter selection.

3. MODEL SELECTION

For model selection we use the expectation-maximisation (EM) algorithm (Bishop, 1995) which aims at maximising the complete data likelihood $P(\mathscr{D})$, the normalisation constant in eq. (2). We see that the set of model parameters can be divided in two groups: a first group related to the likelihood function $p(\boldsymbol{y}|f_{\boldsymbol{x}})$ and a second set that specifies the kernel function $K_0(\boldsymbol{x},\boldsymbol{x}')$.

To optimise **the likelihood parameters** we perform the optimisation of a lower bound on the model evidence using the EM-algorithm with the GP posterior having the "old" likelihood parameters and we want to find the new ones contained in $P(\mathscr{D}|\boldsymbol{f})$:

$$\ln p(\mathscr{D}) \geq \int d\boldsymbol{f}\; p_{post}(\boldsymbol{f}) \ln P(\mathscr{D}|\boldsymbol{f}) \qquad (8)$$

The GP inference is the *E-step* which gives the GP with a fixed set of hyper-parameters.

In the *M-step*, assuming that the posterior GP $p_{post}(\boldsymbol{f})$ is fixed, we optimise eq. (8) with respect to the likelihood parameters.The first term *does not* depend on the likelihood parameters, we only need to optimise the second expression.

Since the likelihood is factorising, the complete log-likelihood is rewritten as a sum. Each component of the sum depends on a single value of the latent GP, meaning that the integrals involved are one-dimensional, usually computable. After the update of the likelihood parameters we re-run the TAP/EP algorithm to find the new posterior $p_{post}(\boldsymbol{f})$ and alternate the steps until convergence.

To test the EM-algorithm, we applied the GP inference for non-symmetric and non-Gaussian additive noise, with the single data likelihood given by:

$$P(y|f_x,\lambda) = \begin{cases} \lambda \exp[-\lambda(y-f_x)] & \text{if } y > f_x \\ 0 & \text{otherwise} \end{cases} \qquad (9)$$

We can integrate the likelihood for a single data, we can apply thus the TAP/EP learning. This example shows the benefits of the online learning, i.e. that approximation can be performed even in cases where the MAP solution cannot be obtained. Fig. 2 shows the results for this toy problem (left-hand sub-figure) and an other GP regression where we used a symmetric Laplace noise. The non-symmetric likelihood provides a very good fit and infers the correct likelihood parameter, this is in contrast with symmetric noise assumption (we used the same asymptotic behaviour) where there is a constant bias and larger Bayesian error bars which are shown with thin dash-dotted lines.

Next we address the optimisation of **the kernel parameters**. The *E-step* is the same as before. In the *M-step* however, instead of upper-bounding the marginal data likelihood, we compute the gradient of the log-evidence with respect to a generic kernel parameter θ. Using matrix algebra and the parameters of the posterior GP $(\boldsymbol{\alpha},\boldsymbol{C})$, we have the following relation:

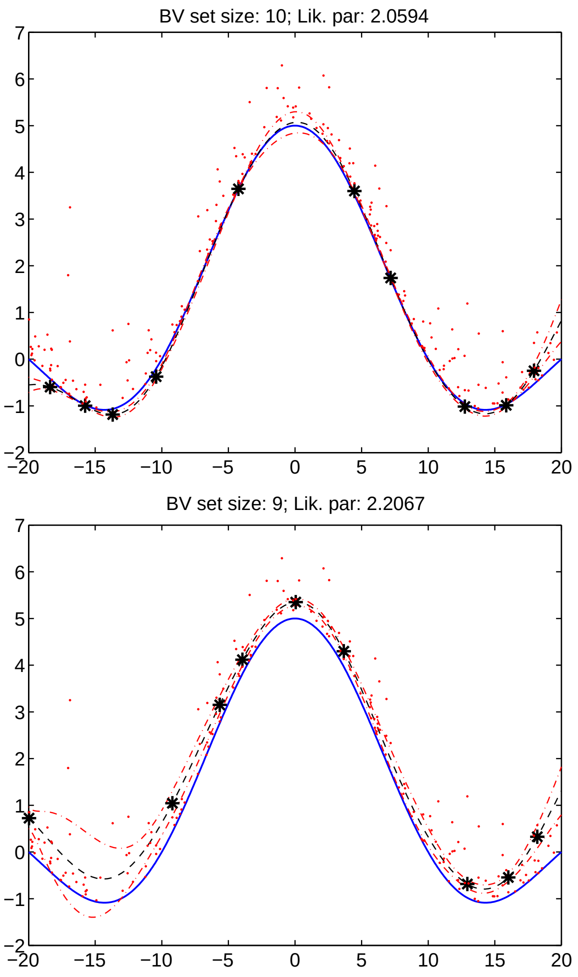

Fig. 2. Finding likelihood parameters of a robust and one-sided regression problem. The left sub-figure shows the result if the correct noise model is assumed and on the right we see the approximation for symmetrical noise. The noise parameter for generating the data was $\lambda = 2$ for both cases.

$$\begin{aligned} \frac{\partial \log \mathrm{P}(\mathscr{D})}{\partial \theta} &= \mathrm{tr}\left[\frac{\partial \log \mathrm{P}(\mathscr{D})}{\partial \boldsymbol{K}} \cdot \frac{\partial \boldsymbol{K}}{\partial \theta}\right] \\ \frac{\partial \log \mathrm{P}(\mathscr{D})}{\partial \boldsymbol{K}} &= -\frac{1}{2}\left(\boldsymbol{\alpha}\boldsymbol{\alpha}^T + \boldsymbol{C}\right) \end{aligned} \qquad (10)$$

where θ is a parameter of the kernel and $\partial \boldsymbol{K}$ is the matrix derivative. An exact update is not possible, we used conjugate gradient algorithm (SCG) to find the optimal kernel parameters. Here needs to be emphasised the sparse usefulness of the approximation: the size of the matrices involved is small, only the size of the $\mathscr{B}V$ set, making the proposed EM algorithm feasible.

In the experiments we used the RBF kernels defined as:

$$K(\boldsymbol{x},\boldsymbol{x}') = A\, \exp\left[-\frac{\sum_i \lambda_i (\boldsymbol{x}_i - \boldsymbol{x}'_i)^2}{2}\right] \qquad (11)$$

where the A is a scaling constant and λ_i are are the relevance parameters (ARD) specifying the importance of the i-th input dimension in predicting the output. We tested the performance of the method first for regression and the Friedman #1 dataset. This is a

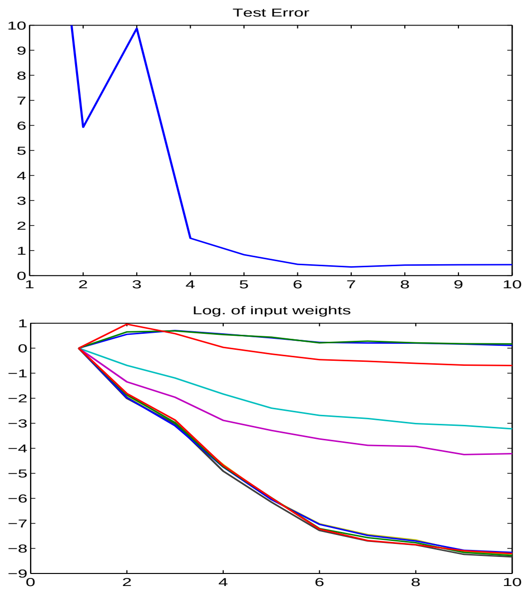

Fig. 3. Learning kernel parameters for noiseless Friedman data. On the top sub-figures the evolution of the average test error is shown. The bottom sub-figure shows the evolution of the $\ln \lambda_i$. The two rows show the results for no noise (top) and additive noise with $\sigma^2 = 1$ (bottom) respectively.

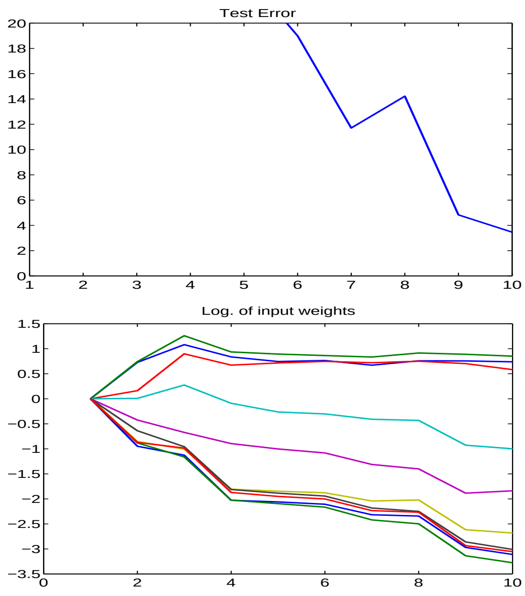

Fig. 4. Learning kernel parameters for noisy Friedman data. For explanation see Fig. 3.

benchmark for the relevance parameters since only 5 out of the 10-dimensional inputs were used to generate the output. Additionally to this source of uncertainty, the outputs are also corrupted with Gaussian noise. In Fig. 3 and 4 we see the test error and the resulting ARD kernel parameters for no noise and $\sigma^2 = 1$ respectively. As it can be seen, the separation of the irrelevant inputs from the relevant ones is clearer for the noiseless case.

Learning kernel parameters can also be done for classification tasks. We applied the EM algorithm for learning Crab data with 6-dimensional inputs (Csató

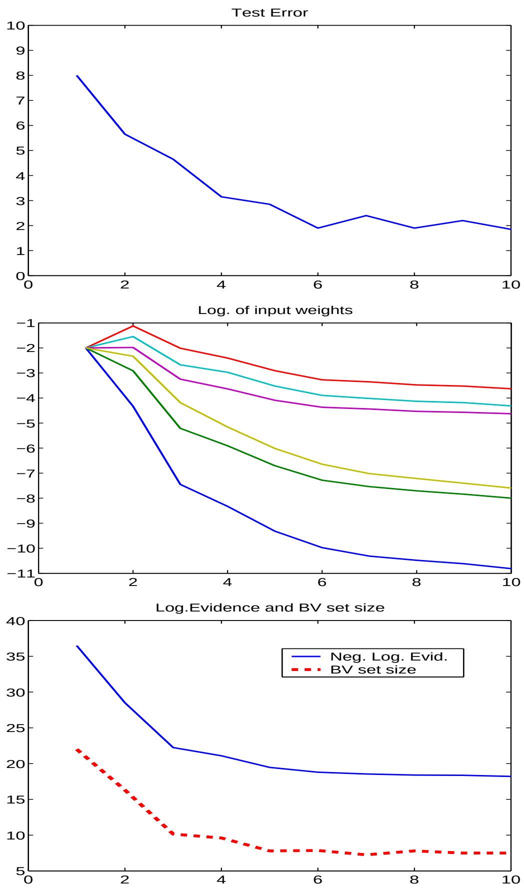

Fig. 5. Results for the Crab data. All figures show the averages over 20 random permutation of the training set. The figures show the test errors (top), the logarithm of the input weights (middle) and on the bottom, in the same plot, the negative log-evidence of the model and $\mathscr{B}V$ set size respectively. On the X-axis the number of the EM-steps is counted.

and Opper, 2002). This dataset has been widely used to assess various machine learning methods and most of the results confirmed that optimal performance is achieved using three of the six components. This is seen in the middle plot of Fig. 5 where three components are close together (approx -4 on a log-scale) with the rest several orders of magnitude smaller, practically removing the corresponding input dimension from the kernel function. It is important that simultaneously to this the the test error (upper plot of Fig. 5) attains a minimum value that outperforms other cited methods (see (Csató, 2002) for details).

It is interesting that, starting with a $\mathscr{B}V$ set size of 25, at the end of the iterations the size of the $\mathscr{B}V$ set is less than 8 on average and the log-evidence is also significantly smaller. We believe that this result encourages further investigation.

4. CONCLUSIONS

A method for probabilistic and sparse GP inference is presented. The inference is based on a representation that is independent of the likelihood function. Joined with the general representation, we present a method to derive a sparse solution for the problem.

The presented method is of selecting the reduced representation is constructive and flexible. It is applicable to problems with non-differentiable likelihood function as well as to classification and Gaussian regression.

Further research is aimed to test the method for other likelihoods and to try to establish conditions when we can expect good performance. Open question is still the performance of the sparse GP inference if *all* hyper-parameters are simultaneously optimised.

5. REFERENCES

Bishop, Christopher M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press. New York, N.Y.

Cover, Thomas M. and Joy A. Thomas (1991). *Elements of Information Theory*. John Wiley & Sons.

Csató, Lehel (2002). Gaussian Processes – Iterative Sparse Approximation. PhD thesis. Neural Computing Research Group. www.ncrg.-aston.ac.uk/Papers.

Csató, Lehel and Manfred Opper (2002). Sparse on-line Gaussian Processes. *Neural Computation* **14**(3), 641–669.

Kimeldorf, G.S. and G. Wahba (1971). Some results on Tchebycheffian spline functions. *J. Math. Anal. Applic.* **33**, 82–95.

Minka, Thomas P. (2000). Expectation Propagation for Approximate Bayesian Inference. PhD thesis. Dep. of El. Eng. & Comp. Sci.; MIT. vismod.www.media.mit.edu/∼tpminka.

Opper, Manfred and Ole Winther (2001). Adaptive and self-averaging TAP mean field theory for probabilistic modeling. *Physical Review E* **64**(056131), 1–14.

Schölkopf, Bernhard, Burges, Christopher J.C. and Smola, Alexander J., Eds.) (1999). *Advances in kernel methods (Support Vector Learning)*. The MIT Press.

Smola, Alexander J. and Peter Bartlett (2001). Sparse greedy Gaussian pocess regression. In: *NIPS* (Todd K. Leen, Thomas G. Diettrich and Volker Tresp, Eds.). Vol. 13. The MIT Press. pp. 619–625.

Tipping, Michael (2000). The Relevance Vector Machine. In: *NIPS* (Sara A. Solla, Todd K. Leen and Klaus-Robert Müller, Eds.). Vol. 12. The MIT Press. pp. 652–658.

Tipping, Michael E. (2001). Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research* **1**, 211–244.

Vapnik, Vladimir N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag. New York, NY.

Wahba, G. (1990). *Splines Models for Observational Data*. Series in Applied Mathematics, Vol. 59, SIAM. Philadelphia.

Williams, Christopher K. I. and Matthias Seeger (2001). Using the Nyström method to speed up kernel machines. In: *NIPS* (Todd K. Leen, Thomas G. Diettrich and Volker Tresp, Eds.). Vol. 13.