Journal of Machine Learning Research 11 (2010) 3011-3015 Submitted 8/10; Revised 9/10; Published 11/10

Gaussian Processes for Machine Learning (GPML) Toolbox

Carl Edward Rasmussen[*] CER54@CAM.AC.UK
Department of Engineering
University of Cambridge
Trumpington Street
Cambridge, CB2 1PZ, UK

Hannes Nickisch HN@TUE.MPG.DE
Max Planck Institute for Biological Cybernetics
Spemannstraße 38
72076 Tübingen, Germany

Editor: Sören Sonnenburg

Abstract

The GPML toolbox provides a wide range of functionality for Gaussian process (GP) inference and prediction. GPs are specified by mean and covariance functions; we offer a library of simple mean and covariance functions and mechanisms to compose more complex ones. Several likelihood functions are supported including Gaussian and heavy-tailed for regression as well as others suitable for classification. Finally, a range of inference methods is provided, including exact and variational inference, Expectation Propagation, and Laplace's method dealing with non-Gaussian likelihoods and FITC for dealing with large regression tasks.

Keywords: Gaussian processes, nonparametric Bayes, probabilistic regression and classification

Gaussian processes (GPs) (Rasmussen and Williams, 2006) have convenient properties for many modelling tasks in machine learning and statistics. They can be used to specify distributions over functions without having to commit to a specific functional form. Applications range from regression over classification to reinforcement learning, spatial models, survival and other time series[1] models. Predictions of GP models come with a natural confidence measure: predictive error-bars.

Although the implementation of the basic principles in the simplest case is straight forward, various complicating features are often desired in practice. For example, a GP is determined by a *mean function* and a *covariance function*, but these functions are mostly difficult to specify fully a priori, and typically they are given in terms of *hyperparameters*, that is, parameters which have to be inferred. Another source of difficulty is the *likelihood function*. For Gaussian likelihoods, inference is analytically tractable; however, in many tasks, Gaussian likelihoods are not appropriate, and approximate inference methods such as Expectation Propagation (EP) (Minka, 2001), Laplace's approximation (LA) (Williams and Barber, 1998) and variational bounds (VB) (Gibbs and MacKay, 2000) become necessary (Nickisch and Rasmussen, 2008). In case of large training data, approximations (Candela and Rasmussen, 2005) like FITC (Snelson and Ghahramani, 2006) are needed.

The GPML toolbox is designed to overcome these hurdles with its variety of mean, covariance and likelihood functions as well as inference methods, while being simple to use and easy to extend.

∗. Also at Max Planck Institute for Biological Cybernetics, Spemannstraße 38, 72076 Tübingen, Germany.

1. Note, that here we typically think of GPs with a more general index set than time.

1. Implementation

The GPML toolbox can be obtained from `http://gaussianprocess.org/gpml/code/matlab/` and also `http://mloss.org/software/view/263/` under the FreeBSD license. Based on simple interfaces for covariance, mean, likelihood functions as well as inference methods, we offer full compatibility to both Matlab 7.x[2] and GNU Octave 3.2.x.[3] Special attention has been given to properly disentangle covariance, likelihood and mean hyperparameters. Also, care has been taken to avoid numerical inaccuracies, for example, safe likelihood evaluations for extreme inputs and stable matrix operations. For example, the covariance matrix $\mathbf{K}$ can become numerically close to singular making its naive inversion numerically unsafe. We handle these situations in a principled way[4] such that Cholesky decompositions are computed of well-conditioned matrices only. As a result, our code shows a high level of robustness along the full spectrum of possible hyperparameters. The focus of the toolbox is on approximate inference using dense matrix algebra. We currently do not support covariance matrix approximation techniques to deal with large numbers of training examples n. Looking at the (growing) body of literature on sparse approximations, this knowledge is still somewhat in flux, and consensus on the best approaches has not yet been reached.

We provide stable and modular code checked by an exhaustive suite of test cases. A single function `gp.m` serves as main interface to the user—it can make inference and predictions and allows the mean, covariance and likelihood function as well as the inference methods to be specified freely.

Furthermore, `gp.m` enables convenient learning of the hyperparameters by maximising the log marginal likelihood $\ln Z$. One of the particularly appealing properties of GP models is that principled and practical approaches exist for learning the parameters of mean, covariance and likelihood functions. Good adaptation of such parameters can be essential to obtain both high quality predictions and insights into the properties of the data. The GPML toolbox is particularly flexible, including a large library of different covariance and mean functions, and flexible ways to combine these into more expressive, specialised functions. The user can choose between two gradient-based optimisers: one uses conjugate gradients (CG)[5] and the other one relies on a quasi-Newton scheme.[6] Computing the derivatives w.r.t. hyperparameters $\frac{\partial}{\partial \theta_i} \ln Z$ with `gp.m` does not need any extra programming effort; every inference method automatically collects the respective derivatives from the mean, covariance and likelihood functions and passes them to `gp.m`.

Our documentation comes in two pieces: a hypertext user documentation[7] `doc/index.html` with examples and code browsing and a technical documentation[8] `doc/manual.pdf` focusing on the interfaces and more technical issues. A casual user will use the hypertext document to quickly get his data analysed, however a power user will consult the pdf document once he wants to include his own mean, covariance, likelihood and inference routines or learn about implementation details.

2. Matlab is available from MathWorks, `http://www.mathworks.com/`.
3. Octave is available from the Free Software Foundation, `http://www.gnu.org/software/octave/`.
4. We do not consider the "blind" addition of a "small ridge" to $\mathbf{K}$ a principled way.
5. Carl Rasmussen's code is available at `http://www.kyb.tuebingen.mpg.de/bs/people/carl/code/minimize/`.
6. Peter Carbonetto's wrapper can be found at `http://www.cs.ubc.ca/~pcarbo/lbfgsb-for-matlab.html`.
7. Documentation can be found at `http://www.gaussianprocess.org/gpml/code/matlab/doc/index.html`.
8. Technical docs are available at `http://www.gaussianprocess.org/gpml/code/matlab/doc/manual.pdf`.

2. The GPML Toolbox

We illustrate the modular structure of the GPML toolbox by means of a simple code example. GPs are used to formalise and update knowledge about distributions over functions. A GP prior distribution on an unknown latent function $f \sim \mathcal{GP}(m_\phi(\mathbf{x}), k_\psi(\mathbf{x},\mathbf{x}'))$, consists of a mean function $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$, and a covariance function $k(\mathbf{x},\mathbf{x}) = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$, both of which typically contain hyperparameters ϕ and ψ, which we want to fit in the light of data. We generally assume independent observations, that is, input/output pairs $(\mathbf{x}_i, y_i)$ of f with joint likelihood $\mathbb{P}_\rho(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^n \mathbb{P}_\rho(y_i|f(\mathbf{x}_i))$ factorising over cases. Finally, after specification of the prior and fitting of the hyperparameters $\theta = \{\phi, \psi, \rho\}$, we wish to compute predictive distributions for test cases.

```
   % 1) SET UP THE GP: COVARIANCE; MEAN, LIKELIHOOD, INFERENCE METHOD
1  mf = {'meanSum',{'meanLinear',@meanConst}};   a = 2; b = 1;    % m(x) = a*x+b
2  cf = {'covSEiso'}; sf = 1; ell = 0.7;  % squared exponential covariance funct
3  lf = 'likLaplace'; sn = 0.2;        % assume Laplace noise with variance sn^2
4  hyp0.mean = [a;b]; hyp0.cov = log([ell;sf]); hyp0.lik  = log(sn); % hypers
5  inf = 'infEP';          % specify expectation propagation as inference method
   % 2) MINIMISE NEGATIVE LOG MARGINAL LIKELIHOOD nlZ wrt. hyp;  do 50 CG steps
6  Ncg = 50; [hyp, nlZ] = minimize(hyp0, 'gp', -Ncg, inf, mf, cf, lf, X, y);
   % 3) PREDICT AT UNKNOWN TEST INPUTS
7  [ymu, ys2] = gp(hyp, inf, mf, cf, lf, X, y, Xs);              % test input Xs
```

In **line 1**, we specify the mean $m_\phi(\mathbf{x}) = \mathbf{a}^\top\mathbf{x} + b$ of the GP with hyperparameters $\phi = \{\mathbf{a}, b\}$. First, the functional form of the mean function is given and its parameters are initialised. The desired mean function, happens not to exist in the library of mean functions; instead we have to make a *composite* mean function from *simple* constituents. This is done using a nested cell array containing the algebraic expression for $m(\mathbf{x})$: As the sum of a linear (`mean/meanLinear.m`) and a constant mean function (`mean/meanConst.m`) it is an affine function. In addition to linear and constant mean functions, the toolbox offers $m(\mathbf{x}) = 0$ and $m(\mathbf{x}) = 1$. These *simple* mean functions can be combined by *composite* mean functions to obtain sums (`mean/meanSum.m`) $m(\mathbf{x}) = \sum_j m_j(\mathbf{x})$, products $m(\mathbf{x}) = \prod_j m_j(\mathbf{x})$, scaled versions $m(\mathbf{x}) = \alpha m_0(\mathbf{x})$ and powers $m(\mathbf{x}) = m_0(\mathbf{x})^d$. This flexible mechanism is used for convenient specification of an extensible algebra of mean functions. Note that functions are referred to either as name strings `'meanConst'` or alternatively function handles `@meanConst`. The order of components of the hyperparameters ϕ is the same as in the specification of the cell array. Every mean function implements its evaluation $\mathbf{m} = m_\phi(\mathbf{X})$ and first derivative computation $\mathbf{m}_i = \frac{\partial}{\partial\phi_i} m_\phi(\mathbf{X})$ on a data set $\mathbf{X}$.

In the same spirit, the squared exponential covariance $k_\psi(\mathbf{x},\mathbf{x}') = \sigma_f{}^2 \exp(-\|\mathbf{x}-\mathbf{x}'\|^2/2\ell^2)$ (`cov/covSEiso.m`) with hyperparameters $\psi = \{\ln\ell, \ln\sigma_f\}$ is set up in **line 2**. Note, that the hyperparameters are represented by the logarithms, as these parameters are naturally positive. Many other *simple* covariance functions are contained in the toolbox. Among others, we offer linear, constant, Matérn, rational quadratic, polynomial, periodic, neural network and finite support covariance functions. *Composite* covariance functions allow for sums $k(\mathbf{x},\mathbf{x}') = \sum_j k_j(\mathbf{x},\mathbf{x}')$, products $k(\mathbf{x},\mathbf{x}') = \prod_j k_j(\mathbf{x},\mathbf{x}')$, positive scaling $k(\mathbf{x},\mathbf{x}') = \sigma_f^2 k_0(\mathbf{x},\mathbf{x}')$ and masking of components $k(\mathbf{x},\mathbf{x}') = k_0(\mathbf{x}_I,\mathbf{x}'_I)$ with $I \subseteq [1,2,..,D]$, $\mathbf{x} \in \mathbb{R}^D$. Again, the interface is simple since only the evaluation of the covariance matrix $\mathbf{K} = k_\psi(\mathbf{X})$ and its derivatives $\partial_i\mathbf{K} = \frac{\partial}{\partial\psi_i} k_\psi(\mathbf{X})$ on a data set $\mathbf{X}$ are required. Furthermore, we need cross terms $\mathbf{k}_* = k_\psi(\mathbf{X},\mathbf{x}_*)$ and $k_{**} = k_\psi(\mathbf{x}_*,\mathbf{x}_*)$ for prediction. There are no restrictions on the composition of both mean and covariance functions—any combination is allowed including nested composition.

The Laplace (`lik/likLaplace.m`) likelihood $\mathbb{P}_{\rho}(y|f) = \exp(-\sqrt{2}/\sigma_n|y-f|)/\sqrt{2}\sigma_n$ with hyperparameters $\boldsymbol{\rho} = \{\ln\sigma_n\}$ is specified in **line 3**. There are only *simple* likelihood functions: Gaussian, Sech-squared, Laplacian and Student's t for ordinary and sparse regression as well as the error and the logistic function for classification. Again, the same inference code is used for any likelihood function. Although the specification of likelihood functions is simple for the user, writing new likelihood functions is slightly more involved as different inference methods require access to different properties; for example, LA requires second derivatives and EP requires derivatives of moments.

All hyperparameters $\boldsymbol{\theta} = \{\phi, \psi, \boldsymbol{\rho}\}$ are stored in a struct `hyp.{mean,cov,lik}`, which is initialised in **line 4**; we select the approximate inference algorithm EP (`inf/infEP.m`) in **line 5**.

We optimise the hyperparameters $\boldsymbol{\theta} \equiv$ `hyp` by calling the CG optimiser (`util/minimize.m`) with initial value $\boldsymbol{\theta}_0 \equiv$ `hyp0` in **line 6** allowing at most $N = 50$ evaluations of the EP approximation to the marginal likelihood $Z_{EP}(\boldsymbol{\theta})$ as done by `gp.m`. Here, $\mathcal{D} = (\mathbf{X}, \mathbf{y}) \equiv$ `(X,y)` is the training data where $\mathbf{X} = \{\mathbf{x}_1, .., \mathbf{x}_n\}$ and $\mathbf{y} \in \mathbb{R}^n$. Under the hood, `gp.m` computes in every step a Gaussian posterior approximation and the derivatives $\frac{\partial}{\partial\boldsymbol{\theta}} \ln Z_{EP}(\boldsymbol{\theta})$ of the marginal likelihood by calling EP.

Predictions with optimised hyperparameters are done in **line 7**, where we call `gp.m` with the unseen test inputs $\mathbf{X}_* \equiv$ `Xs` as additional argument. As a result, we obtain the approximate marginal predictive mean $\mathbb{E}[\mathbb{P}(\mathbf{y}_*|\mathcal{D}, \mathbf{X}_*)] \equiv$ `ymu` and the predictive variance $\mathbb{V}[\mathbb{P}(\mathbf{y}_*|\mathcal{D}, \mathbf{X}_*)] \equiv$ `ys2`.

Likelihood \ Inference	Exact	FITC	EP	Laplace	VB	Type, Output Domain	Alternate Name
Gaussian	✓	✓	✓	✓	✓	regression, $\mathbb{R}$	
Sech-squared			✓	✓	✓	regression, $\mathbb{R}$	logistic distribution
Laplacian			✓		✓	regression, $\mathbb{R}$	double exponential
Student's t				✓	✓	regression, $\mathbb{R}$	
Error function			✓	✓	✓	classification, $\{\pm 1\}$	probit regression
Logistic function			✓	✓	✓	classification, $\{\pm 1\}$	logit regression

Table 1: Likelihood $\leftrightarrow$ inference compatibility in the GPML toolbox

Table 1 gives the legal likelihood/inference combinations. Exact inference and the FITC approximation support the Gaussian likelihood only. Variational Bayesian (VB) inference is applicable to all likelihoods. Expectation propagation (EP) for the Student's t likelihood is inherently unstable due to its non-log-concavity. The Laplace approximation (LA) for Laplace likelihoods is not sensible due to the non-differentiable peak of the Laplace likelihood. Special care has been taken for the non-convex optimisation problem imposed by the combination Student's t likelihood and LA.

If the number of training examples is larger than a few thousand, dense matrix computations become too slow. We provide the FITC approximation for regression with Gaussian likelihood where instead of the exact covariance matrix $\mathbf{K}$, a low-rank plus diagonal matrix $\tilde{\mathbf{K}} = \mathbf{Q} + \mathrm{diag}(\mathbf{K} - \mathbf{Q})$ where $\mathbf{Q} = \mathbf{K}_{\mathbf{u}}^{\top}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{u}}$ is used. The matrices $\mathbf{K}_{\mathbf{uu}}$ and $\mathbf{K}_{\mathbf{u}}$ contain covariances and cross-covariances of and between inducing inputs $\mathbf{u}^i$ and data points $\mathbf{x}^j$. Using `inf/infFITC.m` together with any covariance function wrapped into `cov/covFITC.m` makes the computations feasible for large n.

Acknowledgments

Thanks to Ed Snelson for assisting with the FITC approximation.

References

Joaquin Quiñonero Candela and Carl E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6(6):1935–1959, 2005.

Mark N. Gibbs and David J. C. MacKay. Variational Gaussian process classifiers. *IEEE Transactions on Neural Networks*, 11(6):1458–1464, 2000.

Thomas P. Minka. Expectation propagation for approximate Bayesian inference. In *UAI*, pages 362–369. Morgan Kaufmann, 2001.

Hannes Nickisch and Carl E. Rasmussen. Approximations for binary Gaussian process classification. *Journal of Machine Learning Research*, 9:2035–2078, 10 2008.

Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, MA, 2006.

Ed Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, 2006.

Christopher K. I. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(20):1342–1351, 1998.