Bayesian Modeling with Gaussian Processes using the MATLAB Toolbox GPstuff (v3.4)

Jarno Vanhatalo* JARNO.VANHATALO@HELSINKI.FI
Department of Environmental Sciences
University of Helsinki
P.O. Box 65, FI-00014 Helsinki, Finland

Jaakko Riihimäki JAAKKO.RIIHIMAKI@AALTO.FI
Jouni Hartikainen JOUNI.HARTIKAINEN@AALTO.FI
Pasi Jylänki PASI.JYLANKI@AALTO.FI
Ville Tolvanen VILLE.TOLVANEN@AALTO.FI
Aki Vehtari AKI.VEHTARI@AALTO.FI
Department of Biomedical Engineering and Computational Science
Aalto University School of Science
P.O. Box 12200, FI-00076 Aalto, Finland

Abstract

Gaussian processes (GP) are powerful tools for probabilistic modeling purposes. They can be used to define prior distributions over latent functions in hierarchical Bayesian models. The prior over functions is defined implicitly by the mean and covariance function, which determine the smoothness and variability of the function. The inference can then be conducted directly in the function space by evaluating or approximating the posterior process. Despite their attractive theoretical properties GPs provide practical challenges in their implementation. **GPstuff** is a versatile collection of computational tools for GP models. It has been implemented for Linux and Windows MATLAB and includes, among others, various inference methods, sparse approximations and tools for model assessment. In this work, we review these tools and demonstrate the use of **GPstuff** in several models.

Keywords: Gaussian process, Bayesian hierarchical model, nonparametric Bayes

1. Introduction

This work reviews a free open source toolbox **GPstuff** (version 3.3) which implements a collection of inference methods and tools for inference for various Gaussian process (GP) models. The toolbox is written for MATLAB (Mathworks, 2010) and it is available from `http://becs.aalto.fi/en/research/bayes/gpstuff/`.

GP is a stochastic process, which provides a powerful tool for probabilistic inference directly on distributions over functions (e.g. O'Hagan, 1978) and which has gained much attention in recent years (Rasmussen and Williams, 2006). In many practical GP models, the observations $\mathbf{y} = [y_1, ..., y_n]^{\mathrm{T}}$ related to inputs (covariates) $\mathbf{X} = \{\mathbf{x}_i = [x_{i,1}, ..., x_{i,d}]^{\mathrm{T}}\}_{i=1}^n$ are assumed to be conditionally independent given a latent function (or predictor) $f(\mathbf{x})$ so that the likelihood $p(\mathbf{y} \,|\, \mathbf{f}) = \prod_{i=1}^n p(y_i|f_i)$, where $\mathbf{f} = [f(\mathbf{x}_1), ..., f(\mathbf{x}_n)]^{\mathrm{T}}$, factorizes over cases. GP prior is set for the latent function after which the posterior $p(f|\, \mathbf{y}, \mathbf{X})$ is solved

*. Work done mainly while at the Department of Biomedical Engineering and Computational Science, Aalto University

and used for prediction. GP defines the prior over the latent function implicitly through the mean and covariance function, which determine, for example, the smoothness and variability of the latent function. Usually, the model hierarchy is extended to the third level by giving also prior for the parameters of the covariance function and the observation model.

The models in **GPstuff** follow this general model definition and can be summarized as:

$$\text{Observation model:} \qquad \mathbf{y} \,|\, \mathbf{f}, \phi \sim \prod_{i=1}^{n} p(y_i|f_i, \phi) \tag{1}$$

$$\text{GP prior:} \qquad f(\mathbf{x})|\theta \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'\,|\theta)\right) \tag{2}$$

$$\text{hyperprior:} \qquad \theta, \phi \sim p(\theta)p(\phi). \tag{3}$$

Here θ denotes the parameters in the covariance function $k(\mathbf{x}, \mathbf{x}'\,|\theta)$, and ϕ the parameters in the observation model. We will call the function value $f(\mathbf{x})$ at fixed $\mathbf{x}$ a latent variable and for brevity we sometimes denote all the covariance function and observation model parameters with $\vartheta = [\theta, \phi]$. For the simplicity of presentation the mean function is considered zero, $m(\mathbf{x}) \equiv 0$, throughout this paper. We will also denote both the observation model and likelihood by $p(\mathbf{y} \,|\, \mathbf{f}, \phi)$ and assume the reader is able to distinguish between these two from the context. The likelihood naming convention is used in the toolbox for both likelihood and observation model related functionalities which follows the naming convention used, for example, in **INLA** (Rue et al., 2009) and **GPML** (Rasmussen and Nickisch, 2010) software packages.

Many GP models such as general dynamic linear models and multi-output models, do not fulfill the assumptions (1)-(3). However, the current implementation of **GPstuff** relies on them and as will be seen, the class of models fulfilling these assumptions is rather rich. Even though the observations are assumed to be conditionally independent given the latent variables, the dependencies between the observations are incorporated into the model via the GP prior. At the moment, only independent priors on θ and ϕ are provided, as highlighted by equation (3), but general form $p(\theta, \phi)$ could be used as well.

Early examples of GP models can be found, for example, in time series analysis and filtering (Wiener, 1949) and in geostatistics (e.g. Matheron, 1973). GPs are still widely used in these fields and useful overviews are provided in (Cressie, 1993; Grewal and Andrews, 2001; Diggle and Ribeiro, 2007; Gelfand et al., 2010). O'Hagan (1978) was one of the firsts to consider GPs in a general probabilistic modeling context and provided a general theory of GP prediction. This general regression framework was later rediscovered as an alternative for neural network models (Williams and Rasmussen, 1996; Rasmussen, 1996) and extended for other problems than regression (Neal, 1997; Williams and Barber, 1998). This machine learning perspective is comprehensively summarized by Rasmussen and Williams (2006).

Nowadays, GPs are used, for example, in weather forecasting (Fuentes and Raftery, 2005; Berrocal et al., 2010), spatial statistics (Best et al., 2005; Kaufman et al., 2008; Banerjee et al., 2008), computational biology (Honkela et al., 2011), reinforcement learning (Deisenroth et al., 2009, 2011), healthcare applications (Stegle et al., 2008; Vanhatalo et al., 2010), industrial applications (Kalliomäki et al., 2005), computer model calibration and emulation (Kennedy and O'Hagan, 2001; Conti et al., 2009), prior elicitation (Moala and O'Hagan, 2010) and density estimation (Tokdar and Ghosh, 2007; Tokdar et al., 2010) to

name a few. Despite their attractive theoretical properties and wide range of applications, GPs provide practical challenges in implementation.

GPstuff provides several state-of-the-art inference algorithms and approximative methods that make the inference easier and faster for many practically interesting models. **GPstuff** is a modular toolbox which combines inference algorithms and model structures in an easily extensible format. It also provides various tools for model checking and comparison. These are essential in making model assessment and criticism an integral part of the data analysis. Many algorithms and models in **GPstuff** are proposed by others but reimplemented for **GPstuff**. In each case, we provide reference to the original work but the implementation of the algorithm in **GPstuff** is always unique. The basic implementation of two important algorithms, the Laplace approximation and expectation propagation algorithm (discussed in Section 3), follow the pseudocode from (Rasmussen and Williams, 2006). However, these are later generalized and improved as described in (Vanhatalo et al., 2009, 2010; Vanhatalo and Vehtari, 2010; Jylänki et al., 2011).

There are also many other toolboxes for GP modelling than **GPstuff** freely available. Perhaps the best known packages are nowadays the Gaussian processes for Machine Learning **GPML** toolbox (Rasmussen and Nickisch, 2010) and the flexible Bayesian modelling (**FBM**) toolbox by Radford Neal. A good overview of other packages can be obtained from the Gaussian processes website `http://www.gaussianprocess.org/` and the R Archive Network `http://cran.r-project.org/`. Other GP softwares have some overlap with **GPstuff** and some of them include models that are not implemented in **GPstuff**. The main advantages of **GPstuff** over other GP software are its versatile collection of models and computational tools as well as modularity which allows easy extensions.

Three earlier GP and Bayesian modelling packages have influenced our work. Structure of **GPstuff** is mostly in debt to the **Netlab** toolbox (Nabney, 2001), although it is far from being compatible. The **GPstuff** project was started in 2006 based on the **MCMCStuff**-toolbox (1998-2006) (`http://becs.aalto.fi/en/research/bayes/mcmcstuff/`). **MCMCStuff** for its part was based on **Netlab** and it was also influenced by the **FBM**. The **INLA** software package by Rue et al. (2009) has also motivated some of the technical details in the toolbox. In addition to these, some technical implementations of **GPstuff** rely on the sparse matrix toolbox **SuiteSparse** (Davis, 2005) (`http://www.cise.ufl.edu/research/sparse/SuiteSparse/`).

This work concentrates on discussing the essential theory behind the implementation of **GPstuff**. We explain important parts of the code, but the full code demonstrating the important features of the package (including also data creation and such), are included in the demonstration files `demo_*` to which we refer in the text.

2. Gaussian process models

2.1 Gaussian process prior

GP prior over function $f(\mathbf{x})$ implies that any set of function values $\mathbf{f}$, indexed by the input coordinates $\mathbf{X}$, have a multivariate Gaussian distribution

$$p(\mathbf{f}\,|\mathbf{X},\theta) = \mathrm{N}(\mathbf{f}\,|\mathbf{0},\mathbf{K}_{\mathrm{f,f}}), \tag{4}$$

where $\mathbf{K}_{\mathrm{f,f}}$ is the covariance matrix. Notice, that the distribution over functions will be denoted by $\mathcal{GP}(\cdot,\cdot)$, whereas the distribution over a finite set of latent variables will be denoted by $\mathrm{N}(\cdot,\cdot)$. The covariance matrix is constructed by a covariance function, $[\mathbf{K}_{\mathrm{f,f}}]_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j|\theta)$, which characterizes the correlation between different points in the process. Covariance function can be chosen freely as long as the covariance matrices produced are symmetric and positive semi-definite ($\mathbf{v}^{\mathrm{T}}\,\mathbf{K}_{\mathrm{f,f}}\,\mathbf{v} \geq 0, \forall \mathbf{v} \in \Re^n$). An example of a stationary covariance function is the squared exponential

$$k_{\mathrm{se}}(\mathbf{x}_i, \mathbf{x}_j\,|\theta) = \sigma_{\mathrm{se}}^2 \exp(-r^2/2), \tag{5}$$

where $r^2 = \sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2/l_k^2$ and $\theta = [\sigma_{\mathrm{se}}^2, l_1, ..., l_d]$. Here, σ_{se}^2 is the scaling parameter, and l_k is the length-scale, which governs how fast the correlation decreases as the distance increases in the direction k. Other common covariance functions are discussed, for example, by Diggle and Ribeiro (2007), Finkenstädt et al. (2007) and Rasmussen and Williams (2006) and the covariance functions in **GPstuff** are summarized in the appendix A.

Assume that we want to predict the values $\tilde{\mathbf{f}}$ at new input locations $\tilde{\mathbf{X}}$. The joint prior for latent variables $\mathbf{f}$ and $\tilde{\mathbf{f}}$ is

$$\begin{bmatrix}\mathbf{f}\\ \tilde{\mathbf{f}}\end{bmatrix} |\mathbf{X}, \tilde{\mathbf{X}}, \theta \sim \mathrm{N}\left(\mathbf{0}, \begin{bmatrix}\mathbf{K}_{\mathrm{f,f}} & \mathbf{K}_{\mathrm{f},\tilde{\mathrm{f}}}\\ \mathbf{K}_{\tilde{\mathrm{f}},\mathrm{f}} & \mathbf{K}_{\tilde{\mathrm{f}},\tilde{\mathrm{f}}}\end{bmatrix}\right), \tag{6}$$

where $\mathbf{K}_{\mathrm{f,f}} = k(\mathbf{X}, \mathbf{X}|\theta)$, $\mathbf{K}_{\mathrm{f},\tilde{\mathrm{f}}} = k(\mathbf{X}, \tilde{\mathbf{X}}|\theta)$ and $\mathbf{K}_{\tilde{\mathrm{f}},\tilde{\mathrm{f}}} = k(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}|\theta)$. Here, the covariance function $k(\cdot,\cdot)$ denotes also vector and matrix valued functions $k(\mathbf{x}, \mathbf{X}) : \Re^d \times \Re^{d\ \times n} \rightarrow \Re^{1\times n}$, and $k(\mathbf{X}, \mathbf{X}) : \Re^{d\times n} \times \Re^{d\times n} \rightarrow \Re^{n\times n}$. By definition of GP the marginal distribution of $\tilde{\mathbf{f}}$ is $p(\tilde{\mathbf{f}}|\tilde{\mathbf{X}}, \theta) = \mathrm{N}(\tilde{\mathbf{f}}|\mathbf{0}, \mathbf{K}_{\tilde{\mathrm{f}},\tilde{\mathrm{f}}})$ similar to (4). The conditional distribution of $\tilde{\mathbf{f}}$ given $\mathbf{f}$ is

$$\tilde{\mathbf{f}}|\,\mathbf{f}, \mathbf{X}, \tilde{\mathbf{X}}, \theta \sim \mathrm{N}(\mathbf{K}_{\tilde{\mathrm{f}},\mathrm{f}}\,\mathbf{K}_{\mathrm{f,f}}^{-1}\,\mathbf{f}, \mathbf{K}_{\tilde{\mathrm{f}},\tilde{\mathrm{f}}} - \mathbf{K}_{\tilde{\mathrm{f}},\mathrm{f}}\,\mathbf{K}_{\mathrm{f,f}}^{-1}\,\mathbf{K}_{\mathrm{f},\tilde{\mathrm{f}}}), \tag{7}$$

where the mean and covariance of the conditional distribution are functions of input vector $\tilde{\mathbf{x}}$ and $\mathbf{X}$ serves as a fixed parameter. Thus, the above distribution generalizes to GP with mean function $m(\tilde{\mathbf{x}}|\theta) = k(\tilde{\mathbf{x}}, \mathbf{X}|\theta)\,\mathbf{K}_{\mathrm{f,f}}^{-1}\,\mathbf{f}$ and covariance $k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}'|\theta) = k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}'|\theta) - k(\tilde{\mathbf{x}}, \mathbf{X}|\theta)\,\mathbf{K}_{\mathrm{f,f}}^{-1}\,k(\mathbf{X}, \tilde{\mathbf{x}}'|\theta)$, which define the conditional distribution of the latent function $f(\tilde{\mathbf{x}})$.

2.2 Conditioning on the observations

The cornerstone of the Bayesian inference is Bayes' theorem by which the conditional probability of the latent function and parameters after observing the data can be solved. This posterior distribution contains all information about the latent function and parameters conveyed from the data $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ by the model. Most of the time we cannot solve the posterior but need to approximate it. **GPstuff** is built so that the first inference step is to form (either analytically or approximately) the conditional posterior of the latent variables given the parameters

$$p(\mathbf{f}\,|\mathcal{D}, \theta, \phi) = \frac{p(\mathbf{y}\,|\,\mathbf{f}, \phi)p(\mathbf{f}\,|, \mathbf{X}, \theta)}{\int p(\mathbf{y}\,|\,\mathbf{f}, \phi)p(\mathbf{f}\,|\mathbf{X}, \theta)d\,\mathbf{f}}, \tag{8}$$

which is discussed in the section 3. After this, we can (approximately) marginalize over the parameters to obtain the marginal posterior distribution for the latent variables

$$p(\mathbf{f}\,|\mathcal{D}) = \int p(\mathbf{f}\,|\mathcal{D},\theta,\phi)p(\theta,\phi|\mathcal{D})d\theta d\phi \tag{9}$$

treated in the section 5. The posterior predictive distributions can be obtained similarly by first evaluating the conditional posterior predictive distribution, for example $p(\tilde{f}|\mathcal{D},\theta,\phi,\tilde{\mathbf{x}})$, and then marginalizing over the parameters. The joint predictive distribution $p(\tilde{\mathbf{y}}|\mathcal{D},\theta,\phi,\tilde{\mathbf{x}})$ would require integration over possibly high dimensional distribution $p(\tilde{\mathbf{f}}|\mathcal{D},\theta,\phi,\tilde{\mathbf{x}})$. However, usually we are interested only on the marginal predictive distribution for each $\tilde{y}_i$ separately which requires only one dimensional integrals

$$p(\tilde{y}_i|\mathcal{D},\tilde{\mathbf{x}}_i,\theta,\phi) = \int p(\tilde{y}_i|\tilde{f}_i,\phi)p(\tilde{f}_i|\mathcal{D},\tilde{\mathbf{x}}_i,\theta,\phi)d\tilde{f}_i. \tag{10}$$

If the parameters are considered fixed, GP's marginalization and conditionalization properties can be exploited in the prediction. Given the conditional posterior distribution $p(\mathbf{f}\,|\mathcal{D},\theta,\phi)$, which in general is not Gaussian, we can evaluate the posterior predictive mean from the conditional mean $\mathrm{E}_{\tilde{\mathbf{f}}|\,\mathbf{f},\theta,\phi}[f(\tilde{\mathbf{x}})] = k(\tilde{\mathbf{x}},\mathbf{X})\,\mathbf{K}_{\mathrm{f,f}}^{-1}\,\mathbf{f}$ (see equation (7) and the text below it). Since this holds for any $\tilde{\mathbf{f}}$, we obtain a parametric posterior mean function

$$m_p(\tilde{\mathbf{x}}|\mathcal{D},\theta,\phi) = \int \mathrm{E}_{\tilde{\mathbf{f}}|\,\mathbf{f},\theta,\phi}[f(\tilde{\mathbf{x}})]p(\mathbf{f}\,|\mathcal{D},\theta,\phi)d\,\mathbf{f} = k(\tilde{\mathbf{x}},\mathbf{X}|\theta)\,\mathbf{K}_{\mathrm{f,f}}^{-1}\,\mathrm{E}_{\mathbf{f}\,|\mathcal{D},\theta,\phi}[\mathbf{f}]. \tag{11}$$

The posterior predictive covariance between any set of latent variables, $\tilde{\mathbf{f}}$ is

$$\mathrm{Cov}_{\tilde{\mathbf{f}}|\mathcal{D},\theta,\phi}[\tilde{\mathbf{f}}] = \mathrm{E}_{\mathbf{f}\,|\mathcal{D},\theta,\phi}\left[\mathrm{Cov}_{\tilde{\mathbf{f}}|\,\mathbf{f}}[\tilde{\mathbf{f}}]\right] + \mathrm{Cov}_{\mathbf{f}\,|\mathcal{D},\theta,\phi}\left[\mathrm{E}_{\tilde{\mathbf{f}}|\,\mathbf{f}}[\tilde{\mathbf{f}}]\right], \tag{12}$$

where the first term simplifies to the conditional covariance in equation (7) and the second term to $k(\tilde{\mathbf{x}},\mathbf{X})\,\mathbf{K}_{\mathrm{f,f}}^{-1}\,\mathrm{Cov}_{\mathbf{f}\,|\mathcal{D},\theta,\phi}[\mathbf{f}]\,\mathbf{K}_{\mathrm{f,f}}^{-1}\,k(\mathbf{X},\tilde{\mathbf{x}}')$. The posterior covariance function is then

$$k_p(\tilde{\mathbf{x}},\tilde{\mathbf{x}}'|\mathcal{D},\theta,\phi) = k(\tilde{\mathbf{x}},\tilde{\mathbf{x}}'|\theta) - k(\tilde{\mathbf{x}},\mathbf{X}|\theta)\left(\mathbf{K}_{\mathrm{f,f}}^{-1} - \mathbf{K}_{\mathrm{f,f}}^{-1}\,\mathrm{Cov}_{\mathbf{f}\,|\mathcal{D},\theta,\phi}[\mathbf{f}]\,\mathbf{K}_{\mathrm{f,f}}^{-1}\right)k(\mathbf{X},\tilde{\mathbf{x}}'|\theta). \tag{13}$$

From now on the posterior predictive mean and covariance will be denoted $m_p(\tilde{\mathbf{x}})$ and $k_p(\tilde{\mathbf{x}},\tilde{\mathbf{x}}')$.

Even if the exact posterior $p(\tilde{f}|\mathcal{D},\theta,\phi)$ is not available in closed form, we can still approximate its posterior mean and covariance functions if we can approximate $\mathrm{E}_{\mathbf{f}\,|\mathcal{D},\theta,\phi}$ and $\mathrm{Cov}_{\mathbf{f}\,|\mathcal{D},\theta,\phi}[\mathbf{f}]$. A common practice to approximate the posterior $p(\mathbf{f}\,|\mathcal{D},\theta,\phi)$ is either with Markov chain Monte Carlo (MCMC) (e.g. Neal, 1997, 1998; Diggle et al., 1998; Kuss and Rasmussen, 2005; Christensen et al., 2006) or by giving an analytic approximation to it (e.g. Williams and Barber, 1998; Gibbs and Mackay, 2000; Minka, 2001; Csató and Opper, 2002; Rue et al., 2009). The analytic approximations considered here assume a Gaussian form in which case it is natural to approximate the predictive distribution with a Gaussian as well. In this case, the equations (11) and (13) give its mean and covariance. Detailed considerations on the approximation error and the asymptotic properties of the Gaussian approximation are presented, for example, by Rue et al. (2009) and Vanhatalo et al. (2010).

3. Conditional posterior and predictive distributions

3.1 Gaussian observation model: the analytically tractable case

With a Gaussian observation model, $y_i \sim \mathrm{N}(f_i, \sigma^2)$, where σ^2 is the noise variance, the conditional posterior of the latent variables can be evaluated analytically. Marginalization over $\mathbf{f}$ gives the marginal likelihood

$$p(\mathbf{y}\,|\mathbf{X}, \theta, \sigma^2) = \mathrm{N}(\mathbf{y}\,|\mathbf{0}, \mathbf{K}_{\mathrm{f,f}} + \sigma^2\mathbf{I}). \tag{14}$$

Setting this in the denominator of the equation (8), gives a Gaussian distribution also for the conditional posterior of the latent variables

$$\mathbf{f}\,|\mathcal{D}, \theta, \sigma^2 \sim \mathrm{N}(\mathbf{K}_{\mathrm{f,f}}(\mathbf{K}_{\mathrm{f,f}} + \sigma^2\mathbf{I})^{-1}\,\mathbf{y}, \mathbf{K}_{\mathrm{f,f}} - \mathbf{K}_{\mathrm{f,f}}(\mathbf{K}_{\mathrm{f,f}} + \sigma^2\mathbf{I})^{-1}\,\mathbf{K}_{\mathrm{f,f}}). \tag{15}$$

Since the conditional posterior of $\mathbf{f}$ is Gaussian, the posterior process, or distribution $p(\tilde{f}|\mathcal{D})$, is also Gaussian. By placing the mean and covariance from (15) in the equations (11) and (13) we obtain the predictive distribution

$$\tilde{f}|\mathcal{D}, \theta, \sigma^2 \sim \mathcal{GP}\left(m_{\mathrm{p}}(\tilde{\mathbf{x}}), k_{\mathrm{p}}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')\right), \tag{16}$$

where the mean is $m_{\mathrm{p}}(\tilde{\mathbf{x}}) = k(\tilde{\mathbf{x}}, \mathbf{X})(\mathbf{K}_{\mathrm{f,f}} + \sigma^2\mathbf{I})^{-1}\,\mathbf{y}$ and covariance is $k_{\mathrm{p}}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') - k(\tilde{\mathbf{x}}, \mathbf{X})(\mathbf{K}_{\mathrm{f,f}} + \sigma^2\mathbf{I})^{-1}k(\mathbf{X}, \tilde{\mathbf{x}}')$. The predictive distribution for new observations $\tilde{\mathbf{y}}$ can be obtained by integrating $p(\tilde{\mathbf{y}}|\mathcal{D}, \theta, \sigma^2) = \int p(\tilde{\mathbf{y}}|\tilde{\mathbf{f}}, \sigma^2)p(\tilde{\mathbf{f}}|\mathcal{D}, \theta, \sigma^2)d\tilde{\mathbf{f}}$. The result is, again, Gaussian with mean $\mathrm{E}_{\tilde{\mathbf{f}}|\mathcal{D},\theta}[\tilde{\mathbf{f}}]$ and covariance $\mathrm{Cov}_{\tilde{\mathbf{f}}|\mathcal{D},\theta}[\tilde{\mathbf{f}}] + \sigma^2\mathbf{I}$.

3.2 Laplace approximation

With a non-Gaussian likelihood the conditional posterior needs to be approximated. The Laplace approximation is constructed from the second order Taylor expansion of $\log p(\mathbf{f}\,|\,\mathbf{y}, \theta, \phi)$ around the mode $\hat{\mathbf{f}}$, which gives a Gaussian approximation to the conditional posterior

$$p(\mathbf{f}\,|\mathcal{D}, \theta, \phi) \approx q(\mathbf{f}\,|\mathcal{D}, \theta, \phi) = \mathrm{N}(\mathbf{f}\,|\hat{\mathbf{f}}, \boldsymbol{\Sigma}), \tag{17}$$

where $\hat{\mathbf{f}} = \arg\max_{\mathbf{f}} p(\mathbf{f}\,|\mathcal{D}, \theta, \phi)$ and $\boldsymbol{\Sigma}^{-1}$ is the Hessian of the negative log conditional posterior at the mode (Gelman et al., 2004; Rasmussen and Williams, 2006):

$$\boldsymbol{\Sigma}^{-1} = -\nabla\nabla \log p(\mathbf{f}\,|\mathcal{D}, \theta, \phi)|_{\mathbf{f}=\hat{\mathbf{f}}} = \mathbf{K}_{\mathrm{f,f}}^{-1} + \mathbf{W}, \tag{18}$$

where $\mathbf{W}$ is a diagonal matrix with entries $\mathbf{W}_{ii} = \nabla_{f_i}\nabla_{f_i} \log p(y|f_i, \phi)|_{f_i=\hat{f}_i}$. We call the approximation scheme Laplace method following Williams and Barber (1998), but essentially the same approximation is named Gaussian approximation by Rue et al. (2009).

Setting $\mathrm{E}_{\mathbf{f}\,|\mathcal{D},\theta}[\mathbf{f}] = \hat{\mathbf{f}}$ and $\mathrm{Cov}_{\mathbf{f}\,|\mathcal{D},\theta}[\mathbf{f}] = (\mathbf{K}_{\mathrm{f,f}}^{-1} + \mathbf{W})^{-1}$ into (11) and (13) respectively, gives after rearrangements and using $\mathbf{K}_{\mathrm{f,f}}^{-1}\,\hat{\mathbf{f}} = \nabla \log p(\mathbf{y}\,|\,\mathbf{f})|_{\mathbf{f}=\hat{\mathbf{f}}}$, the approximate posterior predictive distribution

$$\tilde{f}|\mathcal{D}, \theta, \sigma^2 \sim \mathcal{GP}\left(m_{\mathrm{p}}(\tilde{\mathbf{x}}), k_{\mathrm{p}}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')\right). \tag{19}$$

Here the mean and covariance are $m_{\mathrm{p}}(\tilde{\mathbf{x}}) = k(\tilde{\mathbf{x}}, \mathbf{X})\nabla \log p(\mathbf{y}\,|\,\mathbf{f})|_{\mathbf{f}=\hat{\mathbf{f}}}$ and $k_{\mathrm{p}}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') - k(\tilde{\mathbf{x}}, \mathbf{X})(\mathbf{K}_{\mathrm{f,f}} + \mathbf{W})^{-1}k(\mathbf{X}, \tilde{\mathbf{x}}')$. The approximate conditional predictive density of $\tilde{y}_i$ can now be evaluated, for example, with quadrature integration over each $\tilde{f}_i$ separately

$$p(\tilde{y}_i|\mathcal{D}, \theta, \phi) \approx \int p(\tilde{y}_i|\tilde{f}_i, \phi)q(\tilde{f}_i|\mathcal{D}, \theta, \phi)d\tilde{f}_i. \tag{20}$$

3.3 Expectation propagation algorithm

The Laplace method constructs a Gaussian approximation at the posterior mode and approximates the posterior covariance via the curvature of the log density at that point. The expectation propagation (EP) algorithm (Minka, 2001), for its part, tries to minimize the Kullback-Leibler divergence from the true posterior to its approximation

$$q(\mathbf{f}\,|\mathcal{D},\theta,\phi) = \frac{1}{Z_{\mathrm{EP}}} p(\mathbf{f}\,|\theta) \prod_{i=1}^{n} t_i(f_i|\tilde{Z}_i,\tilde{\mu}_i,\tilde{\sigma}_i^2), \tag{21}$$

where the likelihood terms have been replaced by site functions $t_i(f_i|\tilde{Z}_i,\tilde{\mu}_i,\tilde{\sigma}_i^2) = \tilde{Z}_i\,\mathrm{N}(f_i|\tilde{\mu}_i,\tilde{\sigma}_i^2)$ and the normalizing constant by Z_{EP}. Detailed description of the algorithm is provided, for example by Rasmussen and Williams (2006) and Jylänki et al. (2011). EP's conditional posterior approximation is

$$q(\mathbf{f}\,|\mathcal{D},\theta,\phi) = \mathrm{N}(\mathbf{f}\,|\,\mathbf{K}_{\mathrm{f,f}}(\mathbf{K}_{\mathrm{f,f}}+\tilde{\Sigma})^{-1}\tilde{\mu}, \mathbf{K}_{\mathrm{f,f}} - \mathbf{K}_{\mathrm{f,f}}(\mathbf{K}_{\mathrm{f,f}}+\tilde{\Sigma})^{-1}\,\mathbf{K}_{\mathrm{f,f}}), \tag{22}$$

where $\tilde{\Sigma} = \mathrm{diag}[\tilde{\sigma}_1^2,...,\tilde{\sigma}_n^2]$ and $\tilde{\mu} = [\tilde{\mu}_1,...,\tilde{\mu}_n]^{\mathrm{T}}$. The predictive mean and covariance are again obtained from equations (11) and (13) analogically to the Laplace approximation.

From equations (15), (17), and (22) it can be seen that the Laplace and EP approximations are similar to the exact solution with the Gaussian likelihood. The diagonal matrices $\mathbf{W}^{-1}$ and $\tilde{\mathbf{\Sigma}}$ correspond to the noise variance $\sigma^2\mathbf{I}$ and, thus, these approximations can be interpreted as Gaussian approximations to the likelihood (Nickisch and Rasmussen, 2008).

3.4 Markov chain Monte Carlo

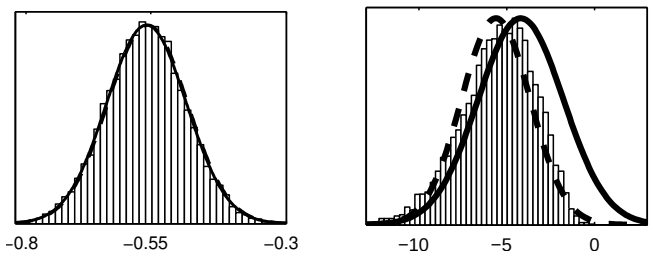
The accuracy of the approximations considered so far is limited by the Gaussian form of the approximating function. An approach, which gives exact solution in the limit of an infinite computational time, is the Monte Carlo integration (Robert and Casella, 2004). This is based on sampling from $p(\mathbf{f}\,|\mathcal{D},\theta,\ \phi)$ and using the samples to represent the posterior distribution.

In MCMC methods (Gilks et al., 1996), one constructs a Markov chain whose stationary distribution is the posterior distribution and uses the Markov chain samples to obtain Monte Carlo estimates. **GPstuff** provides, for example, a scaled Metropolis Hastings algorithm (Neal, 1998) and Hamiltonian Monte Carlo (HMC) (Duane et al., 1987; Neal, 1996) with variable transformation discussed in (Christensen et al., 2006; Vanhatalo and Vehtari, 2007) to sample from $p(\mathbf{f}\,|\mathcal{D},\theta,\ \phi)$. The approximations to the conditional posterior of $\mathbf{f}$ are illustrated in Figure 1.

After having the posterior sample of latent variables, we can sample from the posterior predictive distribution of any set of $\tilde{\mathbf{f}}$ simply by sampling with each $\mathbf{f}^{(i)}$ one $\tilde{\mathbf{f}}^{(i)}$ from $p(\tilde{\mathbf{f}}|\,\mathbf{f}^{(i)},\theta,\phi)$, which is given in the equation (7). Similarly, we can obtain a sample of $\tilde{\mathbf{y}}$ by drawing one $\tilde{\mathbf{y}}^{(i)}$ for each $\tilde{\mathbf{f}}^{(i)}$ from $p(\tilde{\mathbf{y}}|\tilde{\mathbf{f}},\theta,\phi)$.

4. Marginal likelihood given parameters

The marginal likelihood given the parameters, $p(\mathcal{D}|\theta,\phi) = \int p(\mathbf{y}|\,\mathbf{f},\phi)p(\mathbf{f}\,|\mathbf{X},\theta)d\,\mathbf{f}$, is an important quantity when inferring the parameters as discussed in the next section. With a



(a) Disease mapping (b) Classification

Figure 1: Illustration of the Laplace approximation (solid line), EP (dashed line) and MCMC (histogram) for the conditional posterior of a latent variable $p(f_i|\mathcal{D},\theta)$ in two applications. On the left, a disease mapping problem with Poisson likelihood (used in Vanhatalo et al., 2010) where the Gaussian approximation works well. On the right, a classification problem with probit likelihood (used in Vanhatalo and Vehtari, 2010) where the posterior is skewed and the Gaussian approximation is not so good.

Gaussian likelihood it has an analytic solution (14) which gives the log marginal likelihood

$$\log p(\mathcal{D}|\theta,\sigma) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\,\mathbf{K}_{\mathrm{f,f}} + \sigma^2\mathbf{I}| - \frac{1}{2}\,\mathbf{y}^{\mathrm{T}}(\mathbf{K}_{\mathrm{f,f}} + \sigma^2\mathbf{I})^{-1}\,\mathbf{y}\,. \tag{23}$$

If the likelihood is not Gaussian, the marginal likelihood needs to be approximated. The Laplace approximation to the marginal likelihood is constructed, for example, by making a second order Taylor expansion for the integrand $p(\mathbf{y}|\,\mathbf{f},\phi)p(\mathbf{f}\,|\mathbf{X},\theta)$ around $\hat{\mathbf{f}}$. This gives a Gaussian integral over $\mathbf{f}$ multiplied by a constant, and results in the log marginal likelihood approximation

$$\log p(\mathcal{D}|\theta,\phi) \approx \log q(\mathcal{D}|\theta,\phi) \propto -\frac{1}{2}\hat{\mathbf{f}}^{\mathrm{T}}\,\mathbf{K}_{\mathrm{f,f}}^{-1}\,\hat{\mathbf{f}} + \log p(\mathbf{y}|\hat{\mathbf{f}},\phi) - \frac{1}{2}\log|\mathbf{B}|, \tag{24}$$

where $|\mathbf{B}| = |I + \mathbf{W}^{1/2}\,\mathbf{K}_{\mathrm{f,f}}\,\mathbf{W}^{1/2}|$. See also (Tierney and Kadane, 1986; Rue et al., 2009; Vanhatalo et al., 2009) for more discussion.

EP's marginal likelihood approximation is the normalization constant

$$Z_{\mathrm{EP}} = \int p(\mathbf{f}\,|\mathbf{X},\theta)\prod_{i=1}^{n}\tilde{Z}_i\,\mathrm{N}(f_i|\tilde{\mu}_i,\tilde{\sigma}_i^2)df_i \tag{25}$$

in equation (21). This is a Gaussian integral multiplied by a constant $\prod_{i=1}^{n}\tilde{Z}_i$, giving

$$\log p(\mathcal{D}|\theta,\phi) \approx \log Z_{\mathrm{EP}} = -\frac{1}{2}\log|K+\tilde{\Sigma}| - \frac{1}{2}\tilde{\mu}^{\mathrm{T}}\left(K+\tilde{\Sigma}\right)^{-1}\tilde{\mu} + C_{\mathrm{EP}}, \tag{26}$$

where C_{EP} collects the terms that are not explicit functions of θ or ϕ (there is an implicit dependence through the iterative algorithm, though). For more discussion on EP's marginal likelihood approximation see (Seeger, 2005; Nickisch and Rasmussen, 2008).

5. Marginalization over parameters

The previous section treated methods to evaluate exactly (the Gaussian case) or approximately (Laplace and EP approximations) the log marginal likelihood given parameters. Now, we describe approaches for estimating parameters or integrating numerically over them.

5.1 Maximum a posterior estimate of parameters

In a full Bayesian approach we should integrate over all unknowns. Given we have integrated over the latent variables, it often happens that the posterior of the parameters is peaked or predictions are unsensitive to small changes in parameter values. In such case, we can approximate the integral over $p(\theta, \phi|\mathcal{D})$ with the maximum a posterior (MAP) estimate

$$\{\hat{\theta}, \hat{\phi}\} = \arg\max_{\theta,\phi} p(\theta, \phi|\mathcal{D}) = \arg\min_{\theta,\phi} \left[-\log p(\mathcal{D}|\theta, \phi) - \log p(\theta, \phi)\right]. \tag{27}$$

In this approximation, the parameter values are given a point mass one at the posterior mode, and the marginal of the latent function is approximated as $p(\mathbf{f}\,|\mathcal{D}) \approx p(\mathbf{f}\,|\mathcal{D}, \hat{\theta}, \hat{\phi})$.

The log marginal likelihood, and its approximations, are differentiable with respect to the parameters (Seeger, 2005; Rasmussen and Williams, 2006). Thus, also the log posterior is differentiable, which allows gradient based optimization. The advantage of MAP estimate is that it is relatively easy and fast to evaluate. According to our experience good optimization algorithms need usually at maximum tens of optimization steps to find the mode. However, it underestimates the uncertainty in parameters.

5.2 Grid integration

Grid integration is based on weighted sum of points evaluated on grid

$$p(\mathbf{f}\,|\mathcal{D}) \approx \sum_{i=1}^{M} p(\mathbf{f}\,|\mathcal{D}, \vartheta_i) p(\vartheta_i|\mathcal{D}) \Delta_i. \tag{28}$$

Here $\vartheta = [\theta^{\mathrm{T}}, \phi^{\mathrm{T}}]^{\mathrm{T}}$ and Δ_i denotes the area weight appointed to an evaluation point ϑ_i. The implementation follows INLA (Rue et al., 2009) and is discussed in detail by Vanhatalo et al. (2010). The basic idea is to first locate the posterior mode and then to explore the log posterior surface so that the bulk of the posterior mass is included in the integration (see Figure 2(a)). The grid search is feasible only for a small number of parameters since the number of grid points grows exponentially with the dimension of the parameter space d.

5.3 Monte Carlo integration

Monte Carlo integration scales better than the grid integration in large parameter spaces since its error decreases with a rate that is independent of the dimension (Robert and Casella, 2004). There are two options to find a Monte Carlo estimate for the marginal posterior $p(\mathbf{f}\,|\mathcal{D})$. The first option is to sample only the parameters from their marginal posterior $p(\vartheta|\mathcal{D})$ or from its approximation (see Figure 2(b)). In this case, the posterior marginal of the latent variable is approximated with mixture distribution as in the grid integration. The alternative is to sample both the parameters and the latent variables.

The full MCMC is performed by alternate sampling from the conditional posteriors $p(\mathbf{f}\,|\mathcal{D},\vartheta)$ and $p(\vartheta|\mathcal{D},\mathbf{f})$. Possible choices to sample from the conditional posterior of the parameters are, e.g., HMC and slice sampling (SLS) (Neal, 2003). Sampling both the parameters and latent variables is usually slow since due to the strong correlation between them (Vanhatalo and Vehtari, 2007; Vanhatalo et al., 2010). Sampling from the (approximate) marginal, $p(\vartheta|\mathcal{D})$, is an easier task since the parameter space is smaller.

The parameters can be sampled from their marginal posterior (or its approximation) with HMC, SLS (Neal, 2003) or via importance sampling (Geweke, 1989). In importance sampling, we use a Gaussian or Student-t proposal distribution $g(\vartheta)$ with mean $\hat{\vartheta}$ and covariance approximated with the negative Hessian of the log posterior, and approximate the integral with

$$p(\mathbf{f}\,|\mathcal{D}) \approx \frac{1}{\sum_{i=1}^{M} w_i} \sum_{i=1}^{M} q(\mathbf{f}\,|\mathcal{D},\vartheta_i) w_i, \tag{29}$$

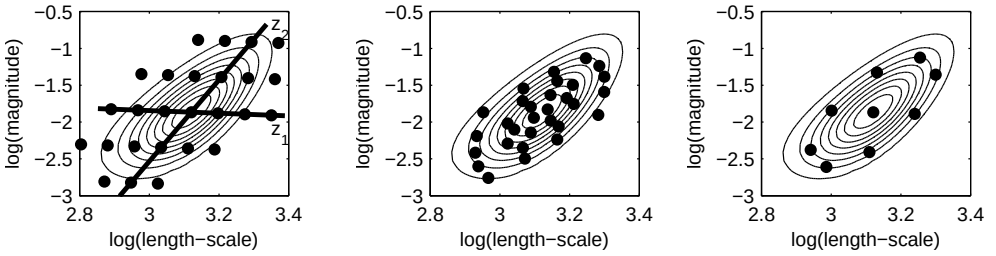
where $w_i = q(\vartheta^{(i)})/g(\vartheta^{(i)})$ are the importance weights. In some situations the naive Gaussian or Student-t proposal distribution is not adequate since the posterior distribution $q(\vartheta|\mathcal{D})$ may be non-symmetric or the covariance estimate is poor. An alternative for these situations is the scaled Student-t proposal distribution (Geweke, 1989) which is adjusted along each main direction of the approximate covariance. The implementation of the importance sampling is discussed in detail by Vanhatalo et al. (2010).

The problem with MCMC is that we are not able to draw independent samples from the posterior. Even with a careful tuning of Markov chain samplers the autocorrelation is usually so large that the required sample size is in thousands, which is a clear disadvantage compared with, for example, the MAP estimate.

5.4 Central composite design

Rue et al. (2009) suggest a central composite design (CCD) for choosing the representative points from the posterior of the parameters when the dimensionality of the parameters, d, is moderate or high. In this setting, the integration is considered as a quadratic design problem in a d dimensional space with the aim at finding points that allow to estimate the curvature of the posterior distribution around the mode. The design used in **GPstuff** is the fractional factorial design (Sanchez and Sanchez, 2005) augmented with a center point and a group of $2d$ star points. The design points are all on the surface of a d-dimensional sphere and the star points consist of $2d$ points along each axis, which is illustrated in Figure 2(c). The integration is then a finite sum (28) with special weights (Vanhatalo et al., 2010).

CCD integration speeds up the computations considerably compared to the grid search or Monte Carlo integration since the number of the design points grows very moderately. The accuracy of the CCD is between the MAP estimate and the full integration with the grid search or Monte Carlo. Rue et al. (2009) report good results with this integration scheme, and it has worked well in moderate dimensions in our experiments as well. Since CCD is based on the assumption that the posterior of the parameter is (close to) Gaussian, the densities $p(\vartheta_i|\mathcal{D})$ at the points on the circumference should be monitored in order to detect serious discrepancies from this assumption. These densities are identical if the posterior is Gaussian and we have located the mode correctly, and thereby great variability on their values indicates that CCD has failed. The posterior of the parameters may be far from a



(a) Grid based (b) Monte Carlo (c) Central composite design

Figure 2: The grid based, Monte Carlo and central composite design integration. Contours show the posterior density $q(\log(\vartheta)|\mathcal{D})$ and the integration points are marked with dots. The left figure shows also the vectors $\mathbf{z}$ along which the points are searched in the grid integration and central composite design. The integration is conducted over $q(\log(\vartheta)|\mathcal{D})$ rather than $q(\vartheta|\mathcal{D})$ since the former is closer to Gaussian. Reproduced from (Vanhatalo et al., 2010).

Gaussian distribution but for a suitable transformation, which is made automatically in the toolbox, the approximation may work well.

6. Getting started with GPstuff: regression and classification

6.1 Gaussian process regression

The demonstration program `demo_regression1` considers a simple regression problem $y_i = f(\mathbf{x}_i) + \epsilon_i$, where $\epsilon_i \sim N(0, \sigma^2)$. We will show how to construct the model with a squared exponential covariance function and how to conduct the inference.

6.1.1 Constructing the model

The model construction requires three steps: 1) Create structures that define likelihood and covariance function, 2) define priors for the parameters, and 3) create a GP structure where all the above are stored. These steps are done as follows:

```
lik = lik_gaussian('sigma2', 0.2^2);
gpcf = gpcf_sexp('lengthScale', [1.1 1.2], 'magnSigma2', 0.2^2)

pn=prior_logunif();
lik = lik_gaussian(lik, 'sigma2_prior', pn);

pl = prior_unif();
pm = prior_sqrtunif();
gpcf = gpcf_sexp(gpcf, 'lengthScale_prior', pl, 'magnSigma2_prior', pm);

gp = gp_set('lik', lik, 'cf', gpcf);
```

Here `lik_gaussian` initializes Gaussian likelihood function and its parameter values and `gpcf_sexp` initializes the squared exponential covariance function and its parameter values. `lik_gaussian` returns structure `lik` and `gpcf_sexp` returns `gpcf` that contain all the information needed in the evaluations (function handles, parameter values etc.). The next five lines create the prior structures for the parameters of the observation model and the covariance function, which are set in the likelihood and covariance function structures. The last line creates the GP structure by giving it the likelihood and covariance function.

Using the constructed GP structure, we can evaluate basic summaries such as covariance matrices, make predictions with the present parameter values etc. For example, the covariance matrices $\mathbf{K}_{\mathrm{f,f}}$ and $\mathbf{C} = \mathbf{K}_{\mathrm{f,f}} + \sigma^2\mathbf{I}$ for three two-dimensional input vectors are:

```
example_x = [-1 -1 ; 0 0 ; 1 1];
[K, C] = gp_trcov(gp, example_x)
K =
    0.0400    0.0187    0.0019
    0.0187    0.0400    0.0187
    0.0019    0.0187    0.0400
C =
    0.0800    0.0187    0.0019
    0.0187    0.0800    0.0187
    0.0019    0.0187    0.0800
```

6.1.2 MAP estimate for the parameters

`gp_optim` works as a wrapper for usual gradient based optimization functions. It is used as follows:

```
opt=optimset('TolFun',1e-3,'TolX',1e-3,'Display','iter');
gp=gp_optim(gp,x,y,'opt',opt);
```

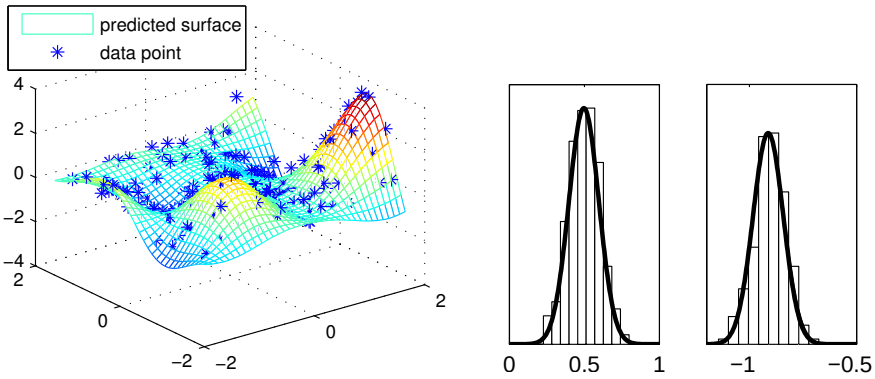
`gp_optim` takes a GP structure, training input $\mathbf{x}$, training target $\mathbf{y}$ (which are defined in `demo_regression1`) and options, and returns a GP structure with parameter values optimized to their MAP estimate. By default `gp_optim` uses `fminscg` function, but `gp_optim` can use also, for example, `fminlbfgs` or `fminunc`. Optimization options are set with `optimset` function. It is also possible to set optimisation options as

```
opt=struct('TolFun',1e-3,'TolX',1e-3,'Display','iter');
```

which is useful when using an optimiser not supported by `optimset`. All the estimated parameter values can be easily checked using the function `gp_pak`, which packs all the parameter values from all the covariance function structures in a vector, usually using log-transformation (other transformations are also possible). The second output argument of `gp_pak` lists the labels for the parameters:

```
[w,s] = gp_pak(gp);
disp(s), disp(exp(w))

    'log(sexp.magnSigma2)'
```



(a) The predictive mean and training data.

(b) The marginal posterior predictive distributions $p(f_i|\mathcal{D})$.

Figure 3: The predictive mean surface, training data, and the marginal posterior for two latent variables in `demo_regression1`. Histograms show the MCMC solution and the grid integration solution is drawn with a line.

```
'log(sexp.lengthScale x 2)'
'log(gaussian.sigma2)'

2.5981    0.8331    0.7878    0.0427
```

It is also possible to set the parameter vector of the model to desired values using `gp_unpak`. `gp_pak` and `gp_unpak` are used internally to allow use of generic optimisation and sampling functions, which take the parameter vector as an input argument.

Predictions for new locations $\tilde{\mathbf{x}}$, given the training data $(\mathbf{x}, \mathbf{y})$, are done by `gp_pred` function, which returns the posterior predictive mean and variance for each $f(\tilde{\mathbf{x}})$ (see equation (16)). This is illustrated below where we create a regular grid where the posterior mean and variance are computed. The posterior mean $m_p(\tilde{\mathbf{x}})$ and the training data points are shown in Figure 3.

```
[xt1,xt2]=meshgrid(-1.8:0.1:1.8,-1.8:0.1:1.8);
xt=[xt1(:) xt2(:)];
[Eft_map, Varft_map] = gp_pred(gp, x, y, xt);
```

6.1.3 Marginalization over parameters with grid integration

To integrate over the parameters we can use any method described in the section 5. The grid integration is performed with the following line:

```
[gp_array, P_TH, th, Eft_ia, Varft_ia, fx_ia, x_ia] = ...
                              gp_ia(gp, x, y, xt, 'int_method', 'grid');
```

`gp_ia` returns an array of GPs (`gp_array`) for parameter values `th` ($[\vartheta_i]_{i=1}^M$) with weights `P_TH` ($[p(\vartheta_i|\mathcal{D})\Delta_i]_{i=1}^M$). Since we use the grid method the weights are proportional to the marginal posterior and $\Delta_i \equiv 1 \forall i$ (see section 5.2). `Ef_ia` and `Varf_ia` contain the predictive mean and variance at the prediction locations. The last two output arguments can be used to plot the predictive distribution $p(\tilde{f}_i|\mathcal{D})$ as demonstrated in Figure 3. `x_ia(i,:)` contains a regular grid of values $\tilde{f}_i$ and `fx_ia(i,:)` contains $p(\tilde{f}_i|\mathcal{D})$ at those values.

6.1.4 Marginalization over parameters with MCMC

The main function for conducting Markov chain sampling is `gp_mc`, which loops through all the specified samplers in turn and saves the sampled parameters in a record structure. In later sections, we will discuss models where also latent variables are sampled, but now we concentrate on the covariance function parameters, which are sampled as follows:

```
[gp_rec,g,opt] = gp_mc(gp, x, y, 'nsamples', 220);
gp_rec = thin(gp_rec, 21, 2);
[Eft_s, Varft_s] = gpmc_preds(rfull, x, y, xt);
[Eft_mc, Varft_mc] = gp_pred(gp_rec, x, y, xt);
```

The `gp_mc` function generates `nsamples` (here 220) Markov chain samples. At each iteration `gp_mc` runs the actual samplers. The function `thin` removes the burn-in from the sample chain (here 21) and thins the chain more (here by 2). This way we can decrease the autocorrelation between the remaining samples. **GPstuff** provides also diagnostic tools for Markov chains. See, for example, (Gelman et al., 2004; Robert and Casella, 2004) for discussion on convergence and other Markov chain diagnostics. The function `gpmc_preds` returns the conditional predictive mean and variance for each sampled parameter value. These are $\mathrm{E}_{p(\mathbf{f}|\mathbf{X},\mathcal{D},\vartheta^{(s)})}[\tilde{\mathbf{f}}], s = 1, ..., M$ and $\mathrm{Var}_{p(\mathbf{f}|\mathbf{X},\mathcal{D},\vartheta^{(s)})}[\tilde{\mathbf{f}}], s = 1, ..., M$, where M is the number of samples. Marginal predictive mean and variance are computed directly with `gp_pred`.

6.2 Gaussian process classification

We will now consider a binary GP classification (see `demo_classific`) with observations, $y_i \in \{-1, 1\}, i = 1, ..., n$, associated with inputs $\mathbf{X} = \{\mathbf{x}\}_{i=1}^n$. The observations are considered to be drawn from a Bernoulli distribution with a success probability $p(y_i = 1|\mathbf{x}_i)$. The probability is related to the latent function via a sigmoid function that transforms it to a unit interval. **GPstuff** provides a probit and logit transformation, which give

$$p_{\mathrm{probit}}(y_i|f(\mathbf{x}_i)) = \Phi(y_i f(\mathbf{x}_i)) = \int_{-\infty}^{y_i f(\mathbf{x}_i)} N(z|0,1)dz \tag{30}$$

$$p_{\mathrm{logit}}(y_i|f(\mathbf{x}_i)) = \frac{1}{1+\exp(-y_i f(\mathbf{x}_i))}. \tag{31}$$

Since the likelihood is not Gaussian we need to use approximate inference methods, discussed in the section 3.

6.2.1 Constructing the model

The model construction for the classification follows closely the steps presented in the previous section. The model is constructed as follows:

```
lik = lik_probit();
gpcf = gpcf_sexp('lengthScale', [0.9 0.9], 'magnSigma2', 10);
gp = gp_set('lik', lik, 'cf', gpcf, 'jitterSigma2', 1e-9);
```

The above lines first initialize the likelihood function, the covariance function and the GP structure. Since we do not specify parameter priors, the default priors are used. The model construction and the inference with logit likelihood (`lik_logit`) would be similar with probit likelihood. A small jitter value is added to the diagonal of the training covariance to make certain matrix operations more stable (this is not usually necessary but is shown here for illustration).

6.2.2 Inference with Laplace approximation

The MAP estimate for the parameters can be found using `gp_optim` as

```
gp = gp_set(gp, 'latent_method', 'Laplace');
gp = gp_optim(gp,x,y,'opt',opt);
```

The first line defines which inference method is used for the latent variables. It initializes the Laplace algorithm and sets needed fields in the GP structure. The default method for latent variables is Laplace, so this line could be omitted. `gp_optim` uses the default optimization function, `fminscg`, with the same options as above in the regression example.

`gp_pred` provides the mean and variance for the latent variables (first two outputs), the log predictive probability for a test observation (third output), and mean and variance for the observations (fourth and fifth output) at test locations `xt`.

```
[Eft_la, Varft_la, lpyt_la, Eyt_la, Varyt_la] = ...
                            gp_pred(gp, x, y, xt, 'yt', ones(size(xt,1),1) );
```

The first four input arguments are the same as in the section 6.1. The fifth and sixth arguments are a parameter-value pair where `yt` tells that we give test observations `ones(size(xt,1),1)` related to `xt` as an optional input, in which case `gp_pred` evaluates their marginal posterior log predictive probabilities `lpyt_la`. Here we evaluate the probability to observe class 1 and thus we give a vector of ones as test observations.

6.2.3 Inference with expectation propagation

EP works as the Laplace approximation. We only need to set the latent method to `'EP'` but otherwise the commands are the same as above:

```
gp = gp_set(gp, 'latent_method', 'EP');
gp = gp_optim(gp,x,y,'opt',opt);
[Eft_ep, Varft_ep, lpyt_ep, Eyt_ep, Varyt_ep] = ...
                     gp_pred(gp, x, y, xt, 'yt', ones(size(xt,1),1) );
```

6.2.4 Inference with MCMC

With MCMC we sample both the latent variables and the parameters:

```
gp = gp_set(gp, 'latent_method', 'MCMC', 'jitterSigma2', 1e-6);
[gp_rec,g,opt]=gp_mc(gp, x, y, 'nsamples', 220);
gp_rec=thin(gp_rec,21,2);
[Ef_mc, Varf_mc, lpy_mc, Ey_mc, Vary_mc] = ...
    gp_pred(gp_rec, x, y, xt, 'yt', ones(size(xt,1),1) );
```

For MCMC we need to add a larger jitter on the diagonal to prevent numerical problems. By default sampling from the latent value distribution $p(\mathbf{f}\,|\theta, \mathcal{D})$ is done with the elliptical slice sampling (Murray et al., 2010). Other samplers provided by the **GPstuff** are a scaled Metropolis Hastings algorithm (Neal, 1998) and a scaled HMC `scaled_hmc` (Vanhatalo and Vehtari, 2007). From the user point of view, the actual sampling and prediction are performed similarly as with the Gaussian likelihood. The `gp_mc` function handles the sampling so that it first samples the latent variables from $p(\mathbf{f}\,|\theta, \mathcal{D})$ after which it samples the parameters from $p(\theta|\,\mathbf{f}, \mathcal{D})$. This is repeated until `nsamples` samples are drawn.

In classification model MCMC is the most accurate inference method, then comes EP and Laplace approximation is the worst. However, the inference times line up in the opposite order. The difference between the approximations is not always this large. For example, with Poisson likelihood, discussed in the section 10.2, Laplace and EP approximations work, in our experience, practically as well as MCMC.

7. Sparse Gaussian processes

The evaluation of the inverse and determinant of the covariance matrix in the log marginal likelihood (or its approximation) and its gradient scale as $O(n^3)$ in time. This restricts the implementation of GP models to moderate size data sets. However, the unfavorable scaling in computational time can be alleviated with sparse approximations to GP and compactly supported (CS) covariance functions. The sparse approximations scale as $O(nm^2)$, where $m < n$ and the CS covariance functions lead to computations which scale, in general, as $O(n^3)$ but with a smaller constant than with traditional globally supported covariance functions. Many sparse GPs are proposed in the literature and some of them are build into **GPstuff**.

7.1 Compactly supported covariance functions

A CS covariance function gives zero correlation between data points whose distance exceeds a certain threshold. This leads to a sparse covariance matrix. The challenge with constructing CS covariance functions is to guarantee their positive definiteness and much literature has been devoted on the subject (see e.g. Sansò and Schuh, 1987; Wu, 1995; Wendland, 1995; Gaspari and Cohn, 1999; Gneiting, 1999, 2002; Buhmann, 2001) The CS functions implemented in **GPstuff** are Wendland's piecewise polynomials $k_{\mathrm{pp},q}$ (Wendland, 2005), such as

$$k_{\mathrm{pp},2} = \frac{\sigma^2_{\mathrm{pp}}}{3}(1-r)_+^{j+2}\left((j^2+4j+3)r^2+(3j+6)r+3\right), \tag{32}$$

where $j = \lfloor d/2 \rfloor + 3$. These functions correspond to processes that are q times mean square differentiable and are positive definite up to an input dimension d. Thus, the degree of the polynomial has to be increased alongside the input dimension. The dependence of CS

covariance functions to the input dimension is very fundamental. There are no radial CS functions that are positive definite on every $\Re^d$ (see e.g. Wendland, 1995, theorem 9.2).

The key idea with CS covariance functions is that, roughly speaking, only the nonzero elements of the covariance matrix are used in the calculations. This may speed up the calculations substantially since in some situations only a fraction of the elements of the covariance matrix are non-zero (see e.g. Vanhatalo and Vehtari, 2008; Rue et al., 2009). In practice, efficient sparse matrix routines are needed (Davis, 2006). These are nowadays a standard utility in many statistical computing packages or available as an additional package for them. **GPstuff** utilizes the sparse matrix routines from **SuiteSparse** written by Tim Davis (`http://www.cise.ufl.edu/research/sparse/SuiteSparse/`) and this package should be installed before using CS covariance functions.

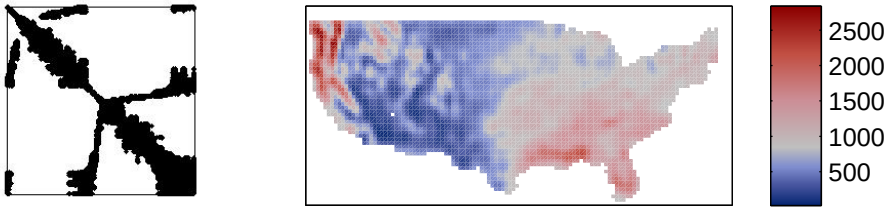
The CS covariance functions have been rather widely studied in the geostatistics applications (see e.g. Gneiting, 2002; Furrer et al., 2006; Moreaux, 2008). There the computational speed-up relies on efficient linear solvers for the predictive mean $[\tilde{\mathbf{f}}] = \mathbf{K}_{\tilde{\mathrm{f}},\mathrm{f}}(\mathbf{K}_{\mathrm{f},\mathrm{f}} + \sigma^2)^{-1}\,\mathbf{y}$. The parameters are either fitted to the empirical covariance, optimized using a line search in one dimension (Kaufman et al., 2008) or sampled with Metropolis Hastings. The benefits from a sparse covariance matrix have been immediate since the problems collapse to solving sparse linear systems. However, utilizing the gradient of the log posterior of the parameters, as is done in **GPstuff** needs extra sparse matrix tools. These are introduced and discussed by Vanhatalo and Vehtari (2008). EP algorithm requires also special considerations with CS covariance functions. The posterior covariance in EP (22) does not remain sparse, and thereby it has to be expressed implicitly. This issue is discussed by Vanhatalo and Vehtari (2010) and Vanhatalo et al. (2010).

7.1.1 Compactly supported covariance functions in GPstuff

The demo `demo_regression_ppcs` contains a regression example with CS covariance function $k_{\mathrm{pp},2}$ (`gpcf_ppcs2`). The data contain US annual precipitation summaries from year 1995 for 5776 observation stations. The user interface of **GPstuff** makes no difference between globally and compactly supported covariance functions but the code is optimized to use sparse matrix routines whenever the covariance matrix is sparse. Thus, we can construct the model, find the MAP estimate for the parameters and predict to new input locations in a familiar way:

```
pn = prior_t('nu', 4, 's2', 0.3);
lik = lik_gaussian('sigma2', 1, 'sigma2_prior', pn);
pl2 = prior_gamma('sh', 5, 'is', 1);
pm2 = prior_sqrtt('nu', 1, 's2', 150);
gpcf2 = gpcf_ppcs2('nin', nin, 'lengthScale', [1 2], 'magnSigma2', 3, ...
                   'lengthScale_prior', pl2, 'magnSigma2_prior', pm2);
gp = gp_set('lik', lik, 'cf', gpcf2, 'jitterSigma2', 1e-6);
gp = gp_optim(gp,x,y,'opt',opt);
Eft = gp_pred(gp, x, y, xx);
```

With this data the covariance matrix is rather sparse since only about 5% of its elements are non-zero. The structure of the covariance matrix is plotted after the approximate minimum



(a) The nonzero elements of $\mathbf{K}_{\mathrm{f,f}}$.

(b) The posterior predictive mean surface.

Figure 4: The nonzero elements of $\mathbf{K}_{\mathrm{f,f}}$ with $k_{\mathrm{pp},2}$ function, and the posterior predictive mean of the latent function in the US precipitation data set.

degree (AMD) permutation (Davis, 2006) in Figure 4. The demo `demo_spatial2` illustrates the use of compactly supported covariance functions with a non-Gaussian likelihood.

7.2 FIC and PIC sparse approximations

Snelson and Ghahramani (2006) proposed a sparse pseudo-input GP (SPGP), which Quiñonero-Candela and Rasmussen (2005) named later fully independent conditional (FIC). The partially independent conditional (PIC) sparse approximation is an extension of FIC (Quiñonero-Candela and Rasmussen, 2005; Snelson and Ghahramani, 2007), and they are both treated here following Quiñonero-Candela and Rasmussen (2005). See also (Vanhatalo and Vehtari, 2008; Vanhatalo et al., 2010) for further discussion. The approximations are based on introducing an additional set of latent variables $\mathbf{u} = \{u_i\}_{i=1}^m$, called *inducing variables*. These correspond to a set of input locations $\mathbf{X}_u$, called *inducing inputs*. The latent function prior is approximated as

$$p(\mathbf{f}\,|\mathbf{X},\theta) \approx q(\mathbf{f}\,|\mathbf{X},\mathbf{X}_u,\theta) = \int q(\mathbf{f}\,|\mathbf{X},\mathbf{X}_u,\mathbf{u},\theta)p(\mathbf{u}\,|\mathbf{X}_u,\theta)\mathrm{d}\mathbf{u}, \tag{33}$$

where $q(\mathbf{f}\,|\mathbf{X},\mathbf{X}_u,\mathbf{u},\theta)$ is an inducing conditional. The above decomposition leads to the exact prior if the true conditional $\mathbf{f}\,|\mathbf{X},\mathbf{X}_u,\mathbf{u},\theta \sim \mathrm{N}(\mathbf{K}_{\mathrm{f,u}}\mathbf{K}_{\mathrm{u,u}}^{-1}\mathbf{u}, \mathbf{K}_{\mathrm{f,f}} - \mathbf{K}_{\mathrm{f,u}}\mathbf{K}_{\mathrm{u,u}}^{-1}\mathbf{K}_{\mathrm{u,f}})$ is used. However, in FIC framework the latent variables are assumed to be conditionally independent given $\mathbf{u}$, in which case the inducing conditional factorizes $q(\mathbf{f}\,|\mathbf{X},\mathbf{X}_u,\mathbf{u},\theta) = \prod q_i(f_i|\mathbf{X},\mathbf{X}_u,\mathbf{u},\theta)$. In PIC latent variables are set in blocks which are conditionally independent of each others, given $\mathbf{u}$, but the latent variables within a block have a multivariate normal distribution with the original covariance. The approximate conditionals of FIC and PIC can be summarized as

$$q(\mathbf{f}\,|\mathbf{X},\mathbf{X}_u,\mathbf{u},\theta,\mathbf{M}) = \mathrm{N}(\mathbf{f}\,|\,\mathbf{K}_{\mathrm{f,u}}\mathbf{K}_{\mathrm{u,u}}^{-1}\mathbf{u}, \mathrm{mask}\left(\mathbf{K}_{\mathrm{f,f}} - \mathbf{K}_{\mathrm{f,u}}\mathbf{K}_{\mathrm{u,u}}^{-1}\mathbf{K}_{\mathrm{u,f}}|\mathbf{M}\right)), \tag{34}$$

where the function $\mathbf{\Lambda} = \mathrm{mask}\,(\cdot|\mathbf{M})$, with matrix $\mathbf{M}$ of ones and zeros, returns a matrix $\mathbf{\Lambda}$ of size $\mathbf{M}$ and elements $\mathbf{\Lambda}_{ij} = [\cdot]_{ij}$ if $\mathbf{M}_{ij} = 1$ and $\mathbf{\Lambda}_{ij} = 0$ otherwise. An approximation with

$\mathbf{M} = \mathbf{I}$ corresponds to FIC and an approximation where $\mathbf{M}$ is block diagonal corresponds to PIC. The inducing variables are given a zero-mean Gaussian prior $\mathbf{u}\,|\theta, \mathbf{X}_u \sim \mathrm{N}(\mathbf{0}, \mathbf{K}_{\mathrm{u,u}})$ so that the approximate prior over latent variables is

$$q(\mathbf{f}\,|\mathbf{X}, \mathbf{X}_u, \theta, \mathbf{M}) = \mathrm{N}(\mathbf{f}\,|\mathbf{0}, \mathbf{K}_{\mathrm{f,u}}\,\mathbf{K}_{\mathrm{u,u}}^{-1}\,\mathbf{K}_{\mathrm{u,f}} + \mathbf{\Lambda}). \tag{35}$$

The matrix $\mathbf{K}_{\mathrm{f,u}}\,\mathbf{K}_{\mathrm{u,u}}^{-1}\,\mathbf{K}_{\mathrm{u,f}}$ is of rank m and $\mathbf{\Lambda}$ is a rank n (block) diagonal matrix. The prior covariance above can be seen as a non-stationary covariance function of its own where the inducing inputs $\mathbf{X}_u$ and the matrix $\mathbf{M}$ are free parameters similar to parameters, which can be optimized alongside θ (Snelson and Ghahramani, 2006; Lawrence, 2007).

The computational savings are obtained by using the Woodbury-Sherman-Morrison lemma (e.g. Harville, 1997) to invert the covariance matrix in (35) as

$$(\mathbf{K}_{\mathrm{f,u}}\,\mathbf{K}_{\mathrm{u,u}}^{-1}\,\mathbf{K}_{\mathrm{u,f}} + \mathbf{\Lambda})^{-1} = \mathbf{\Lambda}^{-1} - \mathbf{V}\mathbf{V}^{\mathrm{T}}, \tag{36}$$

where $\mathbf{V} = \mathbf{\Lambda}^{-1}\,\mathbf{K}_{\mathrm{f,u}}\,\mathrm{chol}[(\mathbf{K}_{\mathrm{u,u}} + \mathbf{K}_{\mathrm{u,f}}\,\mathbf{\Lambda}^{-1}\,\mathbf{K}_{\mathrm{f,u}})^{-1}]$. There is a similar result also for the determinant. With FIC the computational time is dominated by the matrix multiplications, which need time $O(m^2 n)$. With PIC the cost depends also on the sizes of the blocks in $\mathbf{\Lambda}$. If the blocks were of equal size $b \times b$, the time for inversion of $\mathbf{\Lambda}$ would be $O(n/b \times b^3) = O(nb^2)$. With blocks at most the size of the number of inducing inputs, that is $b = m$, the computational cost in PIC and FIC are similar. PIC approaches FIC in the limit of a block size one and the exact GP in the limit of a block size n (Snelson, 2007).

7.2.1 FIC sparse approximation in GPstuff

The same data that were discussed in the section 6.1 is analyzed with sparse approximations in the demo `demo_regression_sparse1`. The sparse approximation is a property of the GP structure and we can construct the model similarly to the full GP models:

```
gp_fic = gp_set('type', 'FIC', 'lik', lik, 'cf', gpcf, 'X_u', X_u)
```

The difference is that we have to define the type of the sparse approximation, here `'FIC'`, and set the inducing inputs `X_u` in the GP structure. Since the inducing inputs are considered as extra parameters common to all of the covariance functions (there may be more than one covariance function in additive models) they are set in the GP structure instead of the covariance function structure. If we want to optimize the inducing inputs alongside the parameters they need to have a prior as well. GP structure initialization gives them a uniform prior by default.

We can optimize all the parameters, including the inducing inputs, as with a full GP. Sometimes, for example in spatial problems, it is better to fix the inducing inputs (Vanhatalo et al., 2010) or it may be more efficient to optimize the parameters and inducing inputs separately, so that we iterate the separate optimization steps until convergence (demonstrated in `demo_regression_sparse1`). The parameters to be optimized are defined by the field `infer_params` in the GP structure. This field regulates which parameters are considered fixed and which are inferred in the group level (covariance function, inducing inputs, likelihood). We may also want to fix one of the parameters inside these groups. For example, covariance function magnitude. If this is the case, then the parameter to be fixed should be given an empty prior (`prior_fixed`). If the parameter has a prior structure it is an indicator that we want to infer that parameter.

7.2.2 PIC sparse approximation in GPstuff

In PIC, in addition to defining the inducing inputs, we need to appoint every data point in a block. The block structure is common to all covariance functions, similarly to the inducing inputs, for which reason the block information is stored in the GP structure. After the blocking and initialization of the inducing inputs the GP structure is constructed as follows:

```
gp_pic = gp_set('type','PIC','lik',lik,'cf',gpcf,'X_u',X_u,'tr_index',trindex);
```

Above, the cell array `trindex` contains the block index vectors for training data. It means that, for example, the inputs and outputs `x(trindex{i},:)` and `y(trindex{i},:)` belong to the i'th block. The optimization of parameters and inducing inputs is done the same way as with FIC or a full GP model. In prediction, however, we have to give one extra input, `tstindex`, for `gp_pred`. This defines how the prediction inputs are appointed in the blocks in a same manner as `trindex` appoints the training inputs.

```
Eft_pic = gp_pred(gp_pic, x, y, xt, 'tstind', tstindex);
```

7.3 Deterministic training conditional, subset of regressors and variational sparse approximation

The deterministic training conditional is based on the works by Csató and Opper (2002) and Seeger et al. (2003) and was earlier called Projected Latent Variables (see Quiñonero-Candela and Rasmussen, 2005, for more details). The approximation can be constructed similarly as FIC and PIC by defining the inducing conditional, which in the case of DTC is

$$q(\mathbf{f}\,|\mathbf{X}, \mathbf{X}_u, \mathbf{u}, \theta) = \mathrm{N}(\mathbf{f}\,|\,\mathbf{K}_{\mathrm{f,u}}\,\mathbf{K}_{\mathrm{u,u}}^{-1}\,\mathbf{u}, 0). \tag{37}$$

This implies that the approximate prior over latent variables is

$$q(\mathbf{f}\,|\mathbf{X}, \mathbf{X}_u, \theta) = \mathrm{N}(\mathbf{f}\,|\mathbf{0}, \mathbf{K}_{\mathrm{f,u}}\,\mathbf{K}_{\mathrm{u,u}}^{-1}\,\mathbf{K}_{\mathrm{u,f}}). \tag{38}$$

The deterministic training conditional is not strictly speaking a proper GP since it uses different covariance function for the latent variables appointed to the training inputs and for the latent variables at the prediction sites, $\tilde{f}$. The prior covariance for $\tilde{f}$ is the true covariance $\mathbf{K}_{\tilde{\mathrm{f}},\tilde{\mathrm{f}}}$ instead of $\mathbf{K}_{\tilde{\mathrm{f}},\mathrm{u}}\,\mathbf{K}_{\mathrm{u,u}}^{-1}\,\mathbf{K}_{\mathrm{u},\tilde{\mathrm{f}}}$. This does not affect the predictive mean since the cross covariance $\mathrm{Cov}[\mathbf{f}, \tilde{\mathbf{f}}] = \mathbf{K}_{\mathrm{f,u}}\,\mathbf{K}_{\mathrm{u,u}}^{-1}\,\mathbf{K}_{\mathrm{u},\tilde{\mathrm{f}}}$, but it gives a larger predictive variance. An older version of DTC is the subset of regressors (SOR) sparse approximation which utilizes $\mathbf{K}_{\tilde{\mathrm{f}},\mathrm{u}}\,\mathbf{K}_{\mathrm{u,u}}^{-1}\,\mathbf{K}_{\mathrm{u},\tilde{\mathrm{f}}}$. However, this resembles a singular Gaussian distribution and thus the predictive variance may be negative. DTC tries to fix this problem by using $\mathbf{K}_{\tilde{\mathrm{f}},\tilde{\mathrm{f}}}$ (see Quiñonero-Candela and Rasmussen, 2005). DTC and SOR are identical in other respects than in the predictive variance evaluation. In spatial statistics, SOR has been used by Banerjee et al. (2008) with a name Gaussian predictive process model.

The approximate prior of the variational approximation by Titsias (2009) is exactly the same as that of DTC. The difference between the two approximations is that in the variational setting the inducing inputs and covariance function parameters are optimized differently. The inducing inputs and parameters can be seen as variational parameters that

should be chosen to maximize the variational lower bound between the true GP posterior and the sparse approximation. This leads to optimization of modified log marginal likelihood

$$V(\theta, \mathbf{X}_u) = \log[N(\mathbf{y}\,|0, \sigma^2\mathbf{I} + \mathbf{Q}_{\mathrm{f,f}})] - \frac{1}{2\sigma^2}\mathrm{tr}(\mathbf{K}_{\mathrm{f,f}} - \mathbf{K}_{\mathrm{f,u}}\,\mathbf{K}_{\mathrm{u,u}}^{-1}\,\mathbf{K}_{\mathrm{u,f}}) \tag{39}$$

with Gaussian likelihood. With non-Gaussian likelihood, the variational lower bound is similar but $\sigma^2\mathbf{I}$ is replaced by $\mathbf{W}^{-1}$ (Laplace approximation) or $\tilde{\mathbf{\Sigma}}$ (EP).

7.3.1 Variational, DTC and SOR sparse approximation in GPstuff

The variational, DTC and SOR sparse approximations are constructed similarly to FIC. Only the type of the GP changes:

```
gp_var = gp_set('type', 'VAR', 'lik', lik, 'cf', gpcf, 'X_u', X_u);
gp_dtc = gp_set('type', 'DTC', 'lik', lik, 'cf', gpcf, 'X_u', X_u);
gp_sor = gp_set('type', 'SOR', 'lik', lik, 'cf', gpcf, 'X_u', X_u);
```

The sparse GP approximations are compared in the demo `demo_regression_sparse2`, and, for example, Quiñonero-Candela and Rasmussen (2005), Snelson (2007), Titsias (2009), and Alvarez et al. (2010) treat them in more detail.

7.4 Sparse GP models with non-Gaussian likelihoods

The extension of sparse GP models to non-Gaussian likelihoods is straightforward in **GPstuff**. User can define the sparse GP just as described in the previous two sections and then continue with the construction of likelihood exactly the same way as with a full GP. The Laplace approximation, EP and integration methods can be used with the same commands as with full GP. This is demonstrated in `demo_spatial1`.

8. Model assessment and comparison

There are various means to assess the goodness of the model and its predictive performance. **GPstuff** provides four basic approaches: marginal likelihood, cross-validation, deviance information criterion and widely applicable information criterion.

8.1 Marginal likelihood

Marginal likelihood is often used for model selection (see, e.g. Kass and Raftery, 1995). It corresponds to ML II or with model priors to MAP II estimate in the model space, selecting the model with the highest marginal likelihood or highest marginal posterior probability. In **GPstuff** the marginal posterior and marginal likelihood (or its approximation in case of non-Gaussian likelihood) given the parameters are computed by function `gp_e`.

8.2 Cross-validation

Cross-validation (CV) is an approach to estimate the predictive performance for the future observations while avoiding the double use of the data. The ith observation $(\mathbf{x}_i, y_i)$ in the training data is left out, and then the predictive distribution for y_i is computed with a model

that is fitted to all of the observations except $(\mathbf{x}_i, y_i)$. By repeating this for every point in the training data, we get a collection of leave-one-out cross-validation (LOO-CV) predictive densities.

For GP with given hyperparameters the LOO-CV predictions can be computed in case of Gaussian likelihood using an analytical solution (Sundararajan and Keerthi, 2001) and in case of non-Gaussian likelihood using EP approximation (Opper and Winther, 2000; Rasmussen and Williams, 2006) or Laplace approximation using linear response approach (to be submitted), which have been implemented in `gp_loopred`. If set of hyperparameters have been obtained using `gp_mc` or `gp_ia` LOO-posterior can be approximated using importance sampling (Gelfand et al., 1992; Vehtari and Lampinen, 2002), which is also implemented in `gp_loopred`.

To reduce computation time, in k-fold-CV, $1/k$ part of the training data is left out, and then the predictive distribution is computed with a model that is fitted to all of the observations except that part. Since the k-fold-CV predictive densities are based on smaller training data sets than the full data set, the estimate is slightly biased. In **GPstuff** first order bias correction proposed by Burman (1989) is used. **GPstuff** provides `gp_kfcv`, which computes k-fold-CV and bias-corrected k-fold-CV with log-score and root mean squared error (RMSE). The function `gp_kfcv` provides also basic variance estimates for the predictive performance estimates. Variance is computed from the estimates for each fold (see, e.g., Dietterich, 1998). See (Vehtari and Lampinen, 2002) for more details on estimating the uncertainty in performance estimates.

8.3 DIC

Deviance information criterion (DIC) is another very popular model selection criterion (Spiegelhalter et al., 2002). DIC is not fully Bayesian approach as it estimates the predictive performance if the predictions were made using point-estimate (plug-in estimate) for the unknowns, instead of using posterior predictive distribution. Additionally DIC is defined only for regular models and thus fails for singular models. DIC is included in **GPstuff** to allow comparisons and better alternative WAIC is described in the next section.

With parametric models without any hierarchy DIC is usually written as

$$p_{\text{eff}} = \mathrm{E}_{\theta|\mathcal{D}}[D(\mathbf{y},\theta)] - D(\mathbf{y}, \mathrm{E}_{\theta|\mathcal{D}}[\theta]) \tag{40}$$

$$\mathrm{DIC} = \mathrm{E}_{\theta|\mathcal{D}}[D(\mathbf{y},\theta)] + p_{\text{eff}}, \tag{41}$$

where p_{eff} is the effective number of parameters and $D = -2\log(p(\mathbf{y}\,|\theta))$ is the deviance. Since our models are hierarchical we need to decide the parameters on focus (see Spiegelhalter et al., 2002, for discussion on this). The parameters on the focus are those over which the expectations are taken when evaluating the effective number of parameters and DIC. In the above equations, the focus is in the parameters and in the case of a hierarchical GP model of **GPstuff** the latent variables would be integrated out before evaluating DIC. If we have a MAP estimate for the parameters, we may be interested to evaluate DIC statistics with the focus on the latent variables. In this case the above formulation would be

$$p_D(\theta) = \mathrm{E}_{\mathbf{f}\,|\mathcal{D},\theta}[D(\mathbf{y},\mathbf{f})] - D(\mathbf{y}, \mathrm{E}_{\mathbf{f}\,|\mathcal{D},\theta}[\mathbf{f}]) \tag{42}$$

$$\mathrm{DIC} = \mathrm{E}_{\mathbf{f}\,|\mathcal{D},\theta}[D(\mathbf{y},\mathbf{f})] + p_D(\theta). \tag{43}$$

Here the effective number of parameters is denoted differently with $p_D(\theta)$ since now we are approximating the effective number of parameters in $\mathbf{f}$ conditionally on θ, which is different from the p_{eff}. $p_D(\theta)$ is a function of the parameters and it measures to what extent the prior correlations are preserved in the posterior of the latent variables given θ. For non-informative data $p_D(\theta) = 0$ and the posterior is the same as the prior. The greater $p_D(\theta)$ is the more the model is fitted to the data and large values compared to n indicate potential overfit. Also, large $p_D(\theta)$ indicates that we cannot assume that the conditional posterior approaches normality by central limit theorem. Thus, $p_D(\theta)$ can be used for assessing the goodness of the Laplace or EP approximation for the conditional posterior of the latent variables as discussed by Rue et al. (2009) and Vanhatalo et al. (2010). The third option is to evaluate DIC with focus on all the variables, $[\mathbf{f}, \theta]$. In this case the expectations are over $p(\mathbf{f}, \theta|\mathcal{D})$.

8.4 WAIC

Watanabe (2009, 2010a,b) presented widely applicable information criterion (WAIC) and gave a formal proof of its properties as an estimate for the predictive performance of posterior predictive distributions for both regular and singular models. A criterion of similar form was independently proposed by Richardson (2002) as a version of DIC, but without formal justification.

Other information criteria are based on Fisher's asymptotic theory assuming a regular model for which the likelihood or the posterior converges to a single point and MLE, MAP, and plug-in estimates are asymptotically equivalent. With singular models the set of true parameters consists of more than one point, the Fisher information matrix is not positive definite, plug-in estimates are not representative of the posterior and the distribution of the deviance does not converge to a χ^2_ν distribution.

Watanabe shows that the Bayesian generalization utility can be estimated by a criterion

$$\text{WAIC}_G = BU_t - 2(BU_t - GU_t), \tag{44}$$

where BU_t is Bayes training utility

$$BU_t = \frac{1}{n}\sum_{i=1}^{n} \log p(y_i|D, M_k) \tag{45}$$

and GU_g is Gibbs generalization utility

$$GU_g = \int p_t(\tilde{y}) \int p(\theta|D, M_k) \log p(\tilde{y}|\theta, M_k) d\theta d\tilde{y} \tag{46}$$

WAIC can also be given as a functional variance form

$$\text{WAIC}_V = BU_t - V/n, \tag{47}$$

where the functional variance

$$V = \sum_{i=1}^{n} \Bigg\{ \text{E}_{\theta|D,M_k}\left[\left(\log p(y_i|x_i, \theta, M_k)\right)^2\right] - \left(\text{E}_{\theta|D,M_k}\left[\log p(y_i|x_i, \theta, M_k)\right]\right)^2 \Bigg\}, \tag{48}$$

describes the fluctuation of the posterior distribution.

WAIC is asymptotically equal to the true logarithmic utility in both regular and singular statistical models and the error in a finite case is $o(1/n)$. Watanabe (2010b) shows also that the WAIC estimate is asymptotically equal to the Bayesian cross-validation estimate (section 8.2). WAIC_G and WAIC_V are asymptotically equal, but the series expansion of $WAIC_V$ has closer resemblance to the series expansion of the logarithmic leave-one-out utility.

8.5 Model assessment demos

The model assessment methods are demonstrated with the functions `demo_modelassesment1` and `demo_modelassesment2`. The former compares the sparse GP approximations to the full GP with regression data and the latter compares the logit and probit likelihoods in GP classification.

Assume that we have built our regression model with a Gaussian noise and used optimization method to find the MAP estimate for the parameters. We evaluate the effective number of latent variables, DIC and WAIC

```
p_eff_latent = gp_peff(gp, x, y);
[DIC_latent, p_eff_latent2] = gp_dic(gp, x, y, 'focus', 'latent');
WAIC = gp_waic(gp,x,y);
```

where p_{eff} is evaluated with two different approximations. Since we have the MAP estimate for the parameters the focus is on the latent variables. In this case we can also use `gp_peff` which returns the effective number of parameters approximated as

$$p_D(\theta) \approx n - \mathrm{tr}(\mathbf{K}_{\mathrm{f,f}}^{-1}(\mathbf{K}_{\mathrm{f,f}}^{-1} + \sigma^{-2}\mathbf{I})^{-1}) \tag{49}$$

(Spiegelhalter et al., 2002). When the focus is on the latent variables, the function `gp_dic` evaluates the DIC statistics and the effective number of parameters as described by the equations (42) and (43).

The k-fold-CV expected utility estimate can be evaluated as follows:

```
cvres =  gp_kfcv(gp, x, y);
```

The `gp_kfcv` takes the ready made model structure `gp` and the training data `x` and `y`. The function divides the data into k groups, conducts inference separately for each of the training groups and evaluates the expected utilities with the test groups. Since no optional parameters are given the inference is conducted using MAP estimate for the parameters. The default division of the data is into 10 groups. The expected utilities and their variance estimates are stored in the structure `cvres`. `gp_kfcv` returns also other statistics if more information is needed and the function can be used to save the results automatically.

Assume now that we have a record structure from `gp_mc` function with Markov chain samples of the parameters stored in it. In this case, we have two options how to evaluate the DIC statistics. We can set the focus on the parameters or all the parameters (that is parameters and latent variables). The two versions of DIC and the effective number of parameters and WAIC are evaluated as follows:

```
rgp = gp_mc(gp, x, y, opt);
[DIC, p_eff] =  gp_dic(rgp, x, y, 'focus', 'param');
[DIC2, p_eff2] =  gp_dic(rgp, x, y, 'focus', 'all');
WAIC = gp_waic(gp,x,y);
```

Here the first line performs the MCMC sampling with options `opt`. The next two lines evaluate the DIC statistics and the last line evaluates WAIC. With Markov chain samples, we cannot use the `gp_peff` function to evaluate $p_D(\theta)$ since that is a special function for models with fixed parameters.

The functions `gp_peff`, `gp_dic` and `gp_kfcv` work similarly for non-Gaussian likelihoods as for a Gaussian one. The only difference is that the integration over the latent variables is done approximately. The way the latent variables are treated is defined in the field `latent_method` of the GP structure and this is initialized when constructing the model as discussed in the section 6.2. The effective number of parameters returned by `gp_peff` is evaluated as in the equation (49) with the modification that $\sigma^{-2}\mathbf{I}$ is replaced by $\mathbf{W}$ in the case of Laplace approximation and $\tilde{\mathbf{\Sigma}}^{-1}$ in the case of EP.

9. Modifying the covariance functions

9.1 Additive models

In many practical situations, a GP prior with only one covariance function may be too restrictive since such a construction can model effectively only one phenomenon. For example, the latent function may vary rather smoothly across the whole area of interest, but at the same time it can have fast local variations. In this case, a more reasonable model would be $f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x})$, where the latent value function is a sum of two functions, slow and fast varying. By placing a separate GP prior for both of the functions g and h we obtain an additive prior

$$f(\mathbf{x})|\theta \sim \mathcal{GP}(0, k_g(\mathbf{x}, \mathbf{x}') + k_h(\mathbf{x}, \mathbf{x}')). \tag{50}$$

The marginal likelihood and posterior distribution of the latent variables are as before with $\mathbf{K}_{\mathrm{f,f}} = \mathbf{K}_{\mathrm{g,g}} + \mathbf{K}_{\mathrm{h,h}}$. However, if we are interested on only, say, phenomenon g, we can consider the h part of the latent function as correlated noise and evaluate the predictive distribution for g, which with the Gaussian likelihood would be

$$\tilde{g}(\tilde{\mathbf{x}})|\mathcal{D}, \theta \sim \mathcal{GP}\left(k_g(\tilde{\mathbf{x}}, \mathbf{X})(\mathbf{K}_{\mathrm{f,f}} + \sigma^2\mathbf{I})^{-1}\mathbf{y}, k_g(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') - k_g(\tilde{\mathbf{x}}, \mathbf{X})(\mathbf{K}_{\mathrm{f,f}} + \sigma^2\mathbf{I})^{-1}k_g(\mathbf{X}, \tilde{\mathbf{x}}')\right). \tag{51}$$

With non-Gaussian likelihood, the Laplace and EP approximations for this are similar since only $\sigma^2\mathbf{I}$ and $(\mathbf{K}_{\mathrm{f,f}} + \sigma^2\mathbf{I})^{-1}\mathbf{y}$ change in the approximations.

The multiple length-scale model can be formed also using specific covariance functions. For example, a rational quadratic covariance function (`gpcf_rq`) can be seen as a scale mixture of squared exponential covariance functions (Rasmussen and Williams, 2006), and could be useful for data that contain both local and global phenomena. However, using sparse approximations with the rational quadratic would prevent it from modeling local phenomena. The additive model (50) suits better for sparse GP formalism since it enables to combine FIC with CS covariance functions.

As discussed in section 7.2, FIC can be interpreted as a realization of a special kind of covariance function. Vanhatalo and Vehtari (2008) proposed to add FIC with CS covariance function which leads to a latent variable prior

$$\mathbf{f} \mid \mathbf{X}, \mathbf{X}_u, \theta \sim \mathrm{N}(\mathbf{0}, \mathbf{K}_{\mathrm{f,u}} \mathbf{K}_{\mathrm{u,u}}^{-1} \mathbf{K}_{\mathrm{u,f}} + \hat{\mathbf{\Lambda}}), \tag{52}$$

referred as CS+FIC. Here, the matrix $\hat{\mathbf{\Lambda}} = \mathbf{\Lambda} + k_{\mathrm{pp,q}}(\mathbf{X}, \mathbf{X})$ is sparse with the same sparsity structure as in $k_{\mathrm{pp,q}}(\mathbf{X}, \mathbf{X})$ and it is fast to use in computations and cheap to store.

9.1.1 Additive models in GPstuff

The additive models are demonstrated in `demo_periodic` and `demo_regression_additive1`. Their construction follows closely the steps introduced in the previous sections. Consider that we want to construct a GP with a covariance function that is a sum of squared exponential and piecewise polynomial $k_{\mathrm{se}}(\mathbf{x}, \mathbf{x}') + k_{\mathrm{pp},2}(\mathbf{x}, \mathbf{x}')$. In **GPstuff** this is done as follows:

```
gpcf1 = gpcf_sexp();
gpcf2 = gpcf_ppcs2('nin', 2);
lik = lik_gaussian('sigma2', 0.1);
gp = gp_set('lik', lik, 'cf', {gpcf1, gpcf2});
```

The difference to previous models is that we give more than one covariance function structure in cell array for the `gp_set`. The inference with additive model is conducted the same way as earlier. If we want to make predictions for the two components k_{se} and $k_{\mathrm{pp},2}$ independently we can do it by giving an extra parameter-value pair for the `gp_pred` as

```
[Ef_sexp, Varf_sexp] = gp_pred(gp, x, y, x, 'predcf', 1);
[Ef_ppcs2, Varf_ppcs2] = gp_pred(gp, x, y, x, 'predcf', 2);
```

Additive models are constructed analogously with sparse approximations by changing the type of the GP model. CS+FIC is a special kind of GP structure and has its own type definition:

```
gp = gp_set('type', 'CS+FIC', 'lik', lik, 'cf', {gpcf1, gpcf2}, 'X_u', Xu);
```

It is worth mentioning few things that should be noticed in relation to sparse approximations. In general, FIC is able to model only phenomena whose length-scale is long enough compared to the distance between adjacent inducing inputs. PIC on the other hand is able to model also fast varying phenomena inside the blocks. Its drawback, however, is that the correlation structure is discontinuous which may result in discontinuous predictions. The CS+FIC model corrects these deficiencies. In FIC and PIC the inducing inputs are parameters of every covariance function, which means that all the correlations are induced through the inducing inputs and the shortest length-scale the GP is able to model is defined by the locations of the inducing inputs. In CS+FIC, the CS covariance functions do not utilise inducing inputs but evaluate the covariance exactly for which reason both the long and short length-scale phenomena can be captured. If there are more than two covariance functions in CS+FIC all the globally supported functions utilize inducing inputs and all the CS functions are added to $\hat{\Lambda}$. A detailed treatment on the subject is given in (Vanhatalo and Vehtari, 2008, 2010; Vanhatalo et al., 2010)

9.2 Additive covariance functions with selected variables

In the demo (`demo_regression_additive2`), we demonstrate how covariance functions can be modified so that they are functions of only a subset of inputs. We model an artificial 2D regression data with additive covariance functions that use only either the first or second input variable. That is, the covariance is $k_1(x_1, x_1'|\theta_1) + k_2(x_2, x_2'|\theta_2) : \Re^2 \times \Re^2 \mapsto \Re$, where the covariance functions are of type $k_1(x_1, x_1'|\theta_1) : \Re \times \Re \mapsto \Re$. In the example the covariance is a sum of a squared exponential, which is a function of the first input dimension x_1, and a linear covariance function, which is a function of the second input dimension x_2. The covariance functions are constructed as follows:

```
gpcf_s1 = gpcf_sexp('selectedVariables', 1, 'lengthScale',0.5, ...
                    'magnSigma2', 0.15);
gpcf_l2 = gpcf_linear('selectedVariables', 2);
```

The smaller set of inputs is chosen with the field input variable pair `'selectedVariables',1`.

9.3 Product of covariance functions

A product of two or more covariance functions $k_1(\mathbf{x}, \mathbf{x}') \cdot k_2(\mathbf{x}, \mathbf{x}')...$ is a valid covariance function as well. Such constructions may be useful in situations where the phenomenon is assumed to be separable. Combining covariance functions into product form is done with a special covariance function `gpcf_prod`. For example, multiplying exponential and Mátern covariance functions to produce separable spatio-temporal model $k(\mathbf{x}, \mathbf{x}') = k_1(x_1, x_1') \cdot k_2([x_2, x_3]^{\mathrm{T}}, [x_2', x_3']^{\mathrm{T}})$ where the temporal component has covariance function k_1 and the spatial components k_2 is done as follows:

```
gpcf1 = gpcf_exp('selectedVariables', 1);
gpcf2 = gpcf_matern32('selectedVariables', [2 3];
gpcf = gpcf_prod('cf', {gpcf1, gpcf2});
```

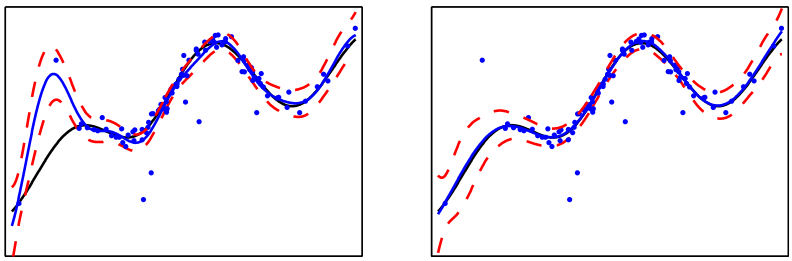
The product covariance `gpcf_prod` can also be used to combine categorical covariance `gpcf_cat` with other covariance functions to build hierarchical linear and non-linear models, as illustrated in `demo_regression_hier`.

10. Other models

In this section, we discuss three more example applications where **GPstuff** has been used. The purpose of this section is to introduce few other likelihood functions implemented in **GPstuff** but not discussed yet.

10.1 Robust regression with Student-t observation model

A commonly used observation model in the GP regression is the Gaussian distribution. This is convenient since the marginal likelihood is analytically tractable. However, a known limitation with the Gaussian observation model is its non-robustness, due which outlying observations may significantly reduce the accuracy of the inference (see Figure 5). A well-



(a) Gaussian likelihood. (b) Student-t likelihood.

Figure 5: An example of regression with outliers from (Neal, 1997). On the left Gaussian and on the right the Student-t likelihood. The real function is plotted with black line.

known robust observation model is the Student-t distribution (O'Hagan, 1979)

$$\mathbf{y} \,|\, \mathbf{f}, \nu, \sigma_t \sim \prod_{i=1}^{n} \frac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)\sqrt{\nu\pi}\sigma_t} \left(1 + \frac{(y_i - f_i)^2}{\nu\sigma_t^2}\right)^{-(\nu+1)/2}, \tag{53}$$

where ν is the degrees of freedom and σ_t the scale parameter. The Student-t distribution can be utilized as such or it can be written via the scale mixture representation

$$y_i | f_i, \alpha, U_i \sim N(f_i, \alpha U_i) \tag{54}$$

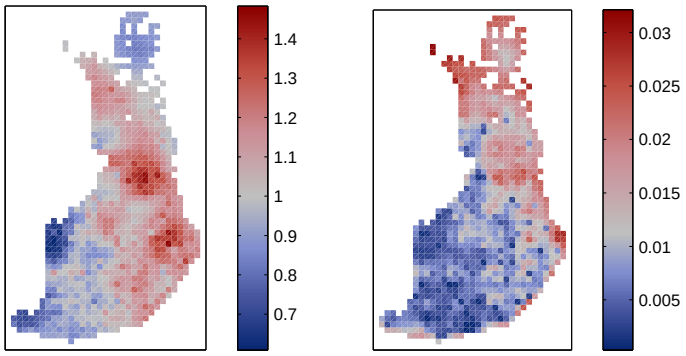
$$U_i \sim \text{Inv-}\chi^2(\nu, \tau^2), \tag{55}$$

where each observation has its own noise variance αU_i that is Inv-χ^2 distributed (Neal, 1997; Gelman et al., 2004). The degrees of freedom ν corresponds to the degrees of freedom in the Student-t distribution and $\alpha\tau$ corresponds to σ_t.

In **GPstuff** both of the representations are implemented. The scale mixture representation is implemented in `lik_gaussiansmt` and can be inferred only with MCMC (as described by Neal, 1998). The Student-t likelihood is implemented in `lik_t` and can be inferred with Laplace and EP approximation and MCMC (as described by Vanhatalo et al., 2009; Jylänki et al., 2011). These are demonstrated in `demo_regression_robust`.

10.2 Disease mapping

Spatial epidemiology concerns both describing and understanding the spatial variation in the disease risk in geographically referenced health data. One of the most common tasks in spatial epidemiology is disease mapping, where the aim is to describe the overall disease distribution on a map and, for example, highlight areas of elevated or lowered mortality or morbidity risk (e.g. Lawson, 2001; Richardson, 2003; Elliot et al., 2001). Here we build a disease mapping model following the general approach discussed, for example, by Best et al. (2005). The data are aggregated into areas with coordinates $\mathbf{x}_i$. The mortality/morbidity in an area is modeled with a Poisson, negative binomial or binomial distribution with mean



(a) The posterior mean. (b) The posterior variance.

Figure 6: The posterior predictive mean and variance of the relative risk in the `demo_spatial1` data set obtained with FIC.

$e_i\mu_i$, where e_i is the standardized expected number of cases (e.g. Ahmad et al., 2000), and μ_i is the relative risk, whose logarithm is given a GP prior. The aim is to infer the relative risk. The implementation of these models in **GPstuff** is discussed in (Vanhatalo and Vehtari, 2007; Vanhatalo et al., 2010).

10.2.1 Poisson likelihood

The Poisson likelihood is implemented in `likelih_poisson` and it is

$$\mathbf{y} \,|\, \mathbf{f}, \mathbf{e} \sim \prod_{i=1}^{n} \text{Poisson}(y_i | \exp(f_i) e_i) \tag{56}$$

where the vector $\mathbf{y}$ collects the numbers of deaths for each area. Here $\mu = \exp(f)$ and its posterior predictive mean and variance solved in the demo `demo_spatial1` are shown in Figure 6.

10.2.2 Negative binomial likelihood

The negative binomial distribution is a robust version of the Poisson distribution similarly as Student-t distribution can be considered as a robustified Gaussian distribution (Gelman et al., 2004). In **GPstuff** it is parametrized as

$$\mathbf{y} \,|\, \mathbf{f}, \mathbf{e}, r \sim \prod_{i=1}^{n} \frac{\Gamma(r + y_i)}{y_i! \Gamma(r)} \left(\frac{r}{r + \mu_i} \right)^r \left(\frac{\mu_i}{r + \mu_i} \right)^{y_i}, \tag{57}$$

where $\mu_i = e_i \exp(f(\mathbf{x}_i))$ and r is the dispersion parameter governing the variance. The model is demonstrated in `demo_spatial2`.

10.2.3 Binomial likelihood

In disease mapping, the Poisson distribution is used to approximate the true binomial distribution. This approximation works well if the background population, corresponding to the number of trials in the binomial distribution, is large. Sometimes this assumption is not adequate and we need to use the exact binomial observation model

$$\mathbf{y} \,|\, \mathbf{f}, \mathbf{z} \sim \prod_{i=1}^{n} \frac{z_i!}{y_i!(z_i - y_i)!} p_i^{y_i} (1 - p_i)^{(z_i - y_i)}, \tag{58}$$

where $p_i = \exp(f(\mathbf{x}_i))/(1+\exp(f(\mathbf{x}_i)))$ is the probability of success, and the vector $\mathbf{z}$ denotes the number of trials. The binomial observation model is not limited to spatial modeling but is an important model for other problems as well. The observation model is demonstrated in `demo_binomial1` with a one dimensional simulated data and `demo_binomial_apc` demonstrates the model in an incidence risk estimation.

10.2.4 Hurdle model

Hurdle models can be used to model excess number of zeros compared to usual Poisson and negative binomial count models. Hurdle models assume a two-stage process, where the first process determines whether the count is larger than zero, and the second process determines the non-zero count (Mullahy, 1986). These processes factorize, and thus hurdle model can be implemented using two independent GPs in GPstuff. `lik_probit` or `lik_logit` can be used for the zero process and `lik_negbinztr` can be used for the count part. `lik_negbinztr` provides zero truncated negative binomial model, which can be used also to approximate zero-truncated Poisson model by using high dispersion parameter value. Construction of a hurdle model is demonstrated in `demo_hurdle`. Gaussian process model with logit negative binomial hurdle model implemented using **GPstuff** was used in reference (Rantonen et al., 2011) to model sick absence days due to low back symptoms.

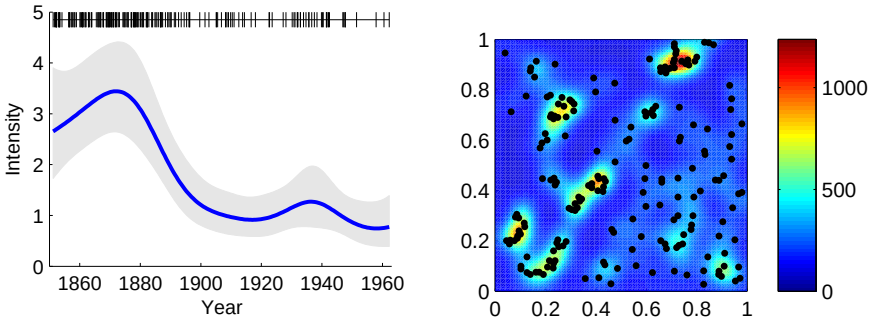
10.3 Log-Gaussian Cox process

Log-Gaussian Cox-process is an inhomogeneous Poisson process model used for point data, with unknown intensity function $\lambda(\mathbf{x})$, modeled with log-Gaussian process so that $f(\mathbf{x}) = \log \lambda(\mathbf{x})$ (see Rathbun and Cressie, 1994; Møller et al., 1998). If the data are points $\mathbf{X} = \mathbf{x}_i$; $i = 1, 2, \ldots, n$ on a finite region $\mathcal{V}$ in $\Re^d$, then the likelihood of the unknown function f is

$$p(\mathbf{X}|f) = \exp\left\{ - \left(\int_{\mathcal{V}} \exp(f(\mathbf{x})) d\,\mathbf{x} \right) + \sum_{i=1}^{n} f(\mathbf{x}_i) \right\}. \tag{59}$$

Evaluation of the likelihood would require nontrivial integration over the exponential of GP. Møller et al. (1998) propose to discretise the region $\mathcal{V}$ and assume locally constant intensity in subregions. This transforms the problem to a form equivalent to having Poisson model for each subregion. Likelihood after the discretisation is

$$p(\mathbf{X}|f) \approx \prod_{k=1}^{K} \text{Poisson}(y_k | \exp(f(\dot{\mathbf{x}}_k))), \tag{60}$$



(a) Coal mine disasters. (b) Redwood data.

Figure 7: Two intensity surfaces estimated with log-Gaussian Cox process. The figures are from the `demo_lgcp`, where the aim is to study an underlying intensity surface of a point process. On the left a temporal and on the right a spatial point process.

where $\dot{\mathbf{x}}$ is the coordinate of the kth sub-region and y_k is the number of data points in it. Tokdar and Ghosh (2007) proved the posterior consistency in limit when sizes of subregions go to zero.

The log-Gaussian Cox process with Laplace and EP approximation is implemented in the function `lgcp` for one or two dimensional input data. The usage of the function is demonstrated in `demo_lgcp`. This demo analyzes two data sets. The first one is one dimensional case data with coal mine disasters (from R distribution). The data contain the dates of 191 coal mine explosions that killed ten or more men in Britain between 15 March 1851 and 22 March 1962. The analysis is conducted using expectation propagation and CCD integration over the parameters and the results are shown in Figure 7. The second data are the redwood data (from R distribution). This data contain 195 locations of redwood trees in two dimensional lattice. The smoothed intensity surface is shown in Figure 7.

10.4 Survival analysis with Weibull model

The Weibull distribution is a widely used parametric baseline hazard function in survival analysis (Ibrahim et al., 2001). The hazard rate for the observation i is

$$h_i(y) = h_0(y) \exp(f(\mathbf{x}_i)), \tag{61}$$

where $y > 0$ and the baseline hazard $h_0(y)$ is assumed to follow the Weibull distribution parametrized in **GPstuff** as

$$h_0(y) = r y^{r-1}, \tag{62}$$

where $r > 0$ is the shape parameter. This follows the parametrization used in Martino et al. (2011). The observation model is defined as

$$\mathbf{y} \,|\, \mathbf{f}, \mathbf{z}, r \sim \prod_{i=1}^{n} r^{1-z_i} \exp\left((1-z_i) f(\mathbf{x}_i) + (1-z_i)(r-1)\log(y_i) - \exp(f(\mathbf{x}_i)) y_i^r\right), \tag{63}$$

where $\mathbf{z}$ is a vector of censoring indicators with $z_i = 0$ for uncensored event and $z_i = 1$ for right censored event for observation i. The use of Weibull model with survival data is demonstrated in `demo_survival_weibull`.

10.5 Derivative observations in GP regression

Incorporating derivative observations in GP regression is fairly straightforward, because a derivative of Gaussian process is a Gaussian process. In short, derivative observation are taken into account by extending covariance matrices to include derivative observations. This is done by forming joint covariance matrices of function values and derivatives. Following equations (Rasmussen and Williams, 2006) state how the covariances between function values and derivatives, and between derivatives are calculated

$$\mathrm{Cov}(f_i, \frac{\partial f_j}{\partial x_{dj}}) = \frac{\partial k(\mathbf{x}_i, \mathbf{x}_j)}{\partial x_{dj}}), \qquad \mathrm{Cov}(\frac{\partial f_i}{\partial x_{di}}, \frac{\partial f_j}{\partial x_{ej}}) = \frac{\partial^2 k(\mathbf{x}_i, \mathbf{x}_j}{\partial x_{di} \partial x_{ej}}.$$

The joint covariance matrix for function values and derivatives is of the following form

$$\begin{aligned}
\mathbf{K} &= \begin{bmatrix} \mathbf{K}_{ff} & \mathbf{K}_{fD} \\ \mathbf{K}_{Df} & \mathbf{K}_{DD} \end{bmatrix} \\
\mathbf{K}_{ff}^{ij} &= k(\mathbf{x}_i, \mathbf{x}_j), \\
\mathbf{K}_{Df}^{ij} &= \frac{\partial k(\mathbf{x}_i, \mathbf{x}_j)}{\partial x_{di}}, \\
\mathbf{K}_{fD} &= (\mathbf{K}_{Df})^\top, \\
\mathbf{K}_{DD}^{ij} &= \frac{\partial^2 k(\mathbf{x}_i, \mathbf{x}_j)}{\partial x_{di} \partial x_{ej}},
\end{aligned}$$

Prediction is done as usual but with derivative observations joint covariance matrices are to be used instead of the normal ones.

Using derivative observations in GPstuff requires two steps: when initializing the GP structure one must set option `'derivobs'` to `'on'`. The second step is to form right sized observation vector. With input size $n \times m$ the observation vector with derivatives should be of size $n + m \cdot n$. The observation vector is constructed by adding partial derivative observations after function value observations

$$\mathbf{y}_{obs} = \begin{bmatrix} y(\mathbf{x}) \\ \frac{\partial y(\mathbf{x})}{\partial x_1} \\ \vdots \\ \frac{\partial y(\mathbf{x})}{\partial x_m} \end{bmatrix}. \tag{64}$$

Different noise level could be assumed for function values and derivative observations but at the moment the implementation allows only same noise for all the observations. The use of derivative observations is demonstrated in `demo_derivativeobs`.

10.6 Mean functions

In the standard GP regression a zero mean function is assumed for the prior process. This is convenient but there are nonetheless some advantages in using a specified mean function. The basic principle in doing GP regression with a mean function is to apply a zero mean GP for the difference between observations and the mean function.

10.6.1 Explicit basis functions

Here we follow closely the presentation of (Rasmussen and Williams, 2006) about the subject and briefly present the main results. A mean function can be specified as a weighted sum of some basis functions h

$$\mathbf{m} = \mathbf{h}(\mathbf{x})^\top \boldsymbol{\beta},$$

with weights $\boldsymbol{\beta}$. The target of modeling, the underlying process $\mathbf{g}$, is assumed to be a sum of mean function and a zero mean GP.

$$\mathbf{g} = \mathbf{h}(\mathbf{x})^\top \boldsymbol{\beta} + \mathcal{GP}(0, \mathbf{K}).$$

By assuming Gaussian prior for the weights $\boldsymbol{\beta} \sim \mathcal{N}(\mathbf{b}, \mathbf{B})$, the weight parameters can be integrated out and the prior for $\mathbf{g}$ is another GP

$$\textit{prior}\, \mathbf{g} \sim \mathcal{GP}\left(\mathbf{h}(\mathbf{x})^\top \mathbf{b},\, \mathbf{K} + \mathbf{h}(\mathbf{x})^\top \mathbf{B}\, \mathbf{h}(\mathbf{x})\right).$$

The predictive equations are obtained by using the mean and covariance of the prior (65) in the zero mean GP predictive equations (15)

$$\begin{aligned}
\mathbb{E}(\mathbf{g}_*) &= \mathbb{E}(\mathbf{f}_*) + \mathbf{R}^\top \overline{\boldsymbol{\beta}}, && (65)\\
\mathrm{Cov}(\mathbf{g}_*) &= \mathrm{Cov}(\mathbf{f}_*) + \mathbf{R}^\top \left(\mathbf{B}^{-1} + \mathbf{H}\mathbf{K}_y^{-1}\mathbf{H}^\top\right)\mathbf{R}, && (66)
\end{aligned}$$

$$\begin{aligned}
\overline{\boldsymbol{\beta}} &= \left(\mathbf{B}^{-1} + \mathbf{H}\mathbf{K}_y^{-1}\mathbf{H}^\top\right)^{-1}\left(\mathbf{B}^{-1}\mathbf{b} + \mathbf{H}\mathbf{K}_y^{-1}\mathbf{y}\right),\\
\mathbf{R} &= \mathbf{H}_* - \mathbf{H}\mathbf{K}_y^{-1}\mathbf{K}_*,\\
\mathbf{H} &= \begin{bmatrix} h_1(\mathbf{x}) \\ h_2(\mathbf{x}) \\ \vdots \\ h_k(\mathbf{x}) \end{bmatrix}, \boldsymbol{x}\ \textit{is row vector.}
\end{aligned}$$

If the prior assumptions about the weights are vague then $\mathbf{B}^{-1} \to O$, (O is a zero matrix) and the predictive equations (65) and (66) don't depend on $\mathbf{b}$ or $\mathbf{B}$

$$\begin{aligned}
\mathbb{E}(\mathbf{g}_*) &= \mathbb{E}(\mathbf{f}_*) + \mathbf{R}^\top \overline{\boldsymbol{\beta}}_{\boldsymbol{v}}, && (67)\\
\mathrm{Cov}(\mathbf{g}_*) &= \mathrm{Cov}(\mathbf{f}_*) + \mathbf{R}^\top \left(\mathbf{H}\mathbf{K}_y^{-1}\mathbf{H}^\top\right)\mathbf{R}, && (68)
\end{aligned}$$

$$\overline{\boldsymbol{\beta}}_{\boldsymbol{v}} = \left(\mathbf{H}\mathbf{K}_y^{-1}\mathbf{H}^\top\right)^{-1}\mathbf{H}\mathbf{K}_y^{-1}\mathbf{y},$$

Corresponding the exact and vague prior for the basis functions' weights there are two versions of the marginal likelihood. With exact prior the marginal likelihood is

$$\begin{aligned}
\log p(\mathbf{y} \mid \mathbf{X}, \mathbf{b}, \mathbf{B}) &= -\frac{1}{2}\mathbf{M}^\top \mathbf{N}^{-1}\mathbf{M} - \frac{1}{2}\log|\mathbf{K}_y| - \frac{1}{2}\log|\mathbf{B}| - \frac{1}{2}\log|\mathbf{A}| - \frac{n}{2}\log 2\pi, \\
\mathbf{M} &= \mathbf{H}^\top \mathbf{b} - \mathbf{y}, \\
\mathbf{N} &= \mathbf{K}_y + \mathbf{H}^\top \mathbf{B}\mathbf{H}, \\
\mathbf{A} &= \mathbf{B}^{-1} + \mathbf{H}\mathbf{K}_y^{-1}\mathbf{H}^\top,
\end{aligned}$$

where n is the amount of observations. Its derivative with respect to hyperparameters are

$$\begin{aligned}
\frac{\partial}{\partial \theta_i}\log p(\mathbf{y} \mid \mathbf{X}, \mathbf{b}, \mathbf{B}) = &+ \frac{1}{2}\mathbf{M}^\top \mathbf{N}^{-1}\frac{\partial \mathbf{K}_y}{\partial \theta_i}\mathbf{N}^{-1}\mathbf{M}^\top \\
&- \frac{1}{2}\mathrm{tr}\left(\mathbf{K}_y^{-1}\frac{\partial \mathbf{K}_y}{\partial \theta_i}\right) - \frac{1}{2}\mathrm{tr}\left(\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial \theta_i}\right), \\
\frac{\partial \mathbf{A}}{\partial \theta_i} = &- \mathbf{H}\mathbf{K}_y^{-1}\frac{\partial \mathbf{K}_y}{\partial \theta_i}\mathbf{K}_y^{-1}\mathbf{H}^\top.
\end{aligned}$$

With a vague prior the marginal likelihood is

$$\begin{aligned}
\log p(\mathbf{y} \mid \mathbf{X}) = &- \frac{1}{2}\mathbf{y}^\top \mathbf{K}_y^{-1}\mathbf{y} + \frac{1}{2}\mathbf{y}^\top \mathbf{C}\mathbf{y} \\
&- \frac{1}{2}\log|\mathbf{K}_y| - \frac{1}{2}\log|\mathbf{A}_v| - \frac{n-m}{2}\log 2\pi, \\
\mathbf{A}_v = &\quad \mathbf{H}\mathbf{K}_y^{-1}\mathbf{H}^\top \\
\mathbf{C} = &\quad \mathbf{K}_y^{-1}\mathbf{H}^\top \mathbf{A}_v^{-1}\mathbf{H}\mathbf{K}_y^{-1},
\end{aligned}$$

where m is the rank of $H^\top$. Its derivative is

$$\begin{aligned}
\frac{\partial}{\partial \theta_i}\log p(\mathbf{y} \mid \mathbf{X}) = &\quad \frac{1}{2}\mathbf{y}^\top \mathbf{P}\mathbf{y} + \frac{1}{2}\left(-\mathbf{y}^\top \mathbf{P}\mathbf{G} - \mathbf{G}^\top \mathbf{P}\mathbf{y} + \mathbf{G}^\top \mathbf{P}\mathbf{G}\right) \\
&- \frac{1}{2}\mathrm{tr}\left(\mathbf{K}_y^{-1}\frac{\partial \mathbf{K}_y}{\partial \theta_i}\right) - \frac{1}{2}\mathrm{tr}\left(\mathbf{A}_v^{-1}\frac{\partial \mathbf{A}_v}{\partial \theta_i}\right), \\
\mathbf{P} &= \mathbf{K}_y^{-1}\frac{\partial \mathbf{K}_y}{\partial \theta_i}\mathbf{K}_y^{-1}, \\
\mathbf{G} &= \mathbf{H}^\top \mathbf{A}^{-1}\mathbf{H}\mathbf{K}_y^{-1}\mathbf{y},
\end{aligned} \tag{69}$$

where has been used the fact that matrices $\mathbf{K}_y^{-1}$, $\frac{\partial \mathbf{K}_y}{\partial \theta_i}$ and $\mathbf{A}_v$ are symmetric. The above expression (69) could be simplified a little further because $\mathbf{y}^\top \mathbf{P}\mathbf{G} + \mathbf{G}^\top \mathbf{P}\mathbf{y} = 2 \cdot \mathbf{y}^\top \mathbf{P}\mathbf{G}$. The use of mean functions is demonstrated in `demo_regression_meanf`.

11. Adding new features in the toolbox

As described in the previous sections, **GPstuff** is build modularly so that the full model is constructed by combining separate building blocks. Each building block, such as the covariance function or likelihood, is written in a MATLAB m-file which contains all the

functions and parameters specific to it. This construction makes it easier to add new model blocks in the package. For example, new covariance functions can be written by copying one of the existing functions to a new file and modifying all the subfunctions and parameters according to the new covariance function. With likelihoods it should be remembered that **GPstuff** assumes currently that they factorize as described in the equation (1). All the inference methods should work for log-concave likelihoods. However, likelihood functions that are not log-concave may lead to problems with Laplace approximation and EP since the conditional posterior of the latent variables may be multimodal and the diagonal matrices $\mathbf{W}$ and $\tilde{\mathbf{\Sigma}}$ may contain negative elements (Vanhatalo et al., 2009). For example, Student-t observation model leads to a likelihood, which is not log-concave, and requires some specific modifications to the Laplace approximation and EP algorithm (see Jylänki et al., 2011). Using MCMC should, however, be straightforward with non-log-concave likelihoods as well. See Appendix E for technical details of `lik`, `gpcf` and `prior` functions, and argument interface for optimisation and sampling functions.

The package is not as modular with respect to the computational techniques as to the model blocks. Usually, each inference method requires specific summaries from the model blocks. For example, the EP algorithm requires that each likelihood has a function to evaluate integrals such as $\int f_i p(y_i|f_i,\phi)N(f_i|\mu,\sigma^2)df_i$ whereas the Laplace approximation requires the Hessian of the log likelihood, $\nabla^2_{\mathbf{f}} \log p(\mathbf{y}\,|\,\mathbf{f},\phi)$. For this reason adding, for example, variational approximation would require modifications also in the likelihood functions.

12. Discussion

Broadly thinking, GPs have been researched intensively for decades and many disciplines have contributed to their study. They have not, necessarily, been called GP but, for example, spatial statistics, signal processing and filtering literature have their own naming conventions. The point of view taken in **GPstuff** comes mainly from the machine learning literature where the subject has flourished to a hot topic since late 1990's. Our aim in the **GPstuff** package has been to collect existing and create new practical tools for analyzing GP models. We acknowledge that most of the basic models and algorithms have been proposed by others. However, the implementation in **GPstuff** is unique and contains several practical solutions for computational problems that are not published elsewhere. Our own contribution on GP theory that is included in **GPstuff** is described mainly in (Vanhatalo and Vehtari, 2007, 2008, 2010; Vanhatalo et al., 2009, 2010; Jylänki et al., 2011)

Our intention has been to create a toolbox with which useful, interpretable results can be produced in a sensible time. **GPstuff** provides approximations with varying level of accuracy. The approximation to be used needs to be chosen so that it approximates well enough the essential aspects in the model. Thus, the choice is always data, model, and problem dependent. For example, MCMC methods are often praised for their asymptotic properties and seemingly easy implementation. Algorithms, such as Metropolis-Hastings, are easy to implement for most models. The problem, however, is that as a data set grows they may not give reliable results in a finite time. With GP models this problem is faced severely. For example, currently problems with over 10 000 data points would be impractical to analyse with a standard office PC using MCMC since the convergence rate and mixing of the sample chain would be too slow. For this reason it is important to have also faster, but perhaps

less accurate, methods. The choice of the method is then a compromise between time and accuracy. The inference is the fastest when using MAP estimate for the parameters and a Gaussian function for the conditional posterior. With a Gaussian likelihood, the Gaussian conditional distribution is exact and the only source of imprecision is the point estimate for the parameters. If the likelihood is other than Gaussian, the conditional distribution is an approximation, whose quality depends on how close to Gaussian the real conditional posterior is, and how well the mean and covariance are approximated. The form of the real posterior depends on many things for which reason the Gaussian approximation has to be assessed independently for every data.

One of our aims with **GPstuff** has been to provide an easy way to detect model misspecifications. This requires the model assessment tools discussed in the section 8 and an easy way to test alternative model structures. The model misfit most often relates either to the likelihood or GP prior. For example, if the data contained outliers or observations with clearly higher variance than what the Gaussian or Poisson observation model predicts, the posterior of the latent function would be highly compromised. For this reason, robust alternatives for traditional models, such as Student-t or negative binomial distribution should be easily testable. Even though the GP prior is very flexible and only the mean and covariance need to be fixed, it still contains rather heavy assumptions. For example, GP associated with the squared exponential covariance function is indefinitely mean square differentiable. This is a very strong assumption on the smoothness of the latent function. In fact it is rather peculiar how little attention other covariance functions have gained in machine learning literature. One of the reasons may be that often machine learning problems take place in high dimensional input spaces where data are enforced to lie sparsely and for which reason the possible solutions are smooth. However, the covariance function sometimes does influence the results (see e.g. Vanhatalo and Vehtari, 2008; Vanhatalo et al., 2010).

In statistics, inference in the parameters is a natural concern but in machine learning literature they are left in less attention. An indicator of this is the usual approach to maximize the marginal likelihood which implies uniform prior for the parameters. **GPstuff** provides an easy way to define priors explicitly so that people would really use them (this principle is also in line with reasoning by Gelman, 2006). We want to stress a few reasons for explicitly defining a prior. In spatial statistics, it is well known that the length-scale and magnitude are underidentifiable and the proportion σ^2/l is more important to the predictive performance than their individual values (Diggle et al., 1998; Zhang, 2004; Diggle and Ribeiro, 2007). These results are shown for Mátern class of covariance functions but according to our experiments they seem to apply for Wendland's piecewise polynomials as well. With them, the property can be taken advantage of since by giving more weight to short length-scales one favors sparser covariance matrices that are faster in computations. Other advantage is that priors make the inference problem easier by narrowing the posterior and making the parameters more identifiable. This is useful especially for MCMC methods but optimization and other integration approximations gain from the priors as well. These two reasons are rather practical. More fundamental reason is that in Bayesian statistics leaving prior undefined (meaning uniform prior) is a prior statement as well, and sometimes it may be really awkward. For example, uniform prior works very badly for the parameters of the Student-t distribution. Thus, it is better to spend some time thinking what the prior actually says than blindly use the uniform.

Much could be done to improve the inference algorithms in **GPstuff**. For example, the shape of the conditional posterior of a latent variable could be estimated more accurately with techniques described by, for example, Rue et al. (2009), Paquet et al. (2009), and Cseke and Heskes (2010) (see also Tierney and Kadane, 1986). Improvements compromise the computational speed but are essential for more reliable results, for which reason these techniques provide an important future extension for the toolbox. There are also other analytic approximations than the Laplace approximation or EP proposed in the literature. Most of these are based on some kind of variational approximation (Gibbs and Mackay, 2000; Csató and Opper, 2002; Tipping and Lawrence, 2005; Kuss, 2006; Opper and Archambeau, 2009). Laplace approximation and EP were chosen to **GPstuff** for a few reasons. They both are, in theory, straightforward to implement for any factorizable likelihood (although sometimes there are practical problems with the implementation). Laplace approximation is also fast and EP is shown to work well in many problems. Here, we have considered parametric observation models but non-parametric versions are possible as well. Examples of possible extensions to that direction are provided by Snelson et al. (2004) and Tokdar et al. (2010).

GPstuff is an ongoing project and the package will be updated whenever new functionalities are ready. The latest version of the toolbox is available from `http://becs.aalto.fi/en/research/bayes/gpstuff/`.

Acknowledgments

The research leading to **GPstuff** was funded by the Academy of Finland, Finnish Funding Agency for Technology and Innovation (TEKES), and the Graduate School in Electronics and Telecommunications and Automation (GETA). The first author also thanks the Finnish Foundation for Economic and Technology Sciences KAUTE, Finnish Cultural Foundation, and Emil Aaltonen Foundation for supporting his post graduate studies during which the code package was prepared.

The package contains pieces of code written by other people than the authors. In the Department of Biomedical Engineering and Computational Science at Aalto University these persons are (in alphabetical order): Toni Auranen, Roberto Calandra, Pasi Jylänki, Tuomas Nikoskinen, Eero Pennala, Heikki Peura, Ville Pietiläinen, Markus Siivola, Simo Särkkä and Ville Tolvanen. People outside Aalto University are (in alphabetical order): Christopher M. Bishop, Timothy A. Davis, Matthew D. Hoffman, Kurt Hornik, Dirk-Jan Kroon, Iain Murray, Ian T. Nabney, Radford M. Neal and Carl E. Rasmussen. We want to thank them all for sharing their code under a free software license.

References

Milton Abramowitz and Irene A. Stegun. *Handbook of mathematical functions.* Dover Publications, Inc., 1970.

Omar B. Ahmad, Cynthia Boschi-Pinto, Alan D. Lopez, Christopher J.L. Murray, Rafael Lozano, and Mie Inoue. Age standardization of rates: A new WHO standard. *GPE Discussion Paper Series*, 31, 2000.

M. A. Alvarez, D. Luengo, M. K. Titsias, and N. D. Lawrence. Efficient multioutput Gaussian processes through variational inducing kernels. *JMLR Workshop and conference proceedings*, 9:25–32, 2010.

Sudipto Banerjee, Alan E. Gelfand, Andrew O. Finley, and Huiyan Sang. Gaussian predictive process models for large spatial data sets. *Journal of the Royal Statistical Society B*, 70 (4):825–848, 2008.

Veronica J. Berrocal, Adrian E. Raftery, Tilmann Gneiting, and Richard C. Steed. Probabilistic weather forecasting for winter road maintenance. *Journal of the American Statistical Association*, 105(490):522–537, 2010.

Nicky Best, Sylvia Richardson, and Andrew Thomson. A comparison of Bayesian spatial models for disease mapping. *Statistical Methods in Medical Research*, 14:35–59, 2005.

M. D. Buhmann. A new class of radial basis functions with compact support. *Mathematics of Computation*, 70(233):307–318, January 2001.

Prabir Burman. A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods. *Biometrika*, 76(3):503–514, 1989.

Ole F. Christensen, Gareth O. Roberts, and Martin Sköld. Robust Markov chain Monte Carlo methods for spatial generalized linear mixed models. *Journal of Computational and Graphical Statistics*, 15:1–17, 2006.

Stefano Conti, John Paul Gosling, Jeremy Oakley, and Anthony O'Hagan. Gaussian process emulation of dynamic computer codes. *Biometrika*, 96:663–676, 2009.

Noel A. C. Cressie. *Statistics for Spatial Data*. John Wiley & Sons, Inc., 1993.

Lehel Csató and Manfred Opper. Sparse online Gaussian processes. *Neural Computation*, 14(3):641–669, 2002.

Botond Cseke and Tom Heskes. Improving posterior marginal approximations in latent Gaussian models. *JMLR Workshop and Conference Proceedings*, 9:121–128, 2010.

Timothy A. Davis. Algorithm 849: A concise sparse cholesky factorization package. *ACM Trans. Math. Softw.*, 31:587–591, December 2005.

Timothy A. Davis. *Direct Methods for Sparse Linear Systems*. SIAM, 2006.

Marc Peter Deisenroth, Carl Edward Rasmussen, and Jan Peters. Gaussian process dynamic programming. *Neurocomputing*, 72(7-9):1508–1524, 2009.

Marc Peter Deisenroth, Carl Edward Rasmussen, and Dieter Fox. Learning to control a low-cost manipulator using data-efficient reinforcement learning. In *In 9th International Conference on Robotics: Science & Systems*, 2011.

Thomas G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

P. J. Diggle, J. A. Tawn, and R. A. Moyeed. Model-based geostatistics. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 47(3):299–350, 1998.

Peter J. Diggle and Paulo J. Ribeiro. *Model-based Geostatistics*. Springer Science+Business Media, LLC, 2007.

Simon Duane, A.D. Kennedy, Brian J. Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, September 1987.

P. Elliot, Jon Wakefield, Nicola Best, and David Briggs, editors. *Spatial Epidemiology Methods and Applications*. Oxford University Press, 2001.

Bärbel Finkenstädt, Leonhard Held, and Valerie Isham. *Statistical Methods for Spatio-Temporal Systems*. Chapman & Hall/CRC, 2007.

Montserrat Fuentes and Adrian E. Raftery. Model evaluation and spatial interpolation by bayesian combination of observations with outputs from numerical models. *Biometrics*, 66:36–45, 2005.

Reinhard Furrer, Marc G. Genton, and Douglas Nychka. Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics*, 15(3): 502–523, September 2006.

G. Gaspari and S.E. Cohn. Construction of correlation functions in two and three dimensions. *Quarterly Journal of the Royal Meteorological Society*, 125(554):723–757, January 1999.

Alan E. Gelfand, D. K. Dey, and H. Chang. Model determination using predictive distributions with implementation via sampling-based methods (with discussion). In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, editors, *Bayesian Statistics 4*, pages 147–167. Oxford University Press, 1992.

Alan E. Gelfand, Peter J. Diggle, Montserrat Fuentes, and Peter Guttorp. *Handbook of Spatial Statistics*. CRC Press, 2010.

Andrew Gelman. Prior distributions for variance parameters in hierarchical models. *Bayesian Analysis*, 1(3):515–533, 2006.

Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC, second edition, 2004.

John Geweke. Bayesian inference in econometric models using Monte Carlo integration. *Econometrica*, 57(6):721–741, 1989.

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

W.R. Gilks, S. Richardson, and D.J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman & Hall, 1996.

Tilmann Gneiting. Correlation functions for atmospheric data analysis. *Quarterly Journal of the Royal Meteorological Society*, 125:2449–2464, 1999.

Tilmann Gneiting. Compactly supported correlation functions. *Journal of Multivariate Analysis*, 83:493–508, 2002.

Mohinder S. Grewal and Angus P. Andrews. *Kalman Filtering: Theory and Practice Using Matlab*. Wiley Interscience, second edition, 2001.

David A. Harville. *Matrix Algebra From a Statistician's Perspective*. Springer-Verlag, 1997.

Antti Honkela, Pei Gao, Jonatan Ropponen, Magnus Rattray, and Neil D. Lawrence. tigre: Transcription factor inference through gaussian process reconstruction of expression for bioconductor. *Bioinformatics*, 27(7):1026–1027, 2011.

Joseph G. Ibrahim, Ming-Hui Chen, and Debajyoti Sinha. *Bayesian Survival Analysis*. Springer, 2001.

Pasi Jylänki, Jarno Vanhatalo, and Aki Vehtari. Gaussian process regression with a student-t likelihood. Technical report, Department of Biomedical Engineering and Computational Science, Aalto University, 2011. arXiv:1106.4431 (accepted for publication in Journal of Machine Learning Research.

Ilkka Kalliomäki, Aki Vehtari, and Jouko Lampinen. Shape analysis of concrete aggregates for statistical quality modeling. *Machine Vision and Applications*, 16(3):197–201, 2005.

Robert E. Kass and Adrian E. Raftery. Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795, 1995.

Cari G. Kaufman, Mark J. Schervish, and Douglas W. Nychka. Covariance tapering for likelihood-based estimation in large spatial data sets. *Journal of the American Statistical Association*, 103(484):1545–1555, 2008.

Mark C. Kennedy and Anthony O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society, B*, 63(3):425–464, 2001.

Malte Kuss. *Gaussian Process Models for Robust Regression, Classification, and Reinforcement Learning*. PhD thesis, Technische Universität Darmstadt, 2006.

Malte Kuss and Carl Edward Rasmussen. Assessing approximate inference for binary Gaussian process classification. *Journal of Machine Learning Research*, 6:1679–1704, October 2005.

Neil Lawrence. Learning for larger datasets with the Gaussian process latent variable model. In M. Meila and X. Shen, editors, *Proceedings of the Eleventh International Workshop on Artificial Intelligence and Statistics*. Omnipress, 2007.

Andrew B. Lawson. *Statistical Methods in Spatial Epidemology*. John Wiley & Sons, Ltd, 2001.

Sara Martino, Rupali Akerkar, and Håvard Rue. Approximate bayesian inference for survival models. *Scandinavian Journal of Statistics*, 38:514–528, 2011.

G. Matheron. The intrinsic random functions and their applications. *Advances in Applied Probability*, 5(3):439–468, December 1973.

Mathworks. *MATLAB Getting Started Quide*. The MathWorks, Inc., fifteenth edition, 2010.

Thomas Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, Massachusetts Institute of Technology, 2001.

Jesper Møller, Anne Randi Syversveen, and Rasmus Plenge Waagepetersen. Log Gaussian Cox processes. *Scandinavian Journal of Statistics*, 25:451–482, September 1998.

Fernando A Moala and Anthony O'Hagan. Elicitation of multivariate prior distributions: a nonparametric bayesian approach. *Journal of Statistical Planning and Inference*, 140: 1635–1655, 2010.

G. Moreaux. Compactly supported radial covariance functions. *Journal of Geodecy*, 82(7): 431–443, July 2008.

J Mullahy. Specification and testing of some modified count data models. *Journal of Econometrics*, 33:341–365, 1986.

Iain Murray, Ryan Prescott Adams, and David J.C. MacKay. Elliptical slice sampling. *JMLR: Workshop and Conference Proceedings*, 9:541–548, 2010.

Ian T. Nabney. *NETLAB: Algorithms for Pattern Recognition*. Springer, 2001.

Radford Neal. Regression and classification using Gaussian process priors. In J. M. Bernardo, J. O. Berger, A. P. David, and A. P. M. Smith, editors, *Bayesian Statistics 6*, pages 475–501. Oxford University Press, 1998.

Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer, 1996.

Radford M. Neal. Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification. Technical Report 9702, Dept. of statistics and Dept. of Computer Science, University of Toronto, January 1997.

Radford M. Neal. Slice sampling. *The Annals of Statistics*, 31(3):705–767, 2003.

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

Anthony O'Hagan. Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society. Series B.*, 40(1):1–42, 1978.

Anthony O'Hagan. On outlier rejection phenomena in Bayes inference. *Journal of the Royal Statistical Society. Series B.*, 41(3):358–367, 1979.

Manfred Opper and Cédric Archambeau. The variational Gaussian approximation revisited. *Neural Computation*, 21(3):786–792, March 2009.

Manfred Opper and Ole Winther. Gaussian processes for classification: Mean-field algorithms. *Neural Computation*, 12(11):2655–2684, 2000.

Ulrich Paquet, Ole Winther, and Manfred Opper. Perturbation corrections in approximate inference: Mixture modelling applications. *Journal of Machine Learning Research*, 10: 1263–1304, 6 2009.

Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6(3):1939–1959, December 2005.

Jorma Rantonen, Satu Luoto, Aki Vehtari, Markku Hupli, Jaro Karppinen, Antti Malmivaara, and Simo Taimela. The effectiveness of two active interventions compared to self-care advice in employees with non-acute low back symptoms. a randomised, controlled trial with a 4-year follow-up in the occupational health setting. *Occupational and Environmental Medicine*, 2011.

Carl Edward Rasmussen. *Evaluations of Gaussian Processes and Other Methods for Non-linear Regression*. PhD thesis, University of Toronto, 1996.

Carl Edward Rasmussen and Hannes Nickisch. Gaussian processes for machine learning (gpml) toolbox. *Journal of Machine Learning Research*, 11:3011–3015, September 2010.

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

Stephen L. Rathbun and Noel Cressie. Asymptotic properties of estimators for the parameters of spatial inhomogeneous poisson point processes. *Advances in Applied Probability*, 26(1):122–154, March 1994.

Sylvia Richardson. Discussion to 'bayesian measures of model complexity and fit' by spiegelhalter et al. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 64(4):626–627, 2002.

Sylvia Richardson. Spatial models in epidemiological applications. In Peter J. Green, Nils Lid Hjort, and Sylvia Richardson, editors, *Highly Structured Stochastic Systems*, pages 237–259. Oxford University Press, 2003.

Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer, second edition, 2004.

Håvard Rue, Sara Martino, and Nicolas Chopin. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal statistical Society B*, 71(2):1–35, 2009.

Susan M. Sanchez and Paul J. Sanchez. Very large fractional factorials and central composite designs. *ACM Transactions on Modeling and Computer Simulation*, 15:362–377, 2005.

F. Sansò and W.-D. Schuh. Finite covariance functions. *Journal of Geodesy*, 61(4):331–347, 1987.

Mathias Seeger, Christopher K. I. Williams, and Neil Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In Christopher M. Bishop and Brendan J. Frey, editors, *Ninth International Workshop on Artificial Intelligence and Statistics*. Society for Artificial Intelligence and Statistics, 2003.

Matthias Seeger. Expectation propagation for exponential families. Technical report, Max Planck Institute for Biological Cybernetics, Tübingen, Germany, 2005.

Edward Snelson. *Flexible and Efficient Gaussian Process Models for Machine Learning*. PhD thesis, University College London, 2007.

Edward Snelson and Zoubin Ghahramani. Sparse Gaussian process using pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*. The MIT Press, 2006.

Edward Snelson and Zoubin Ghahramani. Local and global sparse Gaussian process approximations. In Marina Meila and Xiaotong Shen, editors, *Artificial Intelligence and Statistics 11*. Omnipress, 2007.

Edward Snelson, Carl Edward Rasmussen, and Zoubin Ghahramani. Warped Gaussian processes. In T. G. Diettrich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*. The MIT Press, 2004.

David J. Spiegelhalter, Nicola G. Best, Bradley P. Carlin, and Angelika van der Linde. Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society B*, 64(4):583–639, 2002.

Oliver Stegle, Sebastian V. Fallert, David J. C. MacKay, and Søren Brage. Gaussian process robust regression for noisy heart rate data. *Biomedical Engineering, IEEE Transactions on*, 55(9):2143–2151, September 2008. ISSN 0018-9294.

S. Sundararajan and S. S. Keerthi. Predictive approaches for choosing hyperparameters in gaussian processes. *Neural Computation*, 13(5):1103–1118, May 2001.

Luke Tierney and Joseph B. Kadane. Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association*, 81(393):82–86, 1986.

Michael E. Tipping and Neil D. Lawrence. Variational inference for Student-t models: Robust bayesian interpolation and generalised component analysis. *Neurocomputing*, 69: 123–141, 2005.

Michalis K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. *JMLR Workshop and Conference Proceedings*, 5:567–574, 2009.

Surya T. Tokdar and Jayanta K. Ghosh. Posterior consistency of logistic Gaussian process priors in density estimation. *Journal of Statistical Planning and Inference*, 137:34–42, 2007.

Surya T. Tokdar, Yu M. Zhu, and Jayanta K. Ghosh. Bayesian density regression with logistic Gaussian process and subspace projection. *Bayesian Analysis*, 5(2):319–344, 2010.

Jarno Vanhatalo and Aki Vehtari. Sparse log Gaussian processes via MCMC for spatial epidemiology. *JMLR Workshop and Conference Proceedings*, 1:73–89, 2007.

Jarno Vanhatalo and Aki Vehtari. Modelling local and global phenomena with sparse Gaussian processes. In David A. McAllester and Petri Myllymäki, editors, *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, pages 571–578, 2008.

Jarno Vanhatalo and Aki Vehtari. Speeding up the binary Gaussian process classification. In Peter Grünwald and Peter Spirtes, editors, *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, pages 1–9, 2010.

Jarno Vanhatalo, Pasi Jylänki, and Aki Vehtari. Gaussian process regression with student-t likelihood. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1910–1918. NIPS foundation, 2009.

Jarno Vanhatalo, Ville Pietiläinen, and Aki Vehtari. Approximate inference for disease mapping with sparse gaussian processes. *Statistics in Medicine*, 29(15):1580–1607, 2010.

Aki Vehtari and Jouko Lampinen. Bayesian model assessment and comparison using cross-validation predictive densities. *Neural Computation*, 14(10):2439–2468, 2002.

Sumio Watanabe. *Algebraic Geometry and Statistical Learning Theory*. Cambridge University Press, 2009.

Sumio Watanabe. Equations of states in singular statistical estimation. *Neural Networks*, 23(1):20–34, 2010a.

Sumio Watanabe. Asymptotic equivalence of bayes cross validation and widely applicable information criterion in singular learning theory. *Journal of Machine Learning Research*, 11:3571–3594, 2010b.

Holger Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, 4(1):389–396, December 1995.

Holger Wendland. *Scattered Data Approximation*. Cambridge University Press, 2005.

Norbert Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. MIT Press, 1949.

C. K. I. Williams and C. E. Rasmussen. Gaussian processes for regression. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 514–520. MIT Press, 1996.

Christopher K. I. Williams. Computing with infinite networks. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. The MIT Press, 1996.

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

Zongmin Wu. Compactly supported positive definite radial functions. *Advances in Computational Mathematics*, 4(1):283–292, 1995.

Hao Zhang. Inconsistent estimation and asymptotically equal interpolations in model-based geostatistics. *Journal of the American Statistical Association*, 99(465):250–261, 2004.

Appendix A. Covariance functions

In this section we summarize all the covariance functions in the GPstuff package.

Squared exponential covariance function (`gpcf_sexp`)

Probably the most widely-used covariance function is the squared exponential (SE)

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2_{\text{sexp}} \exp\left(-\frac{1}{2}\sum_{k=1}^{d}\frac{(x_{i,k}-x_{j,k})^2}{l_k^2}\right). \tag{70}$$

The length scale l_k governs the correlation scale in input dimension k and the magnitude σ^2_{sexp} the overall variability of the process. A squared exponential covariance function leads to very smooth GPs that are infinitely times mean square differentiable.

Exponential covariance function (`gpcf_exp`)

Exponential covariance function is defined as

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2_{\text{exp}} \exp\left(-\sqrt{\sum_{k=1}^{d}\frac{(x_{i,k}-x_{j,k})^2}{l_k^2}}\right). \tag{71}$$

The parameters l_k and σ^2_{exp} have similar role as with the SE covariance function. The exponential covariance function leads to very rough GPs that are not mean square differentiable.

Matérn class of covariance functions (`gpcf_maternXX`)

The Matérn class of covariance functions is given by

$$k_\nu(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2_{\text{m}}\frac{2^{1-\nu}}{\Gamma(\nu)}\left(\sqrt{2\nu}r\right)^\nu K_\nu\left(\sqrt{2\nu}r\right), \tag{72}$$

where $r = \left(\sum_{k=1}^{d}\frac{(x_{i,k}-x_{j,k})^2}{l_k^2}\right)^{1/2}$. The parameter ν governs the smoothness of the process, and K_ν is a modified Bessel function (Abramowitz and Stegun, 1970, sec. 9.6). The Matérn

covariance functions can be represent in a simpler form when ν is a half integer. The Matérn covariance functions with $\nu = 3/2$ (`gpcf_matern32`) and $\nu = 5/2$ (`gpcf_matern52`) are:

$$k_{\nu=3/2}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_{\mathrm{m}}^2 \left(1 + \sqrt{3}r\right) \exp\left(-\sqrt{3}r\right) \tag{73}$$

$$k_{\nu=5/2}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_{\mathrm{m}}^2 \left(1 + \sqrt{5}r + \frac{5r^2}{3}\right) \exp\left(-\sqrt{5}r\right). \tag{74}$$

Neural network covariance function (`gpcf_neuralnetwork`)

A neural network with suitable transfer function and prior distribution converges to a GP as the number of hidden units in the network approaches to infinity (Neal, 1996; Williams, 1996; Rasmussen and Williams, 2006). A nonstationary neural network covariance function is

$$k(\mathbf{x}_i, \mathbf{x}_j) = \frac{2}{\pi} \sin^{-1}\left(\frac{2\tilde{\mathbf{x}}_i^{\mathrm{T}} \Sigma \tilde{\mathbf{x}}_j}{(1 + 2\tilde{\mathbf{x}}_i^{\mathrm{T}} \Sigma \tilde{\mathbf{x}}_i)(1 + 2\tilde{\mathbf{x}}_j^{\mathrm{T}} \Sigma \tilde{\mathbf{x}}_j)}\right), \tag{75}$$

where $\tilde{\mathbf{x}} = (1, x_1, \ldots, x_d)^{\mathrm{T}}$ is an input vector augmented with 1. $\Sigma = \mathrm{diag}(\sigma_0^2, \sigma_1^2, \ldots, \sigma_d^2)$ is a diagonal weight prior, where σ_0^2 is a variance for bias parameter controlling the functions offset from the origin. The variances for weight parameters are $\sigma_1^2, \ldots, \sigma_d^2$, and with small values for weights, the neural network covariance function produces smooth and rigid looking functions. The larger values for the weight variances produces more flexible solutions.

Constant covariance function (`gpcf_constant`)

Perhaps the simplest covariance function is the constant covariance function

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 \tag{76}$$

with variance parameter σ^2. This function can be used to implement the constant term in the dot-product covariance function (Rasmussen and Williams, 2006) reviewed below.

Linear covariance function (`gpcf_linear`)

The linear covariance function is

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^{\mathrm{T}} \Sigma \mathbf{x}_j \tag{77}$$

where the diagonal matrix $\Sigma = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_D^2)$ contains the prior variances of the linear model coefficients. Combining this with the constant function above we can form covariance function Rasmussen and Williams (2006), which calls a dot-product covariance function

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 + \mathbf{x}_i^{\mathrm{T}} \Sigma \mathbf{x}_j. \tag{78}$$

Piecewise polynomial functions (`gpcf_ppcsX`)

The piecewise polynomial functions are the only compactly supported covariance functions (see section 7) in GPstuff. There are four of them with the following forms

$$k_{pp0}(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2(1-r)_+^j \tag{79}$$

$$k_{pp1}(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2(1-r)_+^{j+1}\left((j+1)r+1\right) \tag{80}$$

$$k_{pp2}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sigma^2}{3}(1-r)_+^{j+2}((j^2+4j+3)r^2+(3j+6)r+3) \tag{81}$$

$$\begin{aligned} k_{pp3}(\mathbf{x}_i, \mathbf{x}_j) =& \frac{\sigma^2}{15}(1-r)_+^{j+3}((j^3+9j^2+23j+15)r^3+ \\ & (6j^2+36j+45)r^2+(15j+45)r+15) \end{aligned} \tag{82}$$

where $j = \lfloor d/2 \rfloor + q + 1$. These functions correspond to processes that are $2q$ times mean square differentiable at the zero and positive definite up to the dimension d (Wendland, 2005). The covariance functions are named `gpcf_ppcs0`, `gpcf_ppcs1`, `gpcf_ppcs2`, and `gpcf_ppcs3`.

Rational quadratic covariance function (`gpcf_rq`)

The rational quadratic (RQ) covariance function (Rasmussen and Williams, 2006)

$$k_{\mathrm{RQ}}(\mathbf{x}_i, \mathbf{x}_j) = \left(1 + \frac{1}{2\alpha}\sum_{k=1}^{d}\frac{(x_{i,k}-x_{j,k})^2}{l_k^2}\right)^{-\alpha} \tag{83}$$

can be seen as a scale mixture of squared exponential covariance functions with different length-scales. The smaller the parameter $\alpha > 0$ is the more diffusive the length-scales of the mixing components are. The parameter $l_k > 0$ characterizes the typical length-scale of the individual components in input dimension k.

Periodic covariance function (`gpcf_periodic`)

Many real world systems exhibit periodic phenomena, which can be modelled with a periodic covariance function. One possible construction (Rasmussen and Williams, 2006) is

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\sum_{k=1}^{d}\frac{2\sin^2(\pi(x_{i,k}-x_{j,k})/\gamma)}{l_k^2}\right), \tag{84}$$

where the parameter γ controls the inverse length of the periodicity and l_k the smoothness of the process in dimension k.

Product covariance function (`gpcf_product`)

A product of two or more covariance functions, $k_1(\mathbf{x}, \mathbf{x}') \cdot k_2(\mathbf{x}, \mathbf{x}')$..., is a valid covariance function as well. Combining covariance functions in a product form can be done with `gpcf_prod`, for which the user can freely specify the covariance functions to be multiplied with each other from the collection of covariance functions implemented in GPstuff.

Categorical covariance function (`gpcf_cat`)

Categorical covariance function `gpcf_cat` returns correlation 1 if input values are equal and 0 otherwise.

$$k(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 1 & \text{if } \mathbf{x}_i - \mathbf{x}_j = 0 \\ 0 & \text{otherwise} \end{cases} \tag{85}$$

Categorical covariance function can be combined with other covariance functions using `gpcf_prod`, for example, to produce hierarchical models.

Appendix B. Observation models

Here, we summarize all the observation models in GPstuff. Most of them are implemented in files `lik_*` which reminds that at the inference step they are considered likelihood functions.

Gaussian (`lik_gaussian`)

The i.i.d Gaussian noise with variance σ^2 is

$$\mathbf{y} \,|\, \mathbf{f}, \sigma^2 \sim N(\mathbf{f}, \sigma^2 \mathbf{I}). \tag{86}$$

Student-t (`lik_t`, `lik_gaussiansmt`)

The Student-t observation model (implemented in `lik_t`) is

$$\mathbf{y} \,|\, \mathbf{f}, \nu, \sigma_t \sim \prod_{i=1}^{n} \frac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)\sqrt{\nu\pi}\sigma_t} \left(1 + \frac{(y_i - f_i)^2}{\nu \sigma_t^2}\right)^{-(\nu+1)/2}, \tag{87}$$

where ν is the degrees of freedom and σ the scale parameter. The scale mixture version of the Student-t distribution is implemented in `lik_gaussiansmt` and it is parametrized as

$$y_i | f_i, \alpha, U_i \sim N(f_i, \alpha U_i) \tag{88}$$

$$U_i \sim \text{Inv-}\chi^2(\nu, \tau^2), \tag{89}$$

where each observation has its own noise variance αU_i (Neal, 1997; Gelman et al., 2004).

Logit (`lik_logit`)

The logit transformation gives the probability for y_i of being 1 or -1 as

$$p_{\text{logit}}(y_i | f_i) = \frac{1}{1 + \exp(-y_i f_i)}. \tag{90}$$

Probit (`lik_probit`)

The probit transformation gives the probability for y_i of being 1 or -1 as

$$p_{\text{probit}}(y_i | f_i) = \Phi(y_i f(\mathbf{x}_i)) = \int_{-\infty}^{y_i f(\mathbf{x}_i)} N(z|0,1) dz. \tag{91}$$

Poisson (`lik_poisson`)

The Poisson observation model with expected number of cases $\mathbf{e}$ is

$$\mathbf{y} \,|\, \mathbf{f}, \mathbf{e} \sim \prod_{i=1}^{n} \text{Poisson}(y_i | \exp(f_i) e_i). \tag{92}$$

Negative-Binomial (`lik_negbin`)

The negative-binomial is a robustified version of the Poisson distribution. It is parametrized

$$\mathbf{y} \,|\, \mathbf{f}, \mathbf{e}, r \sim \prod_{i=1}^{n} \frac{\Gamma(r + y_i)}{y_i! \Gamma(r)} \left(\frac{r}{r + \mu_i} \right)^r \left(\frac{\mu_i}{r + \mu_i} \right)^{y_i} \tag{93}$$

where $\mu_i = \mathbf{e} \exp(f_i)$, r is the dispersion parameter governing the variance, e_i is the expected number of cases and y_i is positive integer telling the observed count.

Binomial (`lik_binomial`)

The binomial observation model with the probability of success $p_i = \exp(f_i)/(1 + \exp(f_i))$ is

$$\mathbf{y} \,|\, \mathbf{f}, \mathbf{z} \sim \prod_{i=1}^{N} \frac{z_i!}{y_i!(z_i - y_i)!} p_i^{y_i} (1 - p_i)^{(z_i - y_i)}. \tag{94}$$

Here, z_i denotes the number of trials and y_i is the number of successes.

Appendix C. Priors

This appendix lists all the priors implemented in the GPstuff package.

Gaussian prior (`prior_gaussian`)

The Gaussian distribution is parametrized as

$$p(\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2} (\theta - \mu)^2 \right) \tag{95}$$

where μ is a location parameter and σ^2 is a scale parameter.

Log-Gaussian prior (`prior_loggaussian`)

The log-Gaussian distribution is parametrized as

$$p(\theta) = \frac{1}{\theta\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2} (\log(\theta) - \mu)^2 \right) \tag{96}$$

where μ is a location parameter and σ^2 is a scale parameter.

Laplace prior (`prior_laplace`)

The Laplace distribution is parametrized as

$$p(\theta) = \frac{1}{2\sigma} \exp\left(-\frac{|\theta - \mu|}{\sigma}\right) \tag{97}$$

where μ is a location parameter and $\sigma > 0$ is a scale parameter.

Student-t prior (`prior_t`)

The Student-t distribution is parametrized as

$$p(\theta) = \frac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)\sqrt{\nu\pi\sigma^2}} \left(1 + \frac{(\theta-\mu)^2}{\nu\sigma^2}\right)^{-(\nu+1)/2} \tag{98}$$

where μ is a location parameter, σ^2 is a scale parameter and $\nu > 0$ is the degrees of freedom.

Square root Student-t prior (`prior_sqrtt`)

The square root Student-t distribution is parametrized as

$$p(\theta^{1/2}) = \frac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)\sqrt{\nu\pi\sigma^2}} \left(1 + \frac{(\theta-\mu)^2}{\nu\sigma^2}\right)^{-(\nu+1)/2} \tag{99}$$

where μ is a location parameter, σ^2 is a scale parameter and $\nu > 0$ is the degrees of freedom.

Scaled inverse-χ^2 prior (`prior_sinvchi2`)

The scaled inverse-χ^2 distribution is parametrized as

$$p(\theta) = \frac{(\nu/2)^{\nu/2}}{\Gamma(\nu/2)} (s^2)^{\nu/2} \theta^{-(\nu/2+1)} e^{-\nu s^2/(2\theta)} \tag{100}$$

where s^2 is a scale parameter and $\nu > 0$ is the degrees of freedom parameter.

Gamma prior (`prior_gamma`)

The gamma distribution is parametrized as

$$p(\theta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \theta^{\alpha-1} e^{-\beta\theta} \tag{101}$$

where $\alpha > 0$ is a shape parameter and $\beta > 0$ is an inverse scale parameter.

Inverse-gamma prior (`prior_invgamma`)

The inverse-gamma distribution is parametrized as

$$p(\theta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \theta^{-(\alpha+1)} e^{-\beta/\theta} \tag{102}$$

where $\alpha > 0$ is a shape parameter and $\beta > 0$ is a scale parameter.

Uniform prior (`prior_unif`)

The uniform prior is parametrized as

$$p(\theta) \propto 1. \tag{103}$$

Square root uniform prior (`prior_sqrtunif`)

The square root uniform prior is parametrized as

$$p(\theta^{1/2}) \propto 1. \tag{104}$$

Log-uniform prior (`prior_logunif`)

The log-uniform prior is parametrized as

$$p(\log(\theta)) \propto 1. \tag{105}$$

Log-log-uniform prior (`prior_loglogunif`)

The log-log-uniform prior is parametrized as

$$p(\log(\log(\theta))) \propto 1. \tag{106}$$

Appendix D. Transformation of hyperparameters

The inference on the parameters of covariance functions is conducted mainly transformed space. Most of ten used transformation is log-transformation, which has the advantage that the parameter space $(0, \infty)$ is transformed into $(-\infty, \infty)$. The change of parametrization has to be taken into account in the evaluation of the probability densities of the model. If parameter θ with probability density $p_\theta(\theta)$ is transformed into the parameter $w = f(\theta)$ with equal number of components, the probability density of w is given by

$$p_w(w) = |J| p_\theta(f^{-1}(w)), \tag{107}$$

where J is the Jacobian of the transformation $\theta = f^{-1}(w)$. The parameter transformations are discussed shortly, for example, in Gelman et al. (2004)[p. 24].

Due to the log transformation $w = \log(\theta)$ transformation the probability densities $p_\theta(\theta)$ are changed to the densities

$$p_w(w) = |J| p_\theta(\exp(w)) = |J| p_\theta(\theta), \tag{108}$$

where the Jacobian is $J = \frac{\partial \exp(w)}{\partial w} = \exp(w) = \theta$. Now, given Gaussian observation model (see Section 3.1) the posterior of w can be written as

$$p_w(w|\mathcal{D}) \propto p(\mathbf{y}|\mathbf{X}, \theta) p(\theta|\gamma)\theta, \tag{109}$$

which leads to energy function

$$\begin{aligned} E(w) &= -\log p(\mathbf{y}|\mathbf{X}, \theta) - \log p(\theta|\gamma) - \log(|\theta|). \\ &= E(\theta) - \log(\theta), \end{aligned}$$

where the absolute value signs are not shown explicitly around θ because it is strictly positive. Thus, the log transformation just adds term $-\log\theta$ in the energy function.

The inference on w requires also the gradients of an energy function $E(w)$. These can be obtained easily with the chain rule

$$\begin{aligned}\frac{\partial E(w)}{\partial w} &= \frac{\partial}{\partial \theta}\left[E(\theta) - \log(|J|)\right]\frac{\partial \theta}{\partial w} \\ &= \left[\frac{\partial E(\theta)}{\partial \theta} - \frac{\partial \log(|J|)}{\partial \theta}\right]\frac{\partial \theta}{\partial w} \\ &= \left[\frac{\partial E(\theta)}{\partial \theta} - \frac{1}{|J|}\frac{\partial |J|}{\partial \theta}\right] J. \end{aligned} \tag{110}$$

Here we have used the fact that the last term, derivative of θ with respect to w, is the same as the Jacobian $J = \frac{\partial \theta}{\partial w} = \frac{\partial f^{-1}}{\partial w}$. Now in the case of log transformation the Jacobian can be replaced by θ and the gradient is gotten an easy expression

$$\frac{\partial E(w)}{\partial w} = \frac{\partial E(\theta)}{\partial \theta}\theta - 1. \tag{111}$$

Appendix E. Developer appendix

This section provides additional description of `lik`, `gpcf` and `prior` functions, and argument interface for optimisation and sampling functions.

E.1 `lik` functions

New likelihoods can be added by copying and modifying one of the existing `lik` functions. We use `lik_negbin` as an example of log-concave likelihood and `lik_t` as an example of non-log-concave likelihood. Note that adding a new non-log-concave likelihood is more challenging as the posterior of the latent values may be multimodal, which makes it more difficult to implement stable Laplace and EP methods (see, e.g., Vanhatalo et al., 2009; Jylänki et al., 2011).

E.1.1 `lik_negbin`

inputParser is used to make argument checks and set some default values. If the new likelihood does not have parameters, see for example, `lik_poisson`.

```
  ip=inputParser;
  ip.FunctionName = 'LIK_NEGBIN';
  ip.addOptional('lik', [], @isstruct);
  ip.addParamValue('disper',10, @(x) isscalar(x) && x>0);
  ip.addParamValue('disper_prior',prior_logunif(), @(x) isstruct(x) || isempty(x));
  ip.parse(varargin{:});
  lik=ip.Results.lik;
...
```

Function handles to the subfunctions provide object-oriented behaviour (at time when GPstuff project was started, real classes in Matlab were too inefficient). If some of the subfunctions have not been implemented, corresponding function handles should not be defined.

```
  if init
    % Set the function handles to the subfunctions
    lik.fh.pak = @lik_negbin_pak;
    lik.fh.unpak = @lik_negbin_unpak;
    lik.fh.lp = @lik_negbin_lp;
    lik.fh.lpg = @lik_negbin_lpg;
    lik.fh.ll = @lik_negbin_ll;
    lik.fh.llg = @lik_negbin_llg;
    lik.fh.llg2 = @lik_negbin_llg2;
    lik.fh.llg3 = @lik_negbin_llg3;
    lik.fh.tiltedMoments = @lik_negbin_tiltedMoments;
    lik.fh.siteDeriv = @lik_negbin_siteDeriv;
    lik.fh.predy = @lik_negbin_predy;
    lik.fh.predprcty = @lik_negbin_predprcty;
    lik.fh.invlink = @lik_negbin_invlink;
    lik.fh.recappend = @lik_negbin_recappend;
  end

end
```

`lik_negbin_pak` and `lik_negbin_unpak` functions are used to support generic optimization and sampling functions, which assume that the parameters are presented as a vector. Parameters which have empty prior ([]) are ignored. These subfunctions are mandatory (even if there are no parameters).

`lik_negbin_lp` and `lik_negbin_lpg` compute the log prior density of the parameters and its gradient with respect to parameters. Parameters which have empty prior ([]) are ignored. These subfunctions are needed if there are likelihood parameters.

`lik_negbin_ll` and `lik_negbin_llg` compute the log likelihood and its gradient with respect to parameters and latents. Parameters which have empty prior ([]) are ignored. These subfunctions are mandatory.

`lik_negbin_llg2` computes second gradients of the log likelihood. This subfunction is needed when using Laplace approximation or EP for inference with non-Gaussian likelihoods.

`lik_negbin_llg3` computes third gradients of the log likelihood. This subfunction is needed when using Laplace approximation for inference with non-Gaussian likelihoods.

`lik_negbin_tiltedMoments` returns the marginal moments for the EP algorithm. This subfunction is needed when using EP for inference with non-Gaussian likelihoods.

`lik_negbin_siteDeriv` evaluates the expectation of the gradient of the log likelihood term with respect to the likelihood parameters for EP. This subfunction is needed when using EP for inference with non-Gaussian likelihoods and there are likelihood parameters.

`lik_negbin_predy` returns the predictive mean, variance and density of y. This subfunction is needed when computing posterior predictive distributions for future observations.

`lik_negbin_predprcty` returns the percentiles of predictive density of y. This subfunction is needed when using function `gp_predprcty`.

`lik_negbin_init_negbin_norm` is a private function for `lik_negbin`. It returns function handle to a function evaluating Negative-Binomial * Gaussian which is used for evaluating (likelihood * cavity) or (likelihood * posterior). This subfunction is needed by subfunctions `lik_negbin_tiltedMoments`, `lik_negbin_siteDeriv` and `lik_negbin_predy`. Note that this is not needed for those likelihoods for which integral over the likelihood times Gaussian is computed using other than quadrature integration.

`lik_negbin_invlink` returns values of inverse link function. This subfunction is needed when using function `gp_predprctmu`.

`lik_negbin_recappend` This subfunction is needed when using MCMC sampling (`gp_mc`).

E.1.2 `lik_t`

`lik_t` includes some additional subfunctions useful for non-log-concave likelihoods.

`lik_t_tiltedMoments2` returns the marginal moments for the EP algorithm. This subfunction is needed when using robust-EP for inference with non-Gaussian likelihoods.

`lik_t_siteDeriv2` evaluates the expectation of the gradient of the log likelihood term with respect to the likelihood parameters for EP. This subfunction is needed when using robust-EP for inference with non-Gaussian likelihoods and there are likelihood parameters.

`lik_t_optimizef` function to optimize the latent variables with EM algorithm. This subfunction is needed when using likelihood specific optimization method for mode finding in the Laplace algorithm.

E.2 `gpcf` functions

New covariance functions can be added by copying and modifying one of the existing `gpcf` functions. We use `gpcf_sexp` as an example.

E.2.1 `gpcf_sexp`

inputParser is used to make argument checks and set some default values. If the new covariance function does not have parameters, corresponding lines can be removed (see, e.g., `lik_cat`).

```
ip=inputParser;
ip.FunctionName = 'GPCF_SEXP';
ip.addOptional('gpcf', [], @isstruct);
ip.addParamValue('magnSigma2',0.1, @(x) isscalar(x) && x>0);
ip.addParamValue('lengthScale',1, @(x) isvector(x) && all(x>0));
ip.addParamValue('metric',[], @isstruct);
ip.addParamValue('magnSigma2_prior', prior_logunif(), ...
                 @(x) isstruct(x) || isempty(x));
ip.addParamValue('lengthScale_prior',prior_t(), ...
                 @(x) isstruct(x) || isempty(x));
ip.addParamValue('selectedVariables',[], @(x) isempty(x) || ...
                 (isvector(x) && all(x>0)));
```

```
  ip.parse(varargin{:});
  gpcf=ip.Results.gpcf;
...
```

Optional 'metric' can be used to replace default simple euclidean metric. Currently it is possible to use delta distance or group covariates to have common lengthScale parameters. Other potential metrics could be, for example, Manhattan or distance matrix based metrics. The metric function is called only if it has been explicitly set. This adds some additional code branches to the code, but avoids extra overhead in computation when using simple euclidean metric case.

Function handles to the subfunctions provide object-oriented behaviour (at time when GPstuff project was started, real classes in Matlab were too inefficient). If some of the subfunctions have not been implemented, corresponding function handles should not be defined.

```
  if init
    % Set the function handles to the subfunctions
    gpcf.fh.pak = @gpcf_sexp_pak;
    gpcf.fh.unpak = @gpcf_sexp_unpak;
    gpcf.fh.lp = @gpcf_sexp_lp;
    gpcf.fh.lpg= @gpcf_sexp_lpg;
    gpcf.fh.cfg = @gpcf_sexp_cfg;
    gpcf.fh.cfdg = @gpcf_sexp_cfdg;
    gpcf.fh.cfdg2 = @gpcf_sexp_cfdg2;
    gpcf.fh.ginput = @gpcf_sexp_ginput;
    gpcf.fh.ginput2 = @gpcf_sexp_ginput2;
    gpcf.fh.ginput3 = @gpcf_sexp_ginput3;
    gpcf.fh.ginput4 = @gpcf_sexp_ginput4;
    gpcf.fh.cov = @gpcf_sexp_cov;
    gpcf.fh.trcov  = @gpcf_sexp_trcov;
    gpcf.fh.trvar  = @gpcf_sexp_trvar;
    gpcf.fh.recappend = @gpcf_sexp_recappend;
  end
```

`gpcf_sexp_pak` and `gpcf_sexp_unpak` functions are used to support generic optimization and sampling functions, which assume that parameters are presented as a vector. Parameters which have empty prior ([]) are ignored. These subfunctions are mandatory (even if there are no parameters).

`gpcf_sexp_lp` and `gpcf_sexp_lpg` compute the log prior density of the parameters and its gradient with respect to parameters. Parameters which have empty prior ([]) are ignored. These subfunctions are mandatory.

`gpcf_sexp_cov` evaluates covariance matrix between two input vectors. This is a mandatory subfunction used for example in prediction and energy computations.

`gpcf_sexp_trcov` evaluates training covariance matrix of inputs. This is a mandatory subfunction used for example in prediction and energy computations. If available, `gpcf_sexp_trcov` uses faster C-implementation `trcov` for covariance computation (see lin-

uxCsource/trcov.c). `trcov` includes C-code implementation for all currently available covariance functions. If `trcov` is not available, `gpcf_sexp_trcov` uses M-code computation.

`gpcf_sexp_trvar` evaluates training variance vector. This is a mandatory subfunction used for example in prediction and energy computations.

`gpcf_sexp_cfg` evaluates gradient of covariance function with respect to the parameters. `gpcf_sexp_cfg` has four different calling syntaxes. First one is a mandatory used in gradient computations. Second and third are needed when using sparse approximations (e.g. FIC). Fourth one is needed when using memory save option in `gp_set` (without memory save option, all matrix derivatives with respect to all covariance parameters are computed at once, which may take lot of memory if n is large and there are many parameters).

`gpcf_sexp_cfdg` evaluates gradient of covariance function, of which has been taken partial derivative with respect to x, with respect to parameters. This subfunction is needed when using derivative observations.

`gpcf_sexp_cfdg2` evaluates gradient of covariance function, of which has been taken partial derivative with respect to both input variables x, with respect to parameters. This subfunction is needed when using derivative observations.

`gpcf_sexp_ginput` evaluates gradient of covariance function with respect to x. This subfunction is needed when computing gradients with respect to inducing inputs in sparse approximations.

`gpcf_sexp_ginput2` evaluates gradient of covariance function with respect to both input variables x and x2. This subfunction is needed when computing gradients with respect to both input variables x and x2 (in same dimension). This subfunction is needed when using derivative observations.

`gpcf_sexp_ginput3` evaluates gradient of covariance function with respect to both input variables x and x2. This subfunction is needed when computing gradients with respect to both input variables x and x2 (in different dimensions). This subfunction is needed when using derivative observations.

`gpcf_sexp_ginput4` evaluates gradient of covariance function with respect to x. Simplified and faster version of `sexp_ginput`, returns full matrices. This subfunction is needed when using derivative observations.

`gpcf_sexp_recappend` is needed when using MCMC sampling (`gp_mc`).

E.3 `prior` functions

New priors can be added by copying and modifying one of the existing `prior` functions. We use `prior_t` as an example.

inputParser is used to make argument checks and set some default values. If the new prior does not have parameters, corresponding lines can be removed (see, e.g., `prior_unif`).

```
ip=inputParser;
ip.FunctionName = 'PRIOR_T';
ip.addOptional('p', [], @isstruct);
ip.addParamValue('mu',0, @(x) isscalar(x));
ip.addParamValue('mu_prior',[], @(x) isstruct(x) || isempty(x));
ip.addParamValue('s2',1, @(x) isscalar(x) && x>0);
ip.addParamValue('s2_prior',[], @(x) isstruct(x) || isempty(x));
```

```
  ip.addParamValue('nu',4, @(x) isscalar(x) && x>0);
  ip.addParamValue('nu_prior',[], @(x) isstruct(x) || isempty(x));
  ip.parse(varargin{:});
  p=ip.Results.p;
...
```

Function handles to the subfunctions provide object-oriented behaviour (at time when GPstuff project was started, real classes in Matlab were too inefficient). If some of the subfunctions have not been implemented, corresponding function handles should not be defined.

```
  if init
    % set functions
    p.fh.pak = @prior_t_pak;
    p.fh.unpak = @prior_t_unpak;
    p.fh.lp = @prior_t_lp;
    p.fh.lpg = @prior_t_lpg;
    p.fh.recappend = @prior_t_recappend;
  end
```

`prior_t_pak` and `prior_t_unpak` functions are used to support generic optimization and sampling functions, which assume that parameters are presented as a vector. Parameters which have empty prior ([]) are ignored. These subfunctions are mandatory (even if there are no parameters).

`prior_t_lp` and `prior_t_lpg` compute the log prior density of the parameters and its gradient with respect to parameters. Parameters which have empty prior ([]) are ignored. These subfunctions are mandatory.

`prior_t_recappend` This subfunction is needed when using MCMC sampling (`gp_mc`).

E.4 Optimisation functions

Adding new optimisation functions is quite easy as covariance function and likelihood parameters and inducing inputs can be optimised using any optimisation function using same syntax as, for example, `fminunc` by Mathworks or `fminscg` and `fminlbfgs` included in GPstuff.

Syntax is

```
  X = FMINX(FUN, X0, OPTIONS)
```

where FUN accepts input X and returns a scalar function value F and its scalar or vector gradient G evaluated at X, X0 is initial value and OPTIONS is option structure.

Then new optimisation algorithm can be used with code

```
gp=gp_optim(gp,x,y,'opt',opt,'optimf',@fminx);
```

E.5 Other inference methods

Adding new Monte Carlo sampling functions is more complicated. `gp_mc` function is used to allow the use of different sampling methods for covariance function parameters, likelihood parameters and latent values with one function call. Thus adding new sampling method requires modification of `gp_mc`. Note also the subfunctions `gpmc_e` and `gpmc_g`, which return energy (negative log posterior density) and its gradient while handling `infer_params` option properly. See `gp_mc` for further information.

Adding new analytic approximation, for example, variational Bayes approach, would require major effort implementing many inference related functions. To start see functions with name starting `gpla_` or `gpep_` and `gp_set`.