The GPML Toolbox version 3.5

Carl Edward Rasmussen & Hannes Nickisch

July 13, 2015

Abstract


The strength of the function lies in its flexibility, simplicity and extensibility. The function is flexible as firstly it allows specification of the properties of the GP through definition of mean function and covariance functions. Secondly, it allows specification of different inference procedures, such as e.g. exact inference and Expectation Propagation (EP). Thirdly it allows specification of likelihood functions e.g. Gaussian or Laplace (for regression) and e.g. cumulative Logistic (for classification). Simplicity is achieved through a single function and compact code. Extensibility is ensured by modular design allowing for easy addition of extension for the already fairly extensive libraries for inference methods, mean functions, covariance functions and likelihood functions.

This document is a technical manual for a developer containing many details. If you are not yet familiar with the GPML toolbox, the [user documentation](http://gpml.sourceforge.net/) and examples therein are a better way to get started.
## Contents

1. Gaussian Process Training and Prediction ........................................... 3

2. The gp Function ................................................................................. 5

3. Inference Methods ............................................................................... 10
   3.1 Exact Inference ........................................................................... 11
   3.2 Laplace's Approximation ............................................................... 12
   3.3 Expectation Propagation ............................................................... 12
   3.4 Kullback Leibler Divergence Minimisation ........................................ 13
   3.5 Variational Bayes ......................................................................... 14
   3.6 FITC Approximations ................................................................ 15
   3.7 Grid Approximations .................................................................. 15

4. Likelihood Functions ........................................................................... 17
   4.1 Prediction ..................................................................................... 17
   4.2 Interface ..................................................................................... 18
   4.3 Implemented Likelihood Functions .................................................. 20
   4.4 Usage of Implemented Likelihood Functions ..................................... 21
   4.5 Compatibility Between Likelihoods and Inference Methods ............... 22
   4.6 Gaussian Likelihood ..................................................................... 22
      4.6.1 Exact Inference ..................................................................... 24
      4.6.2 Laplace's Approximation ......................................................... 24
      4.6.3 Expectation Propagation .......................................................... 24
      4.6.4 Variational Bayes .................................................................. 24
   4.7 Warped Gaussian Likelihood ............................................................ 25
   4.8 Gumbel Likelihood ....................................................................... 26
   4.9 Laplace Likelihood ....................................................................... 26
   4.10 Student’s t Likelihood ................................................................. 29
   4.11 Cumulative Logistic Likelihood ...................................................... 29
   4.12 GLM Likelihoods: Poisson, Weibull, Gamma, Exponential, Inverse Gaussian and Beta
      4.12.1 Inverse Link Functions ............................................................ 30
      4.12.2 Poisson Likelihood ................................................................. 31
      4.12.3 Weibull Likelihood ................................................................. 32
      4.12.4 Gamma Likelihood ................................................................. 32
      4.12.5 Exponential Likelihood ............................................................ 33
      4.12.6 Inverse Gaussian Likelihood .................................................... 33
      4.12.7 Beta Likelihood .................................................................... 33

5. Mean Functions .................................................................................. 34
   5.1 Interface ...................................................................................... 34
   5.2 Implemented Mean Functions ....................................................... 35
   5.3 Usage of Implemented Mean Functions ........................................... 35

6. Covariance Functions ......................................................................... 37
   6.1 Interface ...................................................................................... 37
   6.2 Implemented Covariance Functions ............................................... 40
   6.3 Usage of Implemented Covariance Functions ................................... 41

7. Hyperpriors ...................................................................................... 42
   7.1 Interface ...................................................................................... 42
7.2 Implemented Hyperpriors ......................................................... 44
7.3 Usage of Implemented Hyperpriors ........................................... 44
1 Gaussian Process Training and Prediction

The gpml toolbox contains a single user function gp described in section 2. In addition there are a number of supporting structures and functions which the user needs to know about, as well as an internal convention for representing the posterior distribution, which may not be of direct interest to the casual user.

Inference Methods: An inference method is a function which computes the (approximate) posterior, the (approximate) negative log marginal likelihood and its partial derivatives w.r.t. the hyperparameters, given a model specification (i.e., GP mean and covariance functions and a likelihood function) and a data set. Inference methods are discussed in section 3. New inference methods require a function providing the desired inference functionality and possibly extra functionality in the likelihood functions applicable.

Hyperparameters: The hyperparameters is a struct controlling the properties of the model, i.e., the GP mean and covariance function and the likelihood function. The hyperparameters is a struct with the three fields mean, cov and lik, each of which is a vector. The number of elements in each field must agree with number of hyperparameters in the specification of the three functions they control (below). If a field is either empty or non-existent it represents zero hyperparameters. When working with FITC approximate inference, the inducing inputs xu can also be treated as hyperparameters for some common stationary covariances.

Hyperparameter Prior Distributions: When optimising the marginal likelihood w.r.t. hyperparameters, it is sometimes useful to softly constrain the hyperparameters by means of prior knowledge. A prior is a probability distribution over individual or a group of hyperparameters, section 7.

Likelihood Functions: The likelihood function specifies the form of the likelihood of the GP model and computes terms needed for prediction and inference. For inference, the required properties of the likelihood depend on the inference method, including properties necessary for hyperparameter learning, section 4.

Mean Functions: The mean function is a cell array specifying the GP mean. It computes the mean and its derivatives w.r.t. the part of the hyperparameters pertaining to the mean. The cell array allows flexible specification and composition of mean functions, discussed in section 5. The default is the zero function.

Covariance Functions: The covariance function is a cell array specifying the GP covariance function. It computes the covariance and its derivatives w.r.t. the part of the hyperparameters pertaining to the covariance function. The cell array allows flexible specification and composition of covariance functions, discussed in section 6.

Inference methods, see section 3, compute (among other things) an approximation to the posterior distribution of the latent variables $f_i$ associated with the training cases, $i = 1, \ldots, n$. This approximate posterior is assumed to be Gaussian, and is communicated via a struct post with the fields post.alpha, post.sW and post.L. Often, starting from the Gaussian prior $p(f) = \mathcal{N}(f|m, K)$ the approximate posterior admits the form

$$q(f|D) = \mathcal{N}(f|m + K\alpha, V = (K^{-1} + W)^{-1}),$$

where $W$ diagonal with $W_{ii} = s_i^2$. (1)

In such cases, the entire posterior can be computed from the two vectors post.alpha and post.sW; the inference method may optionally also return $L = \text{chol(diag(s)K diag(s) + I})$. If on the other hand the posterior doesn’t admit the above form, then post.L returns the matrix
\[ L = -(K + W^{-1})^{-1} \] (and post.sW is unused). In addition, a sparse representation of the posterior may be used, in which case the non-zero elements of the post.alpha vector indicate the active entries.
2 The gp Function

The gp function is typically the only function the user would directly call.

\[
\begin{align*}
\text{gp} & \equiv \\
\text{function } & [\text{varargout}] = \text{gp}(\text{hyp}, \text{inf}, \text{mean}, \text{cov}, \text{lik}, x, y, xs, ys) \\
\end{align*}
\]

It offers facilities for training the hyperparameters of a GP model as well as predictions at unseen inputs as detailed in the following help.

\[
\begin{align*}
\text{gp function help} & \equiv \\
1 & \% Gaussian Process inference and prediction. The gp function provides a \\
2 & \% flexible framework for Bayesian inference and prediction with Gaussian \\
3 & \% processes for scalar targets, i.e. both regression and binary \\
4 & \% classification. The prior is Gaussian process, defined through specification \\
5 & \% of its mean and covariance function. The likelihood function is also \\
6 & \% specified. Both the prior and the likelihood may have hyperparameters \\
7 & \% associated with them. \\
8 & \% \\
9 & \% Two modes are possible: training or prediction: if no test cases are \\
10 & \% supplied, then the negative log marginal likelihood and its partial \\
11 & \% derivatives w.r.t. the hyperparameters is computed; this mode is used to fit \\
12 & \% the hyperparameters. If test cases are given, then the test set predictive \\
13 & \% probabilities are returned. Usage: \\
14 & \% \\
15 & \% training: [nlZ, dnlZ] = \text{gp}(\text{hyp}, \text{inf}, \text{mean}, \text{cov}, \text{lik}, x, y); \\
16 & \% prediction: [ymu, ys2, fmu, fs2] = \text{gp}(\text{hyp}, \text{inf}, \text{mean}, \text{cov}, \text{lik}, x, y, xs); \\
17 & \% or: [ymu, ys2, fmu, fs2, lp] = \text{gp}(\text{hyp}, \text{inf}, \text{mean}, \text{cov}, \text{lik}, x, y, xs, ys); \\
18 & \% \\
19 & \% where: \\
20 & \% \\
21 & \% hyp \quad \text{struct of column vectors of mean/cov/lik hyperparameters} \\
22 & \% inf \quad \text{function specifying the inference method} \\
23 & \% mean \quad \text{prior mean function} \\
24 & \% cov \quad \text{prior covariance function} \\
25 & \% lik \quad \text{likelihood function} \\
26 & \% x \quad \text{n by D matrix of training inputs} \\
27 & \% y \quad \text{column vector of length n of training targets} \\
28 & \% xs \quad \text{ns by D matrix of test inputs} \\
29 & \% ys \quad \text{column vector of length nn of test targets} \\
30 & \% \\
31 & \% nlZ \quad \text{returned value of the negative log marginal likelihood} \\
32 & \% dnlZ \quad \text{struct of column vectors of partial derivatives of the negative} \\
33 & \% \log \text{ marginal likelihood w.r.t. mean/cov/lik hyperparameters} \\
34 & \% ymu \quad \text{column vector (of length ns) of predictive output means} \\
35 & \% ys2 \quad \text{column vector (of length ns) of predictive output variances} \\
36 & \% fmu \quad \text{column vector (of length ns) of predictive latent means} \\
37 & \% fs2 \quad \text{column vector (of length ns) of predictive latent variances} \\
38 & \% lp \quad \text{column vector (of length ns) of log predictive probabilities}
\end{align*}
\]
Depending on the number of input parameters, \( \text{gp} \) knows whether it is operated in training or in prediction mode. The high level structure of the code is as follows. After some initialisations, we perform inference and decide whether test set predictions are needed or only the result of the inference is demanded.

If the number of input arguments is incorrect, we echo a minimalist usage and return.

Set some useful default values for empty arguments, and convert \( \text{inf} \) and \( \text{lik} \) to function handles and mean and cov to cell arrays if necessary. Initialize variables.
Check that the sizes of the hyperparameters supplied in hyp match the sizes expected. The three parts hyp.mean, hyp.cov and hyp.lik are checked separately, and define empty entries if they don’t exist.

```
7a  ⟨check hyperparameters⟩≡ (6b)
1   if ~isfield(hyp,'mean'), hyp.mean = []; end % check the hyp specification
2   if eval(feval(mean{:})) ~= numel(hyp.mean)
3       error('Number of mean function hyperparameters disagree with mean function')
4   end
5   if ~isfield(hyp,'cov'), hyp.cov = []; end
6   if eval(feval(cov{:})) ~= numel(hyp.cov)
7       error('Number of cov function hyperparameters disagree with cov function')
8   end
9   if ~isfield(hyp,'lik'), hyp.lik = []; end
10  if eval(feval(lik{:})) ~= numel(hyp.lik)
11     error('Number of lik function hyperparameters disagree with lik function')
12 end
```

Inference is performed by calling the desired inference method inf. In training mode, we accept a failure of the inference method (and issue a warning), since during hyperparameter learning, hyperparameters causing a numerical failure may be attempted, but the minimize function may gracefully recover from this. During prediction, failure of the inference method is an error.

```
7b  ⟨inference⟩≡ (5a)
1   try % call the inference method
2     % issue a warning if a classification likelihood is used in conjunction with
3     % labels different from +1 and -1
4   if strcmp(lstr,'likErf') || strcmp(lstr,'likLogistic')
5     if ~isstruct(y)
6         uy = unique(y);
7         if any( uy"="+1 & uy"="-1 )
8             warning('You try classification with labels different from \{+1,-1\}')
9         end
10     end
11 end
12 if nargin>7 % compute marginal likelihood and its derivatives only if needed
13   if isstruct(y)
14       post = y; % reuse a previously computed posterior approximation
15   else
16       post = feval(inf{:}, hyp, mean, cov, lik, x, y);
17   end
18 else
19   if nargout<=1
20       [post nlZ] = feval(inf{:}, hyp, mean, cov, lik, x, y); dn1Z = {};
21   else
22       [post nlZ dn1Z] = feval(inf{:}, hyp, mean, cov, lik, x, y);
23   end
24 end
25 catch
26   msgstr = lasterr;
```
if nargin>7, error('Inference method failed [%s]', msgstr); else
dnlZ = struct('cov',0*hyp.cov, 'mean',0*hyp.mean, 'lik',0*hyp.lik);
varargout = {NaN, dnlZ}; return % continue with a warning
end
end

We copy the already computed negative log marginal likelihood to the first output argument, and if desired report its partial derivatives w.r.t. the hyperparameters if running in inference mode.

Predictions are computed in a loop over small batches to avoid memory problems for very large test sets.

\[ \text{compute test predictions} \]
\[ \begin{align*}
\alpha &= \text{post.alpha}; \quad L = \text{post.L}; \quad sW = \text{post.sW}; \\
\text{if issparse} (\alpha) &\quad \% \text{handle things for sparse representations} \\
\text{nz} &= \alpha \neq 0; \quad \% \text{determine nonzero indices} \\
\text{if issparse} (L) &\quad L = \text{full}(L(nz,nz)); \quad \% \text{convert L and sW if necessary} \\
\text{if issparse} (sW) &\quad sW = \text{full}(sW(nz)); \quad \% \text{non-sparse representation} \\
\text{if isempty} (L) &\quad \% \text{in case L is not provided, we compute it}
\end{align*} \]
\[ \begin{align*}
K &= \text{feval(cov{:}, hyp.cov, x(nz,:));} \\
L &= \text{chol(eye(sum(nz)) + sW*sW'.*K);}
\end{align*} \]

In every iteration of the above loop, we compute the predictions for all test points of the batch.

\[ \text{make predictions} \]
\[ \begin{align*}
kss &= \text{feval(cov{:}, hyp.cov, xs(id,:), 'diag');} & \% \text{self-variance} \\
\text{if strcmp} (\text{cstr}, 'covFITC') &\quad \% \text{cross-covariances} \\
Ks &= \text{feval(cov{:}, hyp.cov, x, xs(id,:));} \quad \% \text{res indep. of x} \\
\text{else} &\quad \% \text{avoid computation} \\
ms &= \text{feval(mean{:}, hyp.mean, x(nz,:), xs(id,:));} \\
N &= \text{size(alpha,2)}; \quad \% \text{number of alphas (usually 1; more in case of sampling)} \\
Fmu &= \text{repmat}(ms,1,N) + Ks'*full(alpha(nz,:)); \quad \% \text{conditional mean f|f} \\
fmu(id) &= \text{sum}(Fmu,2)/N; \quad \% \text{predictive means} \\
\text{if Lchol} &\quad \% \text{L contains chol decomp => use Cholesky parameters (alpha,sW,L)} \\
V &= L'\cdot(\text{repmat}(sW,1,length(id)).*Ks); \\
fS2(id) &= kss - \text{sum}(V.*V,1); \quad \% \text{predictive variances} \\
\text{else} &\quad \% \text{L is not triangular => use alternative parametrisation}
\end{align*} \]
if isnumeric(L), LKs = L*Ks; else LKs = L(Ks); end  % matrix or callback
fs2(id) = kss + sum(Ks.*LKs,1)'; % predictive variances
end
fs2(id) = max(fs2(id),0); % remove numerical noise i.e. negative variances
Fs2 = repmat(fs2(id),1,N); % we have multiple values in case of sampling
if nargin<9
    [Lp, Ymu, Ys2] = feval(lik{:},hyp.lik,[], Fmu(:,),Fs2(:));
else
    Ys = repmat(ys(id),1,N);
    [Lp, Ymu, Ys2] = feval(lik{:},hyp.lik,Ys(:,),Fmu(:,),Fs2(:));
end
lp(id) = sum(reshape(Lp, [],N),2)/N; % log probability; sample averaging
ymu(id) = sum(reshape(Ymu, [],N),2)/N; % predictive mean ys|y and ..
ys2(id) = sum(reshape(Ys2, [],N),2)/N; % .. variance
3 Inference Methods

Inference methods are responsible for computing the (approximate) posterior \( \text{post} \), the (approximate) negative log marginal likelihood \( \text{nlZ} \) and its partial derivatives \( \text{dnlZ} \) w.r.t. the hyperparameters \( \text{hyp} \). The arguments to the function are hyperparameters \( \text{hyp} \), mean function \( \text{mean} \), covariance function \( \text{cov} \), likelihood function \( \text{lik} \) and training data \( x \) and \( y \). Several inference methods are implemented and described this section.

\[
\langle \text{infMethods.m} \rangle \equiv
\]

1 % Inference methods: Compute the (approximate) posterior for a Gaussian process.
2 % Methods currently implemented include:
3 %
4 % infExact        Exact inference (only possible with Gaussian likelihood)
5 % infLaplace      Laplace’s Approximation
6 % infEP          Expectation Propagation
7 % infVB          Variational Bayes Approximation
8 % infKL          Kullback-Leibler optimal Approximation
9 % infFITC        Large scale regression with approximate covariance matrix
10 % infFITC_Laplace Large scale inference with approximate covariance matrix
11 % infFITC_EP     Large scale inference with approximate covariance matrix
12 % infMCMC        Markov Chain Monte Carlo and Annealed Importance Sampling
13 %
14 % We offer two samplers.
15 %
16 %  - hmc: Hybrid Monte Carlo
17 %  - ess: Elliptical Slice Sampling
18 %
19 % No derivatives w.r.t. to hyperparameters are provided.
20 %
21 % infLOO         Leave-One-Out predictive probability and Least-Squares Approxim.
22 % infPrior       Perform inference with hyperparameter prior.
23 %
24 % The interface to the approximation methods is the following:
25 %
26 % function [post nlZ dnlZ] = inf...(hyp, cov, lik, x, y)
27 %
28 % where:
29 %
30 % hyp is a struct of hyperparameters
31 % cov is the name of the covariance function (see covFunctions.m)
32 % lik is the name of the likelihood function (see likFunctions.m)
33 % x is a \( n \) by \( D \) matrix of training inputs
34 % y is a (column) vector (of size \( n \)) of targets
35 % nlZ is the returned value of the negative log marginal likelihood
36 % dnlZ is a (column) vector of partial derivatives of the negative
37 % log marginal likelihood w.r.t. each hyperparameter
38 % post struct representation of the (approximate) posterior containing
39 % alpha is a (sparse or full column vector) containing \( \text{inv}(K)\ast(\text{mu}-\text{m}) \),
40 % where \( K \) is the prior covariance matrix, \( m \) the prior mean,
41 % and \( \text{mu} \) the approx posterior mean
42 % sW is a (sparse or full column) vector containing diagonal of \( \text{sqrt}(W) \)
43 % the approximate posterior covariance matrix is \( \text{inv}(\text{inv}(K)+\text{W}) \)
44 % L is a (sparse or full) matrix, \( L = \text{chol}(sW\ast K \ast sW+\text{eye}(n)) \)
45 %
46 % Usually, the approximate posterior to be returned admits the form
47 % \( N(\text{mu}+K\ast\text{alpha}, W=\text{inv}(\text{inv}(K)+W)) \), where \( \text{alpha} \) is a vector and \( W \) is diagonal;
48 % if not, then \( L \) contains instead \( -\text{inv}(K+\text{inv}(W)) \), and \( sW \) is unused.

11
Not all inference methods are compatible with all likelihood functions, e.g., exact inference is only possible with Gaussian likelihood. In order to perform inference, each method needs various properties of the likelihood functions, section 4.

3.1 Exact Inference

For Gaussian likelihoods, GP inference reduces to computing mean and covariance of a multivariate Gaussian which can be done exactly by simple matrix algebra. The program inf/infExact.m does exactly this. If it is called with a likelihood function other than the Gaussian, it issues an error. The Gaussian posterior $q(f|D) = \mathcal{N}(f|\mu, V)$ is exact.

```plaintext
function [post nlZ dnlZ] = infExact(hyp, mean, cov, lik, x, y)
  % Exact inference for a GP with Gaussian likelihood. Compute a parametrization
  % of the posterior, the negative log marginal likelihood and its derivatives
  % w.r.t. the hyperparameters. See also "help infMethods".
  %
  % See also INFMETHODS.M.
  if iscell(lik), likstr = lik{1}; else likstr = lik; end
  if ~ischar(likstr), likstr = func2str(likstr); end
  if ~strcmp(likstr,'likGauss') % NOTE: no explicit call to likGauss
    error('Exact inference only possible with Gaussian likelihood');
  end
  [n, D] = size(x);
  K = feval(cov{:}, hyp.cov, x); % evaluate covariance matrix
  m = feval(mean{:}, hyp.mean, x); % evaluate mean vector
  sn2 = exp(2*hyp.lik); % noise variance of likGauss
  if sn2<1e-6 % very tiny sn2 can lead to numerical trouble
    L = chol(K+sn2*eye(n)); sl = 1; % Cholesky factor of covariance with noise
    pL = -solve_chol(L,eye(n)); % L = -inv(K+inv(sW^2))
  else
    L = chol(K/sn2+eye(n)); sl = sn2; % Cholesky factor of B
    pL = L; % L = chol(eye(n)+sW*sW'.*K)
  end
  alpha = solve_chol(L,y-m)/sl;
  post.alpha = alpha; % return the posterior parameters
  post.sW = ones(n,1)/sqrt(sn2); % sqrt of noise precision vector
  post.L = pL;
  if nargout>1 % do we want the marginal likelihood?
    nlZ = (y-m)'*alpha/2 + sum(log(diag(L))) + n*log(2*pi*sl)/2; % -log marg lik
    if nargout>2 % do we want derivatives?
      dnlZ = hyp; % allocate space for derivatives
      Q = solve_chol(L,eye(n))/sl - alpha*alpha'; % precompute for convenience
    end
  end
end
```

11
for i = 1:numel(hyp.cov)
    dnlZ.cov(i) = sum(sum(Q.*feval(cov{:}, hyp.cov, x, [], i)))/2;
end

dnlZ.lik = sn2*trace(Q);
for i = 1:numel(hyp.mean)
    dnlZ.mean(i) = -feval(mean{:}, hyp.mean, x, i)'*alpha;
end
end
end

3.2 Laplace’s Approximation

For differentiable likelihoods, Laplace’s approximation, approximates the posterior by a Gaussian centered at its mode and matching its curvature inf/infLaplace.m.

More concretely, the mean of the posterior $q(f|D) = \mathcal{N}(f|\mu, V)$ is – defining $\ell_i(f_i) = \ln p(y_i|f_i)$ and $\ell(f) = \sum_{i=1}^n \ell_i(f_i)$ – given by

$$\mu = \arg \min_f \phi(f), \quad \text{where} \quad \phi(f) = \frac{1}{2} (f - m)^\top K^{-1} (f - m) - \ell(f) \leq -\ln[p(f)p(y|f)],$$

which we abbreviate by $\mu \leftarrow \mathcal{L}(\ell)$. The curvature $\frac{\partial^2 \phi}{\partial f^2} = K^{-1} + W$ with $W_{ij} = -\frac{\partial^2}{\partial f_i \partial f_j} \ln p(y_i|f_i)$ serves as precision for the Gaussian posterior approximation $V = (K^{-1} + W)^{-1}$ and the marginal likelihood $Z = \int p(f)p(y|f)df$ is approximated by $Z \approx Z_{LA} = \int \tilde{\phi}(f)df$ where we use the 2nd order Taylor expansion at the mode $\mu$ given by $\tilde{\phi}(f) = \phi(\mu) + \frac{1}{2} (f - \mu)^\top V^{-1} (f - \mu) \approx \phi(f)$.

Laplace’s approximation needs derivatives up to third order for the mode fitting procedure (Newton method)

$$d_k = \frac{\partial^k}{\partial f^k} \log p(y|f), \quad k = 0, 1, 2, 3$$

and

$$d_k = \frac{\partial}{\partial \nu_i} \frac{\partial^k}{\partial f^k} \log p(y|f), \quad k = 0, 1, 2$$

evaluated at the latent location $f$ and observed value $y$. The likelihood calls (see section 4)

- $[d0, d1, d2, d3] = \text{lik}(hyp, y, f, [], 'infLaplace')$

and

- $[d0, d1, d2] = \text{lik}(hyp, y, f, [], 'infLaplace', i)$

return exactly these values.

3.3 Expectation Propagation

The basic idea of Expectation Propagation (EP) as implemented in inf/infEP.m. is to replace the non-Gaussian likelihood terms $p(y_i|f_i)$ by Gaussian functions $t(f_i; \nu_i, \tau_i) = \exp(\nu_i f_i - \frac{1}{2} \tau_i f_i^2)$ and to adjust the natural parameters $\nu_i, \tau_i$ such that the following identity holds:

$$\frac{1}{Z_{t,i}} \int f^k q_{-i}(f) \cdot t(f; \nu_i, \tau_i) df = \frac{1}{Z_{p,i}} \int f^k q_{-i}(f) \cdot p(y_i|f) df, \quad k = 1, 2$$
with the so-called cavity distributions $q_{-i}(f) = N(f|m, K) \prod_{j \neq i} t(f_j; \nu_j, \tau_j) \propto N(f|m, V) / t(f_i; \nu_i, \tau_i)$
eq i\right) \propto N(f|m, V) / t(f_i; \nu_i, \tau_i)$
eq i\right)$ equal to the posterior divided by the $i$th Gaussian approximation function and the two normalisers $Z_{t,i} = \int q_{-i}(f) \cdot t(f_i; \nu_i, \tau_i) df$ and $Z_{p,i} = \int q_{-i}(f) \cdot p(y_i|f_i) df$. The moment matching corresponds to minimising the following local KL-divergence 

$$\nu_i, \tau_i = \arg \min_{\nu, \tau} KL[q_{-i}(f)p(y_i|f_i)/Z_{p,i}\|q_{-i}(f)t(f_i; \nu, \tau)/Z_{t,i}].$$

In order to apply the moment matching steps in a numerically safe way, EP requires the derivatives of the expectations w.r.t. the Gaussian mean parameter $\mu$

$$d_k = \frac{\partial^k}{\partial \mu^k} \log \int p(y|f)N(f|\mu, \sigma^2) df, \quad k=0, 1, 2$$

and the $i$th likelihood hyperparameter $\rho_i$

$$d = \frac{\partial}{\partial \rho_i} \log \int p(y|f)N(f|\mu, \sigma^2) df$$

which can be obtained by the likelihood calls (see section 4)

- $[d_0, d_1, d_2] = lik(hyp, y, \mu, \sigma^2, 'infEP')$

and

- $d = lik(hyp, y, \mu, \sigma^2, 'infEP', i)$.

### 3.4 Kullback Leibler Divergence Minimisation

Another well known approach to approximate inference implemented inf/infKL.m in attempts to directly find the closest Gaussian $q(f|\mathcal{D}) = N(f|m, V)$ to the exact posterior $p(f|\mathcal{D})$ w.r.t. to some proximity measure or equivalently to maximise a lower bound $Z(\mu, V)$ to the marginal likelihood $Z$ as described in Nickisch & Rasmussen [Approximations for Binary Gaussian Process Classification][2], JMLR, 2008. In particular, one minimises $KL(N(f|m, V)||p(f|\mathcal{D}))$ which amounts to minimising $-\ln Z(\mu, V)$ as defined by:

$$-\ln Z = -\ln \int p(f)p(y|f)df = -\ln \int q(f|\mathcal{D}) \frac{p(f)}{q(f|\mathcal{D})} p(y|f) df$$

\[\leq \int q(f|\mathcal{D}) \ln \frac{q(f|\mathcal{D})}{p(f)} df - \int q(f|\mathcal{D}) \ln p(y|f) df =: -\ln Z(\mu, V)\]

$$= KL(N(f|m, V)||N(f|m, K)) - \sum_{i=1}^n \int N(f_i|m_i, \nu_i) \ln p(y_i|f_i) df_i, \quad \nu_i = [V]_{ii}$$

$$= \frac{1}{2} \left( tr(VK^{-1} - I) - \ln |VK^{-1}| \right) + \frac{1}{2} (\mu - m)^\top K^{-1}(\mu - m) - \sum_{i=1}^n t^{KL}(\mu_i, v_{ii})$$

where $t^{KL}(\mu_i) = \int N(f_i|m_i, v_{ii}) \ell_i(f_i) df_i$ is the convolution of the log likelihood $\ell_i$ with the Gaussian $N$ and $v = dg(V)$. Equivalently, one can view $t^{KL}$ as a smoothed version of $\ell$ with univariate smoothing kernel $N$.

From Challis & Barber [Concave Gaussian Variational Approximations for Inference in Large Scale Bayesian Linear Models][3], AISTATS, 2011 we know that the mapping $(\mu, L) \mapsto -\ln Z(\mu, L \cdot L)$ is
jointly convex whenever the likelihoods \( f_i \mapsto \mathbb{P}(y_i|f_i) \) are log concave. In particular, this implies that every \((\mu_i, s_i) \mapsto -\ell_{\text{KL}}(\mu_i, s_i^2)\) is jointly convex.

We use an optimisation algorithm similar to EP (section 3.3) where we minimise the local KL-divergence the other way round \( \mu_i, s_i = \arg \min_{\mu, s} \text{KL}[\mathbb{N}(f|\mu, s^2)\|q_{\mathcal{L}}(f)p(y_i|f_i)/\mathcal{Z}_{\mathcal{P},i}] \). This view was brought forward by Tom Minka [Convex Divergence measures and message passing], MSR-TR, 2005. The KL minimisation constitutes a jointly convex 2d optimisation problem solved by \( \kappa_{\text{min}} \) using a scaled Newton approach which is included as a sub function in \( \text{inf/infKL.m} \). The smoothed likelihood \( \ell_{\text{KL}}(\mu_i, \nu_{i1}) \) is implemented as a meta likelihood in \( \text{likKL} \); it uses Gaussian-Hermite quadrature to compute the required integrals. Note that – as opposed to EP – Gaussian-Hermite quadrature is appropriate since we integrate against the \( \ln \mathbb{P}(y_i|f_i) \) (which can be well approximated by a polynomial) instead of \( \mathbb{P}(y_i|f_i) \) itself. The algorithm is – again unlike EP – provably convergent for log-concave likelihoods (e.g. \text{likGauss}, \text{likLaplace}, \text{likSech2}, \text{likLogistic}, \text{likPoisson}) since it can be regarded as coordinate descent with guaranteed decrease in the objective in every step. Due to the complex update computations, \( \text{infKL} \) can be quite slow although it has the same \( \mathcal{O}(n^3) \) asymptotic complexity as EP and Laplace.

### 3.5 Variational Bayes

One can drive the bounding even further by means of local quadratic lower bounds to the log likelihood \( \ell(f) = \ln p(y|f) \). Suppose that we use a super-Gaussian likelihood \( p(y|f) \) i.e. likelihoods that can be lower bounded by Gaussians of any width \( w \) (e.g. \text{likLaplace}, \text{likT}, \text{likLogistic}, \text{likSech2}). Formally, that means that there are \( b, z \in \mathbb{R} \) such that

\[
\rho(f) = \ln p(y|f) - zf
\]

is symmetric and \( \sqrt{f} \mapsto \rho(f) \) is a convex function for all \( f \geq 0 \). As a result, we obtain the following exact representation of the likelihood

\[
\ell(f) = \ln p(y|f) = \max_{w > 0} \left( (b + wz)f - \frac{wz^2}{2} - \frac{1}{2} h(\gamma) \right),
\]

which can be derived by convex duality and assuming the likelihoods to be super-Gaussian. Details can be found in papers by Palmer et al. [Variational EM Algorithms for Non-Gaussian Latent Variable Models], NIPS, 2006 and Nickisch & Seeger [Convex Variational Bayesian Inference for Large Scale Generalized Linear Models], ICML, 2009.

The bottom line is that we can treat the variational bounding as a sequence of Laplace approximations with the “variational Bayes” log likelihood

\[
\ell^{\text{VB}}(f_i) = \ell(g_i) + b_i(f_i - g_i), \quad g = \text{sgn}(f - z) \odot \sqrt{(f - z)^2 + v + z}
\]

instead of the usual likelihood \( \ell(f_i) = \ln p(y_i|f_i) \) i.e. we solve \( \mathbf{u} \leftarrow \mathcal{L}(\ell^{\text{VB}}) \) instead of \( \mathbf{u} \leftarrow \mathcal{L}(\ell) \). See section 3.2. In the code of \( \text{inf/infVB.m} \), the likelihood is implemented in the function \( \text{likVB} \).

At the end, the optimal value of \( W \) can be obtained analytically via \( w_i = |b_i - \ell'(g_i)|/|g_i - z_i| \).

For the minimisation in \( \text{inf/infVB.m} \), we use a provably convergent double loop algorithm, where in the inner loop a nonlinear least squares problem (convex for log-concave likelihoods) is solved using \( \text{inf/infLaplace.m} \) such that \( \mathbf{u} \leftarrow \mathcal{L}(\ell^{\text{VB}}) \) and in the outer loop, we compute \( v \leftarrow \text{dg}((\mathbf{K}^{-1} + W)^{-1}) \). The only requirement to the likelihood function is that it returns the values \( z \) and \( b \) required by the bound which are delivered by the call (see section 4)

- \([b,z] = \text{lik}(\text{hyp}, y, [], \text{ga}, \text{’infVB’})\)
The negative marginal likelihood upper bound \(-\ln Z_{VB}\) is obtained by integrating the prior times
the exact representation of the likelihood

\[
p(y|f) = \max_{\gamma > 0} q(y|f, \gamma), \quad q(y|f, \gamma) = N(f|\nu, \gamma) \exp \left( -\frac{h(\gamma)}{2} - \frac{\nu^2}{2\gamma} \right) \sqrt{2\pi \gamma}, \quad \gamma = \frac{1}{w}, \quad \nu = b\gamma + z
\]

w.r.t. the latent variables \(f\) yielding

\[
-\ln Z_{VB} = -\ln \int N(f|m, K) \prod_{i=1}^{n} q_i(y_i|f_i, \gamma_i) df
\]

\[
= -\ln N(m|\nu, K + \Gamma) + \frac{1}{2} \left( h(\gamma) - \nu^T \nu^2 - 1^T \ln 2\gamma \right).
\]

### 3.6 FITC Approximations

One of the main problems with GP models is the high computational load for inference computations. In a setting with \(n\) training points \(x\), exact inference with Gaussian likelihood requires \(O(n^3)\) effort; approximations like Laplace of EP consist of a sequence of \(O(n^3)\) operations.

There is a line of research with the goal to alleviate this burden by using approximate covariance functions \(\tilde{k}\) instead of \(k\). A review is given by Candela and Rasmussen [A Unifying View of Sparse Gaussian Process Regression], JMLR, 2005. One basic idea in those approximations is to work with a set of \(m\) inducing inputs \(u\) with a reduced computational load of \(O(nm^2)\). In the following, we will provide a rough idea of the FITC approximation used in the toolbox. Let \(K\) denote the \(n \times n\) covariance matrix between the training points \(x\), \(K_u\) the \(m \times n\) covariance matrix between the \(n\) training points and the \(m\) inducing points, and \(K_{uu}\) the \(m \times m\) covariance matrix between the \(m\) inducing points. The FITC approximation to the covariance is given by

\[
K \approx \tilde{K} = Q + G, \quad G = \text{diag}(g), \quad g = \text{diag}(K - Q), \quad Q = K_u^T Q_u^{-1} K_u, \quad Q_{uu} = K_{uu} + \sigma_u^2 I,
\]

where \(\sigma_u\) is the noise from the inducing inputs. Note that \(\tilde{K}\) and \(K\) have the same diagonal elements \(\text{diag}(\tilde{K}) = \text{diag}(K)\); all off-diagonal elements are the same as for \(Q\). Internally, the necessary covariance evaluations are performed by a meta covariance function \(\text{cov/covFITC.m}\). The toolbox offers FITC versions for regression with Gaussian likelihood \(\text{inf/infFITC.m}\), as well as for Laplace’s approximation \(\text{inf/infFITC_Laplace.m}\) and expectation propagation \(\text{inf/infFITC_EP.m}\).

The user can decide whether to treat the inducing inputs \(u\) as fixed or as hyperparameters. The latter is possible for a number of common stationary covariance functions (and also all three FITC inference methods \(\text{inf/infFITC*.m}\)) and allows to adjust the inducing inputs \(u\) w.r.t. the marginal likelihood. As detailed in the documentation of \(\text{inf/infFITC*.m}\), \(u\) is treated as fixed if it is passed as the 2nd parameter of \(\text{covFITC(cov, xu, ...)}\). If the hyperparameter structure \(\text{hyp}\) contains a field \(\text{hyp.xu}\) in inference method calls like \(\text{infFITC(hyp, ...)}\) or inference/prediction calls like \(\text{gp(hyp, @infFITC*, ...)}\) the inducing inputs \(u\) are treated as hyperparameters and can be optimised.

### 3.7 Grid Approximations

Another way to bring down computational costs is to take advantage of (partial) grid structure present in the training points \(x\). For example, in geostatistics or image processing, the training data \(x \in \mathbb{R}^{n \times D}\) could be a complete 2d lattice of size \(n_1 \times n_2\) as given by the axes \(x_1 \in \mathbb{R}^{n_1}, x_2 \in \mathbb{R}^{n_2}\) so that \(n = n_1 \cdot n_2, D = 2\) and \(x = [\text{vec}(x_1 1^T), \text{vec}(1x_2^T)]\). In general, a \(p\)-dimensional grid \(x_g \in \mathbb{R}^{N \times D}\) is specified by a set of axis matrices \(\{x_i \in \mathbb{R}^{n_i \times D_i}\}_{i=1..p}\) so that \(N = \prod_{i=1}^{p} n_i\) and \(D = \sum_{i=1}^{p} D_i\) where the axes do not need to be 1d nor do their components need to be sorted. As a consequence,
\( x_g \) represents a Cartesian product of its axes \( x_g = x_1 \times x_2 \times \ldots \times x_p \). The \texttt{cov/covGrid.m} covariance function represents a Kronecker product covariance matrix

\[
K = K_p \otimes \ldots \otimes K_2 \otimes K_1
\]

whose factorisation structure is given by the grid \( x_g \). The gain in computational efficiency is due to the fact that matrix-vector product, determinant, inverse and eigenvalue computations decompose so that the overall cost of \( O(N^3) \) turns into \( O(\sum_{i=1}^p n_i^3) \). Internally, we use a meta covariance function \texttt{cov/covGrid.m} to represent the Kronecker covariance matrix and a Gaussian regression inference method \texttt{inf/infGrid.m}. We also support incomplete grids where \( n < N \). A good starting point is Yunus Saatçi’s PhD thesis [Scalable Inference for Structured Gaussian Process Models], University of Cambridge, 2011. For incomplete grids, we use the extensions by Wilson et. al, [Fast Kernel Learning for Multidimensional Pattern Extrapolation], NIPS, 2014 where conjugate gradients and a determinant approximations are used. See \texttt{doc/demoGrid.m} for an illustration. There is also an extension to non-Gaussian likelihoods by Seth Flaxman [Fast Kronecker inference in Gaussian processes with non-Gaussian likelihoods].
4 Likelihood Functions

A likelihood function \( p_\rho(y|f) \) (with hyperparameters \( \rho \)) is a conditional density \( \int p_\rho(y|f)dy = 1 \) defined for scalar latent function values \( f \) and outputs \( y \). In the GPML toolbox, we use iid. likelihoods \( p_\rho(y|f) = \prod_{i=1}^{n} p_\rho(y_i|f_i) \). The approximate inference engine does not explicitly distinguish between classification and regression likelihoods: it is fully generic in the likelihood allowing to use a single code in the inference step.

Likelihood functionality is needed both during inference and while predicting.

4.1 Prediction

A prediction at \( x_* \) conditioned on the data \( D = (X,y) \) (as implemented in \texttt{gpml}) consists of the predictive mean \( \mu_{y_*} \) and variance \( \sigma_{y_*}^2 \), which are computed from the the latent marginal moments \( \mu_{f_*}, \sigma_{f_*}^2 \), i.e. the Gaussian marginal approximation \( N(f_*|\mu_{f_*},\sigma_{f_*}^2) \) via

\[
p(y_*|D, x_*) = \int p(y_*|f_*)p(f_*|D, x_*)df_* \approx \int p(y_*|f_*)N(f_*|\mu_{f_*},\sigma_{f_*}^2)df_*.
\] (3)

The moments are given by \( \mu_{y_*} = \int y_*p(y_*|D, x_*)dy_* \) and \( \sigma_{y_*}^2 = \int (y_* - \mu_{y_*})^2p(y_*|D, x_*)dy_* \). The likelihood call

- \([lp, ymu, ys2] = \text{lik(hyp, [], fmu, fs2})\)

does exactly this. Evaluation of the logarithm of \( p_{y_*} = p(y_*|D, x_*) \) for values \( y_* \) can be done via

- \([lp, ymu, ys2] = \text{lik(hyp, y, fmu, fs2})\)

where \( lp \) contains the number \( \ln p_{y_*} \).

Using the moments of the likelihood \( \mu(f_*) = \int y_*p(y_*|f_*)dy_* \) and \( \sigma^2(f_*) = \int (y_* - \mu(f_*))^2p(y_*|f_*)dy_* \) we obtain for the predictive moments the following (exact) expressions

\[
\mu_{y_*} = \int \mu(f_*)p(f_*|D, x_*)df_*, \quad \text{and}
\]

\[
\sigma_{y_*}^2 = \int \left[ \sigma^2(f_*) + (\mu(f_*) - \mu_{y_*})^2 \right] p(f_*|D, x_*)df_*.
\]

1. The binary case is simple since \( y_* \in \{-1, +1\} \) and \( 1 = p_{y_*} + p_{-y_*} \). Using \( \pi_* = p_{+1} \), we find

\[
p_{y_*} = \begin{cases} 
\pi_* & y_* = +1 \\
1 - \pi_* & y_* = -1 
\end{cases}
\]

\[
\mu_{y_*} = \sum_{y_* = \pm 1} y_* p(y_*|D, x_*) = 2 \cdot \pi_* - 1 \in [-1, 1], \quad \text{and}
\]

\[
\sigma_{y_*}^2 = \sum_{y_* = \pm 1} (y_* - \mu_{y_*})^2 p(y_*|D, x_*) = 4 \cdot \pi_*(1 - \pi_*) \in [0, 1].
\]

2. The continuous case for homoscedastic likelihoods depending on \( r_* = y_* - f_* \) only and having noise variance \( \sigma^2(f_*) = \sigma_n^2 \) is also simple since the identity \( p(y_*|f_*) = p(y_* - f_*|0) \) allows to substitute \( y_* \leftarrow y_* + f_* \) yielding \( \mu(f_*) = f_* + \int y_* p(y_*|0)dy_* \) and assuming \( \int y_* p(y_*|0)dy_* = 0 \) we arrive at

\[
\mu_{y_*} = \mu_{f_*}, \quad \text{and}
\]

\[
\sigma_{y_*}^2 = \sigma_{f_*}^2 + \sigma_n^2.
\]
3. The generalised linear model (GLM) case is also feasible. Evaluation of the predictive distribution is done by quadrature

\[ p(y_s) = \int p(y_s|f_s)p(f_s|D, x_s)df_s \approx \int p(y_s|f_s)N(f_s|\mu_{f_s}, \sigma^2_{f_s})df_s. \]

For GLMs the mean is given by \( \mu(f_s) = g(f_s) \) and the variance is usually given by a simple function of the mean \( \sigma^2(f_s) = v(g(f_s)) \), hence we use Gaussian-Hermite quadrature with \( N(f_s|\mu_{f_s}, \sigma^2_{f_s}) \approx p(f_s|D, x_s) \) to compute

\[
\mu_{y_s} = \int g(f_s)p(f_s|D, x_s)df_s, \text{ and } \\
\sigma^2_{y_s} = \int \left[ v(g(f_s)) + (g(f_s) - \mu_{y_s})^2 \right] p(f_s|D, x_s)df_s \neq v(\mu_{y_s}).
\]

4. Finally the warped Gaussian likelihood predictive distribution with strictly monotonically increasing warping function \( g \) is given by the expression

\[ p(y_s|D, x_s) = g'(y_s)N\left(g(y_s)|\mu_{f_s}, \sigma^2_n + \sigma^2_{f_s}\right) \]

so that the predictive moments can be computed by Gaussian-Hermite quadrature.

In the following, we will detail how and which likelihood functions are implemented in the GPML toolbox. Further, we will mention dependencies between likelihoods and inference methods and provide some analytical expressions in addition to some likelihood implementations.

### 4.2 Interface

The likelihoods are in fact the most challenging object in our implementation. Different inference algorithms require different aspects of the likelihood to be computed, therefore the interface is rather involved as detailed below.

```matlab
likFunctions.m
1 % likelihood functions are provided to be used by the gp.m function:
2 %
3 % likErf (Error function, classification, probit regression)
4 % likLogistic (Logistic, classification, logit regression)
5 % likUni (Uniform likelihood, classification)
6 %
7 % likGauss (Gaussian, regression)
8 % likGaussWarp (Warped Gaussian, regression)
9 % likGumbel (Gumbel likelihood for extremal values)
10 % likLaplace (Laplacian or double exponential, regression)
11 % likSech2 (Sech-square, regression)
12 % likT (Student’s t, regression)
13 %
14 % likPoisson (Poisson regression, count data)
15 % likGamma (Nonnegative regression, positive data)
16 % likExp (Nonnegative regression, positive data)
17 % likInvGauss (Nonnegative regression, positive data)
18 % likBeta (Beta regression, interval data)
19 %
20 % likMix (Mixture of individual covariance functions)
21 %
22 % The likelihood functions have three possible modes, the mode being selected
```
as follows (where "lik" stands for any likelihood function in "lik/lik*.m").

1) With one or no input arguments: [REPORT NUMBER OF HYPERPARAMETERS]

   s = lik OR s = lik(hyp)

The likelihood function returns a string telling how many hyperparameters it expects, using the convention that "D" is the dimension of the input space. For example, calling "likLogistic" returns the string '0'.

2) With three or four input arguments: [PREDICTION MODE]

   lp = lik(hyp, y, mu) OR [lp, ymu, ys2] = lik(hyp, y, mu, s2)

This allows to evaluate the predictive distribution. Let p(y_*|f_*) be the likelihood of a test point and N(f_*|mu,s2) an approximation to the posterior marginal p(f_*|x_*,x,y) as returned by an inference method. The predictive distribution p(y_*|x_*,x,y) is approximated by.

   q(y_*) = \int N(f_*|mu,s2) p(y_*|f_*) df_*

lp = log( q(y) ) for a particular value of y, if s2 is [] or 0, this corresponds to log( p(y|mu) )

ymu and ys2 the mean and variance of the predictive marginal q(y)

note that these two numbers do not depend on a particular value of y

All vectors have the same size.

3) With five or six input arguments, the fifth being a string [INFEERENCE MODE]

   [varargout] = lik(hyp, y, mu, s2, inf) OR
   [varargout] = lik(hyp, y, mu, s2, inf, i)

There are three cases for inf, namely a) infLaplace, b) infEP and c) infVB.
The last input i, refers to derivatives w.r.t. the ith hyperparameter.

a1) [lp,dlp,d2lp,d3lp] = lik(hyp, y, f, [], 'infLaplace')
   lp, dlp, d2lp and d3lp correspond to derivatives of the log likelihood

   log(p(y|f)) w.r.t. to the latent location f.

   lp = log( p(y|f) )
   dlp = d log( p(y|f) ) / df
   d2lp = d^2 log( p(y|f) ) / df^2
   d3lp = d^3 log( p(y|f) ) / df^3

a2) [lp_dhyp,dlp_dhyp,d2lp_dhyp] = lik(hyp, y, f, [], 'infLaplace', i)

returns derivatives w.r.t. to the ith hyperparameter

   1p_dhyp = d log( p(y|f) ) / ( dhyp_i)
   d1p_dhyp = d^2 log( p(y|f) ) / (df dhyp_i)
   d2p_dhyp = d^3 log( p(y|f) ) / (df^2 dhyp_i)

b1) [lZ,dlZ,d2lZ] = lik(hyp, y, mu, s2, 'infEP')

let Z = \int p(y|f) N(f|mu,s2) df then

   lZ = log(Z)
   d1Z = d log(Z) / dmu
   d21Z = d^2 log(Z) / dmu^2

20
4.3 Implemented Likelihood Functions

The following table enumerates all (currently) implemented likelihood functions that can be found at `lik/lik<NAME>.m` and their respective set of hyperparameters $\rho$.

| lik<NAME>     | function | $y_i \in \mathbb{R}$ | $p_\rho(y_i|f_i) = \cdots$ | $\rho =$ |
|------------------|----------|----------------------|----------------------------|---------|
| Gauss            | Gaussian | $\mathcal{N}(y_i|f_i, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(y_i-f_i)^2}{2\sigma^2}\right)$ | $\ln \sigma$ |         |
| GaussWarp        | Warped Gaussian | $\mathcal{N}(g_\theta(y_i|f_i, \sigma^2)|g_\theta(y_i)$ | $\theta_1, \theta_n, \ln \sigma$ |         |
| Gumbel           | Gumbel    | $\frac{\sqrt{\pi}}{\sigma} \exp\left(-z_i - e^{-z_i}\right), z_i = \gamma + \frac{\pi(y_i-f_i)}{\sigma \sqrt{2}}, |s| = 1$ | $\ln \sigma$ |         |
| Sech2            | Sech-squared | $\frac{\rho}{\sqrt{\cosh^{-1}|y_i-f_i|^2}}, \tau = \frac{\rho}{\sqrt{2} \sigma^2}$ | $\ln \sigma$ |         |
| Laplace          | Laplacian | $\frac{1}{\sqrt{2\pi\rho}} \exp\left(-\frac{|y_i-f_i|^2}{2\rho}\right), b = \frac{\sigma}{\sqrt{2}}$ | $\ln \sigma$ |         |
| T                | Student's t | $\frac{\Gamma\left(\frac{n+1}{2}\right)}{\Gamma\left(\frac{n}{2}\right)} \frac{\rho}{\sqrt{n\sigma^2}} \left(1 + \frac{(y_i-f_i)^2}{n\sigma^2}\right)^{-\frac{n+1}{2}}$ | $\ln(n/v-1), \ln \sigma$ |         |

| lik<NAME>     | function | $y_i \in \{\pm 1\}$ | $p_\rho(y_i|f_i) = \cdots$ | $\rho =$ |
|------------------|----------|----------------------|----------------------------|---------|
| Logistic         | Logistic function | $1_{(y_i \neq f_i)}$ | $\emptyset$ |         |
| Uni              | Label noise | $\frac{1}{2}$ | $\emptyset$ |         |

| lik<NAME>     | function | $y_i \in \mathbb{N}$ | $p_\rho(y_i|f_i) = \cdots$ | $\rho =$ |
|------------------|----------|----------------------|----------------------------|---------|
| Poisson          | Poisson | $\mu^y \cdot e^{-\mu}, \mu = e^f$ or $\mu = \log(1+e^f)$ | $\emptyset$ |         |
| lik<NAME>     | function | $y_i \in \mathbb{R}_+ \setminus \{0\}$ | $p_\rho(y_i|f_i) = \cdots$ | $\rho =$ |
| Weibull          | Weibull | $\mu y_1/\mu + (y_1/\mu)^{1-k} \exp\left(-(y_1/\mu)^k\right)$ | $\ln \kappa$ |         |
| Gamma            | Gamma    | $\frac{\alpha y_i^{\gamma_1}}{\Gamma(\gamma_1)} \mu^{-\gamma_1} \exp\left(-\frac{\gamma_1 \mu}{\mu_y}\right)$ | $\ln \alpha$ |         |
| Exp              | Exponential | $\mu^{-1} \exp\left(-\frac{y_i}{\mu}\right)$ | $\emptyset$ |         |
| InvGauss         | Inverse Gaussian | $\sqrt{2 \pi \mu_y} \exp\left(-\frac{\lambda (y_i/\mu)^2}{2 \mu_y y_i}\right)$ | $\ln \lambda$ |         |

| lik<NAME>     | function | $y_i \in [0, 1]$ | $p_\rho(y_i|f_i) = \cdots$ | $\rho =$ |
|------------------|----------|------------------|----------------------------|---------|
| Beta             | Beta    | $\frac{1}{\Gamma(\lambda)} (\Phi(1-\mu) y_i^{\lambda-1} (1-y)_i^{1-\mu})$ | $\ln \phi$ |         |

Composite likelihood functions $p_\rho(y_i|f_i), p_\rho(y_i|f_i) \rightarrow p_\rho(y_i|f_i)$
4.4 Usage of Implemented Likelihood Functions

Some code examples taken from doc/usageLik.m illustrate how to use simple and composite likelihood functions to specify a GP model.

Syntactically, a likelihood function \( l \) is defined by

\[
\text{lk} := \text{'func'} \mid @\text{func} \quad \text{// simple}
\]

\[
\text{lf} := \{\text{lk}\} \mid \{\text{param, lk}\} \mid \{\text{lk, \ldots, lk}\} \quad \text{// composite}
\]

i.e., it is either a string containing the name of a likelihood function, a pointer to a likelihood function or one of the former in combination with a cell array of likelihood functions and an additional list of parameters.

```matlab
% demonstrate usage of likelihood functions
%
% See also likFunctions.m.
%
% gpml copyright

clear all, close all
n = 5; f = randn(n,1); % create random latent function values

% set up simple classification likelihood functions
yc = sign(f);
lc0 = {'likErf'}; hypc0 = [];
lc1 = {@likLogistic}; hypc1 = [];
lc2 = {'likUni'}; hypc2 = [];
lc3 = {'likMix','likUni',@likErf}; hypc3 = log([1;2]); %mixture

% set up simple regression likelihood functions
yr = f + randn(n,1)/20;

% noise standard deviation
sn = 0.1;

lr0 = {'likGauss'}; hypr0 = log(sn);
lr1 = {'likLaplace'}; hypr1 = log(sn);
lr2 = {'likSech2'}; hypr2 = log(sn);
lr3 = {'likT'}; hypr3 = [log(nu-1); log(sn)];
lr4 = {'likMix',{lr0,lr1}}; hypr4 = [log([1,2]);hypr0;hypr1];

a = 1; % set up warped Gaussian with \( g(y) = y + a*sign(y)\cdot y^2 \)

% number of degrees of freedom
nu = 4;

lr5 = {'likGauss','+poly2'}; hypr5 = log([a;sn]);
lr6 = {'likGumbel','+'}; hypr6 = log(sn);

% set up Poisson regression
yp = fix(abs(f)) + 1;
lp0 = @likPoisson,'logistic'; hypp0 = [];
lp1 = @likPoisson,'exp'; hypp1 = [];

% set up other GLM likelihoods for positive or interval regression
lg1 = @likGamma,'logistic'; al = 2; hyp.lik = log(al);
lg2 = @likInvGauss,'exp'; lam = 1.1; hyp.lik = log(lam);
lg3 = @likBeta,'expexp'; phi = 2.1; hyp.lik = log(phi);
lg4 = @likBeta,'logit'; phi = 4.7; hyp.lik = log(phi);

% specify the likelihood function
lik = lc0; hyp = hypc0; y = yc;
lik = lr4; hyp = hypr4; y = yr;
lik = lp1; hyp = hypp1; y = yp;
```
45 46 % 1) query the number of parameters
47 feval(lik{:})
48
49 % 2) evaluate the likelihood function on f
50 exp(feval(lik{:}, hyp, y, f))
51
52 % 3a) evaluate derivatives of the likelihood
53 [lp,dlp,d2lp,d3lp] = feval(lik{:}, hyp, y, f, [], 'infLaplace');
54
55 % 3b) compute Gaussian integrals w.r.t. likelihood
56 mu = f; s2 = rand(n,1);
57 [lZ,dlZ,d2lZ] = feval(lik{:}, hyp, y, mu, s2, 'infEP');
58
59 % 3c) obtain lower bound on likelihood
60 ga = rand(n,1);
61 [b,z] = feval(lik{:}, hyp, y, [], ga, 'infVB');

4.5 Compatibility Between Likelihoods and Inference Methods

The following table lists all possible combinations of likelihood function and inference methods.

<table>
<thead>
<tr>
<th>Likelihood \ Inference</th>
<th>Exact</th>
<th>EP</th>
<th>Laplace</th>
<th>VB</th>
<th>MCMC</th>
<th>LOO</th>
<th>Type, Output Domain</th>
<th>Alternative Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>regression, R</td>
<td></td>
</tr>
<tr>
<td>Warped Gaussian</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>regression, R</td>
<td></td>
</tr>
<tr>
<td>Gumbel</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>regression, R</td>
<td></td>
</tr>
<tr>
<td>Sech-squared</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>regression, R</td>
<td>logistic distribution</td>
</tr>
<tr>
<td>Laplacian</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>regression, R</td>
<td>double exponential</td>
</tr>
<tr>
<td>Student's t</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>regression, R</td>
<td></td>
</tr>
<tr>
<td>Mixture</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>mixing meta likelihood</td>
<td></td>
</tr>
<tr>
<td>Error function</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>classification, [±1]</td>
<td>probit regression</td>
</tr>
<tr>
<td>Logistic function</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>classification, [±1]</td>
<td>logit regression</td>
</tr>
<tr>
<td>Uniform</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>classification, [±1]</td>
<td>label noise</td>
</tr>
<tr>
<td>Weibull</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>positive data, R+ {0}</td>
<td>nonnegative regression</td>
</tr>
<tr>
<td>Gamma</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>positive data, R+ {0}</td>
<td>nonnegative regression</td>
</tr>
<tr>
<td>Exp</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>positive data, R+ {0}</td>
<td>nonnegative regression</td>
</tr>
<tr>
<td>Inverse Gaussian</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>positive data, R+ {0}</td>
<td>nonnegative regression</td>
</tr>
<tr>
<td>Poisson</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>count data, N</td>
<td>Poisson regression</td>
</tr>
<tr>
<td>Beta</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>interval data, [0, 1]</td>
<td>beta regression</td>
</tr>
</tbody>
</table>

(✓)* EP might not converge in some cases since quadrature is used.

Exact inference is only tractable for Gaussian likelihoods. Expectation propagation together with Student's t likelihood is inherently unstable due to non-log-concavity. Laplace's approximation for Laplace likelihoods is not sensible because at the mode the curvature and the gradient is undefined due to the non-differentiable peak of the Laplace distribution. Special care has been taken for the non-convex optimisation problem imposed by the combination Student's t likelihood and Laplace's approximation.

4.6 Gaussian Likelihood

The Gaussian likelihood is the simplest likelihood because the posterior distribution is not only Gaussian but can be computed analytically. In principle, the Gaussian likelihood would only be operated in conjunction with the exact inference method but we chose to provide compatibility with all other inference algorithms as well because it enables code testing and allows to switch between
different regression likelihoods very easily.

\[ \text{likGauss.m} \]

```matlab
function [varargout] = likGauss(hyp, y, mu, s2, inf, i)
% likGauss - Gaussian likelihood function for regression. The expression for the
% likelihood is
% likGauss(t) = exp(-(t-y)^2/2*sn^2) / sqrt(2*pi*sn^2),
% where y is the mean and sn is the standard deviation.
%
% The hyperparameters are:
%
% log(sn) [varargout] = likGauss(hyp, y, mu, s2, inf, i)

if nargin<3, varargout = {'1'}; return; end % report number of hyperparameters

sn2 = exp(2*hyp);

if nargin<5 % prediction mode if inf is not present
else
switch inf
    case 'infLaplace'
    case 'infEP'
    case 'infVB'
end
end

if isempty(y), y = zeros(size(mu)); end
sn2zero = 1; if nargin>3&&numel(s2)>0&&norm(s2)>eps, sn2zero = 0; end % s2==0 ?

if s2zero % log probability
    lp = -(y-mu).^2./sn2/2-log(2*pi*sn2)/2; s2 = 0;
else
    lp = likGauss(hyp, y, mu, s2, 'infEP'); % prediction
end

ymu = {}; ys2 = {};
if nargout>1
    ymu = mu; % first y moment
    if nargout>2
        ys2 = s2 + sn2; % second y moment
    end
end
varargout = {lp,ymu,ys2};
```

The Gaussian likelihood function has a single hyperparameter \( \rho \), the log of the noise standard deviation \( \sigma_n \).
4.6.1 Exact Inference

Exact inference doesn’t require any specific likelihood related code; all computations are done directly by the inference method, section 3.1.

4.6.2 Laplace’s Approximation

\begin{align*}
\langle \text{Laplace’s method with Gaussian likelihood} \rangle & \equiv \\
\text{if nargin}<6 & \% \text{no derivative mode} \\
\text{ymmu} = y - \mu; & dlp = {}; \quad \text{d2lp} = {}; \quad \text{d3lp} = {};
\end{align*}

\begin{align*}
\text{lp} & = -\text{ymmu} \cdot \frac{1}{2} \cdot \frac{1}{\text{sn2}} - \frac{\log(2\pi\text{sn2})}{2} \\
\text{if nargout}>1 & \\
\text{dlp} & = \frac{\text{ymmu}}{\text{sn2}}; \quad \% \text{dlp, derivative of log likelihood} \\
\text{if nargout}>2 & \% \text{d2lp, 2nd derivative of log likelihood} \\
\text{d2lp} & = -\frac{\text{ones(size(ymmu))}}{\text{sn2}} \\
\text{if nargout}>3 & \% \text{d3lp, 3rd derivative of log likelihood} \\
\text{d3lp} & = \text{zeros(size(ymmu))}; \\
\end{align*}

\begin{align*}
\text{if nargin}<6 & \% \text{no derivative mode} \\
\text{lZ} & = -\frac{(y - \mu)^2}{\text{sn2} + \text{s2}} / 2 - \frac{\log(2\pi(\text{sn2} + \text{s2})/2)}{2} \\
\text{dlZ} & = {}; \quad \text{d2lZ} = {};
\end{align*}

\begin{align*}
\text{if nargout}>1 & \\
\text{dlZ} & = \frac{(y - \mu)}{(\text{sn2} + \text{s2})}; \quad \% \text{1st derivative w.r.t. mean} \\
\text{if nargout}>2 & \% \text{2nd derivative w.r.t. mean} \\
\text{d2lZ} & = -\frac{1}{(\text{sn2} + \text{s2})} \\
\end{align*}

\begin{align*}
\text{if nargin}<6 & \% \text{no derivative mode} \\
\text{t(s)} & = \exp(-\frac{(y-s)^2}{2\text{sn2}})/\sqrt{2\pi\text{sn2}} \\
\text{the bound has the form:} & (b+z/\text{ga}) \cdot f - f^2/(2\text{ga}) - h(\text{ga})/2 \\
\text{n} & = \text{numel(s2)}; \quad \text{b} = \text{zeros(n,1)}; \quad \text{y} = y \cdot \text{ones(n,1)}; \quad \text{z} = y \\
\text{varargout} & = \{\text{b},\text{z}\};
\end{align*}

4.6.3 Expectation Propagation

\begin{align*}
\langle \text{EP inference with Gaussian likelihood} \rangle & \equiv \\
\text{if nargin}<6 & \% \text{no derivative mode} \\
\text{d1Z} & = {}; \quad \text{d21Z} = {};
\end{align*}

\begin{align*}
\text{if nargout}>1 & \\
\text{d1Z} & = \frac{(y - \mu)}{(\text{sn2} + \text{s2})}; \quad \% \text{1st derivative w.r.t. mean} \\
\text{if nargout}>2 & \% \text{2nd derivative w.r.t. mean} \\
\text{d21Z} & = -\frac{1}{(\text{sn2} + \text{s2})} \\
\end{align*}

\begin{align*}
\text{if nargin}<6 & \% \text{no derivative mode} \\
\text{dlZhyp} & = ((y - \mu) \cdot \frac{2}{(\text{sn2} + \text{s2})}) - 1) / (1 + \text{s2} / \text{sn2}); \quad \% \text{deriv. w.r.t. hyp.lik} \\
\text{varargout} & = \{\text{dlZhyp}\};
\end{align*}

4.6.4 Variational Bayes

\begin{align*}
\langle \text{Variational Bayes inference with Gaussian likelihood} \rangle & \equiv \\
\% \text{variational lower site bound} \\
\% \text{t(s)} = \exp(-\frac{(y-s)^2}{2\text{sn2}})/\sqrt{2\pi\text{sn2}} \\
% \text{the bound has the form:} & (b+z/\text{ga}) \cdot f - f^2/(2\text{ga}) - h(\text{ga})/2 \\
\text{n} & = \text{numel(s2)}; \quad \text{b} = \text{zeros(n,1)}; \quad \text{y} = y \cdot \text{ones(n,1)}; \quad \text{z} = y \\
\text{varargout} & = \{\text{b},\text{z}\};
\end{align*}
4.7 Warped Gaussian Likelihood

Starting from the likelihood $p(y|f)$ we are sometimes facing the situation where the data $y \in \mathcal{Y} \subseteq \mathbb{R}$ is not distributed according to $p(y|f)$ but some nonlinear transformation of the data $g(y) = z$ so that $z \sim p(z|f)$. Here, the warping function $g : \mathcal{Y} \rightarrow \mathbb{R}$ needs to be strictly monotonically increasing i.e. $g'(y) > 0$. Formally, we start from the fact that $p(z|f)$ integrates to one and use the derivative $dz = g'(y)dy$ to substitute

$$\int p(z|f)dz = 1 = \int p_g(y|f)dy, \ p_g(y|f) = p(g(y)|f)g'(y)$$

where we have defined the log warped likelihood $\ln p_g(y|f) = \ln p(g(y)|f) + \ln g'(y)$. The interesting bit is that approximate inference methods such as infExact, infLaplace, infEP, infVB, infKL remain fully feasible; only prediction and derivatives become more involved. The usual GP inference is recovered by using the identity warping function $g : y \mapsto y$. The construction works in principle for any likelihood but our implementation in likGaussWarp is limited to the Gaussian likelihood.

Hyperparameter derivatives

Hyperparameter derivatives for infLaplace are obtained as follows

$$\frac{\partial}{\partial \theta} \ln \frac{\partial^k}{\partial f^k} p_g(y|f) = \frac{\partial}{\partial \theta} \ln \frac{\partial^k}{\partial f^k} p(g(y)|f) + \frac{\partial}{\partial \theta} \frac{\partial^k}{\partial f^k} \ln g'(y), \ k = 0, 1, 2$$

$$= -\frac{\partial^{k+1}}{\partial f^{k+1}} \ln p(g(y)|f) \frac{\partial}{\partial \theta} \ln g(y) + \frac{\partial}{\partial \theta} \frac{\partial^k}{\partial f^k} \ln g'(y).$$

Similarly for infEP the derivatives are given by

$$\frac{\partial}{\partial \theta} \int \ln p_g(y|f)N(f|\mu, \sigma^2)df = \frac{\partial}{\partial \theta} \int \ln p(g(y)|f)N(f|\mu, \sigma^2)df + \frac{\partial}{\partial \theta} \ln g'(y)$$

$$= -\frac{\partial}{\partial \mu} \ln \int p(g(y)|f)N(f|\mu, \sigma^2)df \frac{\partial}{\partial \theta} g(y) + \frac{\partial}{\partial \theta} \ln g'(y).$$

This trick above works for any homoscedastic likelihood where $p(y|f) = p(y + \beta|f + \beta)$ such as likGauss, likLaplace, likSech2 and likT.

Predictive moments

As detailed in \(\mathbb{Z}\), the predictive distribution is – for Gaussian likelihood – given by

$$p(z_*|\mathcal{D}, x_*) = \int p(z_*|f_*)p(f_*|\mathcal{D}, x_*)df_* = \int \mathcal{N}(z_*|f_*, \sigma_n^2)\mathcal{N}(f_*|\mu_f, \sigma_f^2)df_*$$

$$= \mathcal{N}(z_*|\mu_{f_*}, \sigma_n^2 + \sigma_f^2), \text{ where } z_* = g(y_*)$$

$$p(y_*|\mathcal{D}, x_*) = g'(y_*)\mathcal{N}(g(y_*)|\mu_{f_*}, \sigma_n^2 + \sigma_f^2).$$

Hence, the predictive moments are obtained by the 1d integrals

$$\mu_{y_*} = \int y_* g'(y_*) \mathcal{N}(g(y_*)|\mu_{f_*}, \sigma_n^2 + \sigma_f^2)dy_*$$

$$= \int g^{-1}(z_*) \mathcal{N}(z_*|\mu_{f_*}, \sigma_n^2 + \sigma_f^2)dz_*, \text{ and}$$

$$\sigma_{y_*}^2 = \int (y_* - \mu_{y_*})^2 g'(y_*) \mathcal{N}(g(y_*)|\mu_{f_*}, \sigma_n^2 + \sigma_f^2)dy_*$$

$$= \int (g^{-1}(z_*) - \mu_{y_*})^2 \mathcal{N}(z_*|\mu_{f_*}, \sigma_n^2 + \sigma_f^2)dz_*.$$
4.8 Gumbel Likelihood

Distributions of extrema are well captured by the Gumbel distribution

\[ p(y) = \frac{\pi}{\sigma \sqrt{6}} \exp \left(-\frac{y - \eta}{\sigma} z \right), \quad z = s \frac{y - \eta}{\beta}, \quad s \in \{\pm 1\} \]

with mean \( \mu = \eta + \beta y \) and variance \( \sigma^2 = \pi^2 \beta^2 / 6 \) where \( \gamma = 0.57721566490153 \) denotes Euler–Mascheroni’s constant. Skewness is approximately given by 1.1395 between left and right skewness and kurtosis is 12/5. The final expression for the Gumbel likelihood is

\[ p(y|f) = \frac{\pi}{\sigma \sqrt{6}} \exp \left(-\frac{y - \eta}{\sigma} z \right), \quad z = \gamma + s \frac{\pi}{\sigma \sqrt{6}} (y - f), \quad s \in \{\pm 1\}. \]

4.9 Laplace Likelihood

Laplace’s Approximation

The following derivatives are needed:

\[
\begin{align*}
\ln p(y|f) &= -\ln(2b) - \frac{|f - y|}{b} \\
\frac{\partial \ln p}{\partial f} &= \text{sign}(f - y) \\
\frac{\partial^2 \ln p}{(\partial f)^2} &= \frac{\partial^3 \ln p}{(\partial f)^3} = \frac{\partial^3 \ln p}{(\partial \ln \sigma_n)(\partial f)^{\frac{3}{2}}} = 0 \\
\frac{\partial \ln p}{\partial \ln \sigma_n} &= \frac{|f - y|}{b} - 1
\end{align*}
\]

Expectation Propagation

Expectation propagation requires integration against a Gaussian measure for moment matching.

We need to evaluate \( \ln Z = \ln \int \mathcal{L}(y|f, \sigma_n^2) N(f|\mu^*, \sigma^2) df \) as well as the derivatives \( \frac{\partial \ln Z}{\partial \mu} \) and \( \frac{\partial^2 \ln Z}{\partial \mu^2} \) where \( N(f|\mu^*, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma^2} \exp \left(-\frac{(f - \mu^*)^2}{2\sigma^2}\right), \) \( \mathcal{L}(y|f, \sigma_n^2) = \frac{1}{2b} \exp \left(-\frac{|y - f|}{b}\right), \) and \( b = \frac{\sigma_n}{\sqrt{2}}. \) As a first step, we reduce the number of parameters by means of the substitution \( \bar{f} = \frac{f - y}{\sigma_n} \) yielding

\[
Z = \int \mathcal{L}(y|f, \sigma_n^2) N(f|\mu, \sigma^2) df
= \frac{1}{\sqrt{2\pi}\sigma} \int \frac{\sqrt{2}}{2\sigma\sqrt{2\pi}} \int \exp \left(-\frac{(f - \mu)^2}{2\sigma^2}\right) \exp \left(-\frac{z^2}{2}\right) df
= \frac{\sqrt{2}}{2\sigma\sqrt{2\pi}} \int \exp \left(-\frac{(\sigma_n \bar{f} + y - \mu)^2}{2\sigma^2}\right) \exp \left(-\frac{z^2}{2}\right) d\bar{f}
= \frac{\sigma_n}{\sqrt{2\pi}} \int \frac{\sigma_n^2 (\bar{f}^2 - \mu^2)}{2\sigma^2} \mathcal{L}(\bar{f}|0, 1) d\bar{f}
= \frac{1}{\sigma_n} \int \mathcal{L}(f|0, 1) N(f|\bar{\mu}, \sigma^2) df
\]

\[
\ln Z = \ln \hat{Z} - \ln \sigma_n = \ln \int \mathcal{L}(f|0, 1) N(f|\mu^*, \sigma^2) df - \ln \sigma_n
\]

27
with \( \tilde{\mu} = \frac{\mu - y}{\sigma_n} \) and \( \tilde{\sigma} = \frac{\sigma}{\sigma_n} \). Thus, we concentrate on the simpler quantity \( \ln \tilde{Z} \).

\[
\ln Z = \ln \left[ \exp \left( \frac{(f - \tilde{\mu})^2}{2\tilde{\sigma}^2} - \sqrt{2}f \right) \right] d\ln \tilde{\sigma} \ln \sqrt{2\sigma_n} - \ln \sqrt{2}\sigma_n
\]

\[
= \ln \left[ \int_{-\infty}^{0} \exp \left( \frac{(f - \tilde{\mu})^2}{2\tilde{\sigma}^2} + \sqrt{2}f \right) df + \int_{0}^{\infty} \exp \left( \frac{(f - \tilde{\mu})^2}{2\tilde{\sigma}^2} - \sqrt{2}f \right) df \right] + C
\]

\[
= \ln \left[ \int_{-\infty}^{0} \exp \left( \frac{m_+}{2\tilde{\sigma}^2} \right) df + \int_{0}^{\infty} \exp \left( \frac{m_+}{2\tilde{\sigma}^2} \right) df \right] \left[ \ln \left( \frac{m_+}{2\tilde{\sigma}^2} \right) - \ln \left( \frac{m_+}{2\tilde{\sigma}^2} \right) \right] - \ln \sqrt{2}\sigma_n
\]

Here, \( \Phi(z) = \int_{-\infty}^{z} \mathcal{N}(0, 1) df \) denotes the cumulative Gaussian distribution. Finally, we have

\[
\ln Z = \ln \left[ \exp \left( -\sqrt{2}\tilde{\mu} \right) \Phi \left( \frac{m_-}{\tilde{\sigma}} \right) + \exp \left( \sqrt{2}\tilde{\mu} \right) \Phi \left( \frac{m_+}{\tilde{\sigma}} \right) \right] + \tilde{\sigma}^2 - \ln \sqrt{2}\sigma_n
\]

where \( z_+ = \frac{\tilde{\mu}}{\tilde{\sigma}} + \sqrt{2} = \frac{\mu - y}{\sigma_n} + \frac{\sigma}{\sigma_n} \sqrt{2} \), \( z_- = \frac{\tilde{\mu}}{\tilde{\sigma}} - \sqrt{2} = \frac{\mu - y}{\sigma_n} - \frac{\sigma}{\sigma_n} \sqrt{2} \) and \( \tilde{\mu} = \frac{\mu - y}{\sigma_n}, \tilde{\sigma} = \frac{\sigma}{\sigma_n} \).

Now, using \( \frac{d}{d\sigma} \ln \Phi(z) = \frac{1}{\Phi(z)} \frac{d}{dz} \Phi(z) = \frac{\mathcal{N}(z)}{\Phi(z)} \frac{dz}{d\sigma} \) we tackle first derivative

\[
\frac{\partial \ln Z}{\partial \mu} = e^{a_+} \frac{\partial a_+}{\partial \mu} + e^{a_-} \frac{\partial a_-}{\partial \mu}
\]

\[
\frac{\partial a_+}{\partial \mu} = \frac{\partial}{\partial \mu} \ln \Phi(z_+) + \frac{\sqrt{2}}{\sigma_n}
\]

\[
= -\frac{N(z_+)}{\sigma \Phi(z_+)} + \frac{\sqrt{2}}{\sigma_n} = -\frac{q_+}{\sigma} + \frac{\sqrt{2}}{\sigma_n}
\]

\[
\frac{\partial a_-}{\partial \mu} = \frac{\partial}{\partial \mu} \ln \Phi(z_-) - \frac{\sqrt{2}}{\sigma_n}
\]

\[
= \frac{N(z_-)}{\sigma \Phi(z_-)} - \frac{\sqrt{2}}{\sigma_n} = \frac{q_-}{\sigma} - \frac{\sqrt{2}}{\sigma_n}
\]

\[
\frac{\partial a_\pm}{\partial \mu} = \pm \frac{q_\pm}{\sigma} \pm \frac{\sqrt{2}}{\sigma_n}
\]
as well as the second derivative

\[
\frac{\partial^2 \ln Z}{\partial \mu^2} = \frac{\partial}{\partial \mu} \left( e^{a_+} \frac{\partial a_+}{\partial \mu} \right) + \frac{\partial}{\partial \mu} \left( e^{a_-} \frac{\partial a_-}{\partial \mu} \right) - \left( \frac{\partial \ln Z}{\partial \mu} \right)^2
\]

\[
\frac{\partial}{\partial \mu} \left( e^{a_\pm} \frac{\partial a_\pm}{\partial \mu} \right) = e^{a_\pm} \left[ \left( \frac{\partial a_\pm}{\partial \mu} \right)^2 + \frac{\partial^2 a_\pm}{\partial \mu^2} \right]
\]

\[
\frac{\partial^2 a_+}{\partial \mu^2} = -1 \frac{\partial}{\partial \mu} N(-z_+) \Phi(-z_+) - \frac{\partial}{\partial \mu} \Phi(-z_+) N(-z_+)
\]

\[
\frac{\partial^2 a_-}{\partial \mu^2} = \frac{\partial}{\partial \mu} N(z_-) \Phi(z_-) - \frac{\partial}{\partial \mu} \Phi(z_-) N(z_-)
\]

\[
\frac{\partial^2 a_\pm}{\partial \mu^2} = -q_\pm \pm q_\pm z_\pm
\]

which can be simplified to

\[
\frac{\partial^2 \ln Z}{\partial \mu^2} = \frac{e^{a_+} + e^{a_-}}{e^{a_+} + e^{a_-}} - \left( \frac{\partial \ln Z}{\partial \mu} \right)^2
\]

using

\[
b_\pm = \left( \frac{\partial a_\pm}{\partial \mu} \right)^2 + \frac{\partial^2 a_\pm}{\partial \mu^2} = \left( \frac{q_\pm}{\sigma} \pm \frac{\sqrt{2}}{\sigma_n} \right)^2 - \frac{q_\pm^2 \pm q_\pm z_\pm}{\sigma^2}
\]

\[
= \left( \frac{q_\pm}{\sigma} - \frac{\sqrt{8}}{\sigma_n} \pm \frac{z_\pm}{\sigma^2} \right) q_\pm.
\]

We also need

\[
\frac{\partial \ln Z}{\partial \ln \sigma_n} = \frac{e^{a_+} \frac{\partial a_+}{\ln \sigma_n} + e^{a_-} \frac{\partial a_-}{\ln \sigma_n}}{e^{a_+} + e^{a_-}} - \frac{2\sigma^2}{\sigma_n^2} - 1.
\]

Variational Bayes

We need \( h(\gamma) \) and its derivatives as well as \( \beta(\gamma) \):
\[ h(\gamma) = \frac{2}{\sigma_n^2} \gamma + \ln(2\sigma_n^2) + y^2\gamma^{-1} \]
\[ h'(\gamma) = \frac{2}{\sigma_n^2} - y^2\gamma^{-2} \]
\[ h''(\gamma) = 2y^2\gamma^{-3} \]
\[ \beta(\gamma) = y\gamma^{-1} \]

4.10 Student’s t Likelihood

The likelihood has two hyperparameters (both represented in the log domain to ensure positivity): the degrees of freedom \( \nu \) and the scale \( \sigma_n \) with mean \( y \) (for \( \nu > 1 \)) and variance \( \frac{\nu}{\nu - 2} \sigma_n^2 \) (for \( \nu > 2 \)).

\[ p(y|f) = Z \cdot \left(1 + \frac{(f - y)^2}{\nu \sigma_n^2}\right)^{-\frac{\nu+1}{2}}, \quad Z = \frac{\Gamma \left(\frac{\nu+1}{2}\right)}{\Gamma \left(\frac{\nu}{2}\right) \sqrt{\nu \pi \sigma_n^2}} \]

Laplace’s Approximation

For the mode fitting procedure, we need derivatives up to third order; the hyperparameter derivatives at the mode require some mixed derivatives. All in all, using \( r = y - f \), we have

\[ \ln p(y|f) = \ln \Gamma \left(\frac{\nu+1}{2}\right) - \ln \Gamma \left(\frac{\nu}{2}\right) - \frac{1}{2} \ln \nu \pi \sigma_n^2 - \frac{y + 1}{2} \ln \left(1 + \frac{r^2}{\nu \sigma_n^2}\right) \]
\[ \frac{\partial \ln p}{\partial f} = \frac{(\nu + 1) r}{r^2 + \nu \sigma_n^2} \]
\[ \frac{\partial^2 \ln p}{(\partial f)^2} = \frac{(\nu + 1) (r^2 - \nu \sigma_n^2)}{(r^2 + \nu \sigma_n^2)^2} \]
\[ \frac{\partial^3 \ln p}{(\partial f)^3} = 2(\nu + 1) \frac{r^3 - 3r \nu \sigma_n^2}{(r^2 + \nu \sigma_n^2)^3} \]
\[ \frac{\partial \ln p}{\partial \ln \nu} = \frac{\partial Z}{\partial \ln \nu} - \frac{\nu}{2} \ln \left(1 + \frac{r^2}{\nu \sigma_n^2}\right) + \frac{\nu + 1}{2} \cdot \frac{r^2}{r^2 + \nu \sigma_n^2} \]
\[ \frac{\partial Z}{\partial \ln \nu} = \frac{\nu d \ln \Gamma \left(\frac{\nu+1}{2}\right)}{2 d \ln \nu} - \frac{\nu d \ln \Gamma \left(\frac{\nu}{2}\right)}{2 d \ln \nu} + 1 \]
\[ \frac{\partial^3 \ln p}{(\partial \ln \nu)(\partial f)^2} = \frac{\nu (r^2 - 3(\nu + 1) \sigma_n^2) + \nu \sigma_n^2}{(r^2 + \nu \sigma_n^2)^3} \]
\[ \frac{\partial \ln p}{\partial \ln \sigma_n} = \frac{\nu(r^2)}{(r^2 + \nu \sigma_n^2)^3} - 1 \]
\[ \frac{\partial^3 \ln p}{(\partial \ln \sigma_n)(\partial f)^2} = 2 \nu \sigma_n^2 (\nu + 1) \frac{\nu \sigma_n^2 - 3r^2}{(r^2 + \nu \sigma_n^2)^3} \]

4.11 Cumulative Logistic Likelihood

The likelihood has one hyperparameter (represented in the log domain), namely the standard deviation \( \sigma_n \).

\[ p(y|f) = Z \cdot \cosh^{-2} \left(\tau(f - y)\right), \quad \tau = \frac{\pi}{2\sigma_n \sqrt{3}}, \quad Z = \frac{\pi}{4\sigma_n \sqrt{3}} \]
Laplace’s Approximation

The following derivatives are needed where \( \phi(x) \equiv \ln(\cosh(x)) \)

\[
\begin{align*}
\ln p(y|f) &= \ln(\pi) - \ln(4\sigma_n \sqrt{3}) - 2\phi(\tau(f - y)) \\
\frac{\partial \ln p}{\partial f} &= 2\tau\phi'(\tau(f - y)) \\
\frac{\partial^2 \ln p}{(\partial f)^2} &= -2\tau^2\phi''(\tau(f - y)) \\
\frac{\partial^3 \ln p}{(\partial f)^3} &= 2\tau^3\phi'''(\tau(f - y)) \\
\frac{\partial^3 \ln p}{(\partial \ln \sigma_n)(\partial f)^2} &= 2\tau^2 \left(2\phi''(\tau(f - y)) + \tau(f - y)\phi'''(\tau(f - y))\right) \\
\frac{\partial \ln p}{\partial \ln \sigma_n} &= 2\tau(f - y)\phi'(\tau(f - y)) - 1
\end{align*}
\]

4.12 GLM Likelihoods: Poisson, Weibull, Gamma, Exponential, Inverse Gaussian and Beta

Data \( y \) from a space other than \( \mathbb{R} \) e.g. \( \mathbb{N}, \mathbb{R}_+ \) or \([0, 1]\) can be modeled using generalised linear model likelihoods \( p(y|f) \) where the expected value \( \mathbb{E}[y] = \mu \) is related to the underlying Gaussian process \( f \) by means of an inverse link function \( \mu = g(f) \). Typically, the likelihoods are from an exponential family, hence the variance \( \mathbb{V}[y] = \nu(\mu) \), is a simple function of the mean \( \mu \) as well as higher order moments such as skewness \( S[y] = s(\mu) \) and kurtosis \( K[y] = k(\mu) \).

Here, we directly specify the inverse link function \( \mu = g(f) \) defining the mapping from the GP \( f \) to the mean intensity \( \mu \). For numerical reasons, we work with the log of the inverse link function \( h(f) = \ln g(f) \) and use its derivatives \( h', h'' \) and \( h''' \) for subsequent computations. In the table below, we have summarised the GLM likelihood expressions, the moments, the range of their variables and the applicable inverse link functions.

| Likelihood | \( \nu(\mu) \) | \( s(\mu) \) | \( k(\mu) \) | \( p(y|f) = \) | \( y \in \) | \( \mu \in \) | Inverse Links |
|------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| Poisson    | \( \exp \)      | \( \logistic \) | \( \logistic \) | \( \mu^\theta \exp(-\mu/y) \) | \( \mathbb{N} \) | \( \mathbb{R}_+ \) | exp, logistic |
| Weibull    | \( \ln x \)     | \( \mu^\theta \) | \( \mu^{1/\theta} \) | \( \frac{\vartheta}{\exp\vartheta} - \frac{\vartheta}{\exp\vartheta} \) | \( \mathbb{R}_+, \mathbb{R}_+ \) | \( \mathbb{R}_+ \) | \( \mathbb{R}_+ \) exp, logistic |
| Gamma      | \( \ln x \)     | \( \mu^\theta \) | \( \mu^{1/\theta} \) | \( \frac{\vartheta}{\exp\vartheta} - \frac{\vartheta}{\exp\vartheta} \) | \( \mathbb{R}_+, \mathbb{R}_+ \) | \( \mathbb{R}_+ \) | \( \mathbb{R}_+ \) exp, logistic |
| Exponential | \( \exp \)      | \( \logistic \) | \( \logistic \) | \( \mu^\theta \exp(-\mu/y) \) | \( \mathbb{R}_+, \mathbb{R}_+ \) | \( \mathbb{R}_+ \) | \( \mathbb{R}_+ \) exp, logistic |
| Inv. Gauss | \( \ln x \)     | \( \mu^\theta \) | \( \mu^{1/\theta} \) | \( \frac{\vartheta}{\exp\vartheta} - \frac{\vartheta}{\exp\vartheta} \) | \( \mathbb{R}_+, \mathbb{R}_+ \) | \( \mathbb{R}_+ \) | \( \mathbb{R}_+ \) exp, logistic |
| Beta       | \( \ln x \)     | \( \mu(1 - \mu)/(1 + \mu) \) | \( \mu^\theta \exp(-\mu/y) \) | \( \mu^\theta \exp(-\mu/y) \) | \( \mathbb{R}_+, \mathbb{R}_+ \) | \( \mathbb{R}_+ \) | \( \mathbb{R}_+ \) exp, logistic |

4.12.1 Inverse Link Functions

Possible inverse link functions and their properties (\( \cup \) convex, \( \cap \) concave, \( \uparrow \) monotone) are summarised below:

<table>
<thead>
<tr>
<th>( \text{util/glm_invlk_*} )</th>
<th>( g(f) = \mu = )</th>
<th>( g : \mathbb{R} \to )</th>
<th>( g \text{ is} )</th>
<th>( h(f) = \ln \mu = )</th>
<th>( h \text{ is} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>exp</td>
<td>( e^x )</td>
<td>( \mathbb{R}_+ )</td>
<td>( \cup, \uparrow )</td>
<td>( f )</td>
<td>( \cup, \cap, \uparrow )</td>
</tr>
<tr>
<td>logistic</td>
<td>( \ln(1 + e^x) )</td>
<td>( \mathbb{R}_+ )</td>
<td>( \cup, \uparrow )</td>
<td>( \ln(1 + e^x) )</td>
<td>( \cap, \uparrow )</td>
</tr>
<tr>
<td>expexp</td>
<td>( \exp(-e^{-x}) )</td>
<td>( [0, 1] )</td>
<td>( \uparrow )</td>
<td>( -e^{-x} )</td>
<td>( \cup )</td>
</tr>
<tr>
<td>logit</td>
<td>( 1/(1 + e^{-x}) )</td>
<td>( [0, 1] )</td>
<td>( \uparrow )</td>
<td>( -\ln(1 + e^{-x}) )</td>
<td>( \cup )</td>
</tr>
</tbody>
</table>
**Exponential inverse link:** $\exp$

For $g(f) = e^f$ things are simple since $h(f) = f$, $h'(f) = 1$ and $h''(f) = h'''(f) = 0$.

**Logistic inverse link:** $\text{logistic}$

For $g(f) = \ln(1 + e^f)$ the derivatives of $h(f)$ are given by

$$
\begin{align*}
  h(f) &= \ln(\ln(1 + e^f)) \\
  h'(f) &= \frac{1}{\ln(1 + e^f)} s(-f), \quad s(f) = \frac{1}{1 + e^f}, \quad s'(f) = \frac{-e^f}{(1 + e^f)^2} = -s(-f)s(f) \\
  h''(f) &= \frac{1}{\ln(1 + e^f)} \frac{e^{-f}}{(1 + e^f)^2} - \frac{1}{\ln^2(1 + e^f)} \frac{e^f}{1 + e^f} \frac{1}{1 + e^{-f}} \\
  &= h'(f) [s(f) - h'(f)] \\
  h'''(f) &= h''(f) [s(f) - h'(f)] + h'(f) \left[ \frac{-e^f}{(1 + e^f)^2} - h''(f) \right] \\
  &= h''(f) [s(f) - 2h'(f)] - h'(f)s(f)s(-f).
\end{align*}
$$

Note that $g(f) = e^{h(f)} = \ln(1 + e^f)$ is convex and $h(f) = \ln(\ln(1 + e^f))$ with

$$h''(f) = \frac{1}{\ln(1 + e^f)} \left( 1 - \frac{e^f}{\ln(1 + e^f)} \right) \frac{1}{1 + e^f} \frac{1}{1 + e^{-f}} \leq 0$$

is concave since $e^f \geq \ln(1 + e^f)$ for all $f \in \mathbb{R}$.

**Double negative exponential inverse link:** $\text{expexp}$

For $g(f) = \exp(-e^{-f})$ the derivatives of $h(f)$ are given by

$$
\begin{align*}
  h(f) &= -e^{-f} \\
  h'(f) &= -h(f) \\
  h''(f) &= h(f) \\
  h'''(f) &= -h(f)
\end{align*}
$$

**Logit regression inverse link:** $\text{logit}$

For $g(f) = 1/(1 + e^{-f})$ the derivatives of $h(f)$ can be computed using the logistic inverse link function $h_\ell(f)$ since $h(f) = f - \exp(h_\ell(f))$

$$
\begin{align*}
  h(f) &= f - e^{h_\ell(f)} \\
  h'(f) &= 1 - e^{h_\ell(f)}h_\ell'(f) \\
  h''(f) &= -e^{h_\ell(f)}[h_\ell'(f)^2 + h_\ell''(f)] = e^{h_\ell(f)}s_\ell(-f)s_\ell^2(f) \\
  h'''(f) &= -e^{h_\ell(f)}[h_\ell'(f)^3 + 3h_\ell''(f)h_\ell'(f) + h_\ell'''(f)]
\end{align*}
$$

4.12.2 **Poisson Likelihood**

Count data $y \in \mathbb{N}^n$ can be modeled in the GP framework using the Poisson distribution $p(y) = \mu^y e^{-\mu}/y!$ with mean/variance $\mathbb{E}[y] = \mathbb{V}[y] = \mu$, skewness $S[y] = 1/\sqrt{\mu}$ and kurtosis $K[y] = 1/\mu$
leading to the likelihood

\[ p(y|f) = \mu^y \exp(-\mu/y), \quad \mu = g(f) \]
\[ \iff \ln p(y|f) = y \cdot \ln g(f) - g(f) - \ln \Gamma(y + 1). \]

For Laplace’s method to work, we need the first three derivatives of the log likelihood \( \ln p(y|f) \), where \( h(f) = \ln g(f) \)

\[
\frac{\partial}{\partial f} \ln p(y|f) = h'(f) [y - \exp(h(f))] \\
\frac{\partial^2}{\partial f^2} \ln p(y|f) = h''(f) [y - \exp(h(f))] - [h'(f)]^2 \exp(h(f)) \\
\frac{\partial^3}{\partial f^3} \ln p(y|f) = h'''(f) [y - \exp(h(f))] - 3h'(f) \cdot h''(f) \exp(h(f)) - [h'(f)]^3 \exp(h(f)) \\
\]

Note that if \( \ln \mu = h(f) \) is concave and \( \mu = g(f) \) is convex then the Poisson likelihood \( p(y|f) \) is log-concave in \( f \) which is the case for both exp and logistic.

### 4.12.3 Weibull Likelihood

Nonnegative data \( y \in \mathbb{R}_+ \) such as time-to-failure can be modeled in the GP framework using the Weibull distribution \( p(y) = \kappa / \lambda (y/\lambda)^{\kappa-1} e^{-(y/\lambda)^\kappa} \) with shape parameter \( \kappa > 0 \), scale parameter \( \lambda > 0 \), mean \( \mathbb{E}[y] = \lambda \gamma_1 = \mu \) where \( \gamma_j = \Gamma(1+j/\kappa) \), variance \( \mathbb{V}[y] = \lambda^2 \gamma_2 - \mu^2 = \mu^2 (\gamma_2 - \gamma_1^2) \), skewness \( \mathbb{S}[y] = (\gamma_3 - 3 \gamma_1 \gamma_2 + 2 \gamma_1^3) / (\gamma_2 - \gamma_1^2)^{3/2} \) and kurtosis \( \mathbb{K}[y] = (\gamma_4 - 4 \gamma_1 \gamma_3 + 12 \gamma_1^2 \gamma_2 - 3 \gamma_2^2 - 6 \gamma_1^4) / (\gamma_2 - \gamma_1^2)^2 \). Using the substitution \( \mu = \lambda \gamma_1 \iff 1/\lambda = \gamma_1/\mu \), we obtain

\[
p(y|f) = \gamma_1^{\kappa} \mu^{\kappa-1} \exp\left(-\left(\frac{y_1}{\mu}\right)^{\kappa}\right), \quad \mu = g(f) > 0 \\
\iff \ln p(y|f) = \ln \left(\gamma_1^{\kappa} \mu^{\kappa-1}\right) + (\kappa - 1) \ln \left(\frac{y_1}{\mu}\right) + \left(\frac{y_1}{\mu}\right)^{\kappa}.
\]

Note that the Weibull likelihood \( p(y|f) \) is log-concave in \( f \) neither for the exp nor for the logistic inverse link.

### 4.12.4 Gamma Likelihood

Nonnegative data \( y \in \mathbb{R}_+ \) can be modeled in the GP framework using the Gamma distribution \( p(y) = \theta^{-\alpha} \Gamma(\alpha) y^{\alpha-1} e^{-y/\theta} \) with shape parameter \( \alpha > 0 \), scale parameter \( \theta > 0 \), mean \( \mathbb{E}[y] = \alpha \theta = \mu \), variance \( \mathbb{V}[y] = \alpha \theta^2 = \mu^2/\alpha \), skewness \( \mathbb{S}[y] = 2/\sqrt{\alpha} \) and kurtosis \( \mathbb{K}[y] = 6/\alpha \). Using the substitution \( \mu = \alpha \theta \iff \alpha/\mu = 1/\theta \), we obtain

\[
p(y|f) = \alpha^\kappa y^{\kappa-1} \mu^{-\kappa} \exp\left(-\frac{y \alpha}{\mu}\right), \quad \mu = g(f) > 0 \\
\iff \ln p(y|f) = -\kappa \left(\ln \mu + \frac{y}{\mu}\right) - \ln Z_\alpha(y), \quad \ln Z_\alpha(y) = \ln \Gamma(\alpha) - \alpha \ln \alpha + (1 - \alpha) \ln y.
\]

Note that if \( \ln \mu = h(f) \) was convex and \( \mu = g(f) \) was concave then the Gamma likelihood \( p(y|f) \) would be log-concave in \( f \) which is not the case for both exp and logistic.
4.12.5 Exponential Likelihood

Nonnegative data \( y \in \mathbb{R}_+ \) can be modeled in the GP framework using the Exponential distribution \( p(y) = \theta^{-1} e^{-y/\theta} \) with scale parameter \( \theta > 0 \), mean \( \mathbb{E}[y] = \theta = \mu \), variance \( \mathbb{V}[y] = \mu^2 \), skewness \( \mathbb{S}[y] = 2 \) and kurtosis \( \mathbb{K}[y] = 6 \). We obtain

\[
p(y|f) = \mu^{-1} \exp \left( -\frac{y}{\mu} \right), \quad \mu = g(f) > 0
\]

\[
\Leftrightarrow \ln p(y|f) = -\ln \mu - \frac{y}{\mu}.
\]

Note that for \( \exp \) (but not for \( \logistic \)) the likelihood is log-concave. The exponential distribution corresponds to the Gamma distribution with \( \alpha = 1 \) and the Weibull distribution with \( \kappa = 1 \).

4.12.6 Inverse Gaussian Likelihood

Nonnegative data \( y \in \mathbb{R}_+^n \) can be modeled in the GP framework using the Inverse Gaussian distribution

\[
p(y) = \sqrt{\lambda/(2\pi y^3)} \exp(-\lambda(y - \mu)^2/(2\mu^2y))
\]

with shape parameter \( \lambda > 0 \), mean parameter \( \mu > 0 \), mean \( \mathbb{E}[y] = \mu \), variance \( \mathbb{V}[y] = \mu^3/\lambda \), skewness \( \mathbb{S}[y] = 3\sqrt{\mu/\lambda} \) and kurtosis \( \mathbb{K}[y] = 15\mu/\lambda \). We obtain

\[
p(y|f) = \sqrt{\frac{\lambda}{2\pi y^3}} \exp \left( -\frac{\lambda(y - \mu)^2}{2\mu^2 y} \right), \quad \mu = g(f) > 0
\]

\[
\Leftrightarrow \ln p(y|f) = -\frac{\lambda(y - \mu)^2}{2\mu^2 y} - \ln Z_\alpha(y), \quad \ln Z_\alpha(y) = -\frac{1}{2}(\ln \lambda - \ln 2\pi y^3).
\]

The inverse Gaussian likelihood is in general not log-concave in \( f \) for both \( \exp \) and \( \logistic \).

4.12.7 Beta Likelihood

Interval data \( y \in [0, 1]^n \) can be modeled in the GP framework using the Beta distribution \( p(y) = y^{\alpha-1}(1-y)^{\beta-1}/B(\alpha,\beta) \) with shape parameters \( \alpha, \beta > 0 \), mean \( \mathbb{E}[y] = \alpha/(\alpha + \beta) \) and variance \( \mathbb{V}[y] = \alpha\beta/[(\alpha + \beta)^2(\alpha + \beta + 1)] \) and \( 1/B(\alpha,\beta) = \Gamma(\alpha + \beta)/[\Gamma(\alpha)\Gamma(\beta)] \). Reparametrising using the mean parameter \( \mu = \mathbb{E}[y] = \alpha/(\alpha + \beta) \), the shape parameter \( \phi = \alpha + \beta \), the variance \( \mathbb{V}[y] = \mu(1 - \mu)/(1 + \phi) \) and hence

\[
p(y|f) = \frac{\Gamma(\phi)}{\Gamma(\mu\phi)\Gamma((1-\mu)\phi)} y^{\mu\phi-1}(1-y)^{(1-\mu)\phi-1}, \quad \mu = g(f) > 0
\]

\[
\Leftrightarrow \ln p(y|f) = \ln \Gamma(\phi) - \ln \Gamma(\mu\phi) - \ln \Gamma((1-\mu)\phi) - (\mu\phi - 1) \ln y + ((1-\mu)\phi - 1) \ln(1-y).
\]

The Beta likelihood is in general not log-concave in \( f \) for both \( \exp \) and \( \logistic \).
5 Mean Functions

A mean function \( m_\phi : \mathcal{X} \rightarrow \mathbb{R} \) (with hyperparameters \( \phi \)) of a GP \( f \) is a scalar function defined over the whole domain \( \mathcal{X} \) that computes the expected value \( m(x) = \mathbb{E}[f(x)] \) of \( f \) for the input \( x \).

5.1 Interface

In the GPML toolbox, a mean function \( m : \mathcal{X} \rightarrow \mathbb{R} \) needs to implement evaluation \( m = m_\phi(X) \) and first derivatives \( m_i = \frac{\partial}{\partial \phi_i} m \) with respect to the components \( i \) of the parameter \( \phi \in \Phi \) as detailed below.

```plaintext
(meanFunctions.m)
1 % mean functions to be use by Gaussian process functions. There are two
2 % different kinds of mean functions: simple and composite:
3 %
4 % simple mean functions:
5 %
6 % meanZero - zero mean function
7 % meanOne - one mean function
8 % meanConst - constant mean function
9 % meanLinear - linear mean function
10 % meanPoly - polynomial mean function
11 % meanDiscrete - precomputed mean for discrete data
12 % meanGP - predictive mean of another GP
13 % meanGPexact - predictive mean of a regression GP
14 % meanNN - nearest neighbor mean function
15 %
16 % composite covariance functions (see explanation at the bottom):
17 %
18 % meanScale - scaled version of a mean function
19 % meanPow - power of a mean function
20 % meanProd - products of mean functions
21 % meanSum - sums of mean functions
22 % meanMask - mask some dimensions of the data
23 % meanPref - difference mean for preference learning
24 %
25 % Naming convention: all mean functions are named "mean/mean*.m".
26 %
27 % 1) With no or only a single input argument:
28 %
29 % s = meanNAME or s = meanNAME(hyp)
30 %
31 % The mean function returns a string s telling how many hyperparameters hyp it
32 % expects, using the convention that "D" is the dimension of the input space.
33 % For example, calling "meanLinear" returns the string 'D'.
34 %
35 % 2) With two input arguments:
36 %
37 % m = meanNAME(hyp, x)
38 %
39 % The function computes and returns the mean vector where hyp are the
40 % hyperparameters and x is an n by D matrix of cases, where D is the dimension
41 % of the input space. The returned mean vector is of size n by 1.
42 %
43 % 3) With three input arguments:
```

35
We offer simple and composite mean functions producing new mean functions $m(x)$ from existing mean functions $\mu_j(x)$. All code files are named according to the pattern `mean/mean<NAME>.m` for simple identification. This modular specification allows to define affine mean functions $m(x) = c + a^\top x$ or polynomial mean functions $m(x) = (c + a^\top x)^2$. All currently available mean functions are summarised in the following table.

### Simple mean functions $m(x)$

<table>
<thead>
<tr>
<th>&lt;NAME&gt;</th>
<th>Meaning</th>
<th>$m(x)$</th>
<th>(\phi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>mean vanishes always</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>One</td>
<td>mean equals 1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Const</td>
<td>mean equals a constant</td>
<td>$c$</td>
<td>$c \in \mathbb{R}$</td>
</tr>
<tr>
<td>Linear</td>
<td>mean linearly depends on $x \in \mathcal{X} \subseteq \mathbb{R}^D$</td>
<td>$a^\top x$</td>
<td>$a \in \mathbb{R}^D$</td>
</tr>
<tr>
<td>Poly</td>
<td>mean polynomially depends on $x \in \mathcal{X} \subseteq \mathbb{R}^D$</td>
<td>$\sum_d a_d x^d$</td>
<td>$a \in \mathbb{R}^{D \times d}$</td>
</tr>
<tr>
<td>Discrete</td>
<td>precomputed mean for discrete data $x \in \mathcal{X} \subseteq \mathbb{N}$</td>
<td>$\mu_x$</td>
<td>$\mu \in \mathbb{R}^s$</td>
</tr>
<tr>
<td>GP</td>
<td>predictive mean of another GP</td>
<td>$\int y \cdot p(y</td>
<td>D, x) , dy$</td>
</tr>
<tr>
<td>GPexact</td>
<td>predictive mean of a regression GP</td>
<td>$\int y \cdot p(y</td>
<td>D, x) , dy$</td>
</tr>
<tr>
<td>NN</td>
<td>nearest neighbor for a set $(z_j, m_j) \in \mathcal{X} \times \mathbb{R}$</td>
<td>$m_i$, $i = \arg\min_j d(x, z_j)$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>

### Composite mean functions $[\mu_1(x), \mu_2(x), ..] \mapsto m(x)$

<table>
<thead>
<tr>
<th>&lt;NAME&gt;</th>
<th>Meaning</th>
<th>$m(x)$</th>
<th>(\phi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale</td>
<td>scale a mean</td>
<td>$\alpha \mu(x)$</td>
<td>$\alpha \in \mathbb{R}$</td>
</tr>
<tr>
<td>Sum</td>
<td>add up mean functions</td>
<td>$\sum_j \mu_j(x)$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>Prod</td>
<td>multiply mean functions</td>
<td>$\prod_j \mu_j(x)$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>Pow</td>
<td>raise a mean to a power</td>
<td>$\mu(x)^d$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>Mask</td>
<td>act on components $I \subseteq [1, 2, .., D]$ of $x \in \mathcal{X} \subseteq \mathbb{R}^D$ only</td>
<td>$\mu(x_I)$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>Pref</td>
<td>preference learning mean $x = [x_1; x_2], x_i \subseteq \mathbb{R}^{D/2}$</td>
<td>$\mu(x_1) - \mu(x_2)$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>

### 5.3 Usage of Implemented Mean Functions

Some code examples taken from `doc/usageMean.m` illustrate how to use simple and composite mean functions to specify a GP model.

Syntactically, a mean function \(mf\) is defined by

```matlab
mn := 'func' | @func // simple
mf := {mn} | {mn, {param, mf}} | {mn, {mf, .., mf}} // composite
```

i.e., it is either a string containing the name of a mean function, a pointer to a mean function or one of the former in combination with a cell array of mean functions and an additional list of parameters.

```matlab
⟨doc/usageMean.m⟩
```
% demonstrate usage of mean functions
%
% See also meanFunctions.m.
%
\[gpml\copyright\]

clear all, close all
n = 5; D = 2; x = randn(n,D); % create a random data set

% set up simple mean functions
m0 = {'meanZero'}; hyp0 = []; % no hyperparameters are needed
m1 = {'meanOne'}; hyp1 = []; % no hyperparameters are needed
mc = {@meanConst}; hypc = 2; % also function handles are possible
ml = {@meanLinear}; hypl = [2;3]; % m(x) = 2*x1 + 3*x2
mn = {@meanNN,[1,0; 0,1],[0.9,0.5]}; hypn = []; % nearest neighbor
s = 12; hypd = randn(s,1); % discrete mean with 12 hypers
md = {'meanDiscrete',s};

m0 = {'meanZero'}; hyp0 = []; % no hyperparameters are needed
m1 = {'meanOne'}; hyp1 = []; % no hyperparameters are needed
mc = {'meanConst'}; hypl = [2;3]; % also function handles are possible
ml = {'meanLinear'}; hypl = [2;3]; % m(x) = 2*x1 + 3*x2
mn = {'meanNN',[1,0; 0,1],[0.9,0.5]}; hypn = []; % nearest neighbor
s = 12; hypd = randn(s,1); % discrete mean with 12 hypers
md = {'meanDiscrete',s};

hypo = [0;0;log(0.1)]; % regression GP predictive mean
xt = randn(2*n,D); yt = sign(xt(:,1)-xt(:,2)); % training data
mg = {'meanGP',hyp,@infEP,'meanZero',@covSEiso,@likErf,xt,yt};
hype = [0;0;log(0.1)]; % regression GP predictive mean
xt = randn(2*n,D); yt = xt(:,1).*xt(:,2); % training data
me = {'meanGPexact','meanZero','covSEiso',xt,yt};

% set up composite mean functions
msc = {'meanScale',{m1}}; hypsc = [3;hypo]; % scale by 3
msu = {'meanSum',{m0,mc,ml}}; hypsu = [hypo;hyc;hypl]; % sum
mph = {'meanProd',{mc,ml}}; hyppr = [hyc;hypl]; % product
mpo = {'meanPow',3,msu}; hyppo = hypsu; % third power
mask = [false,true]; % mask excluding all but the 2nd component
mma = {'meanMask',mask,ml}; hypma = hypl(mask);
mpf = {'meanPref',ml}; hyppf = 2; % linear pref with slope 3

% 0) specify mean function
mean = md; hyp = hypd; x = randi([1,s],n,1);
mean = mn; hyp = hypn;
mean = mg; hyp = hypl;
mean = me; hyp = hype;
mean = m0; hyp = hypo;
mean = msu; hyp = hypsu;
mean = mph; hyp = hyppr;
mean = mpo; hyp = hyppo;
mean = mpf; hyp = hyppf;

% 1) query the number of parameters
feval(mean{:})

% 2) evaluate the function on x
feval(mean{:},hyp,x)

% 3) compute the derivatives w.r.t. to hyperparameter i
i = 2; feval(mean{:},hyp,x,i)
6 Covariance Functions

A covariance function $k_\psi : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ (with hyperparameters $\psi$) of a GP $f$ is a scalar function defined over the whole domain $\mathcal{X}^2$ that computes the covariance $k(x, x') = \mathbb{V}[f(x), f(x')] = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))]$ of $f$ between the inputs $x$ and $x'$.

6.1 Interface

Again, the interface is simple since only evaluation of the full covariance matrix $K = k_\psi(X)$ and its derivatives $K_i = \frac{\partial}{\partial \psi_i} K$ as well as cross terms $k_* = k_\psi(X, x_*)$ and $k_{**} = k_\psi(x_*, x_*)$ for prediction are required.

```matlab
 covFunctions.m
1 1% covariance functions to be use by Gaussian process functions. There are two 2% different kinds of covariance functions: simple and composite:
3 %
4 4% simple covariance functions:
5 5% covConst - covariance for constant functions
6 6% covCos - sine periodic covariance function (1d) with unit period
7 7% covLIN - linear covariance function without parameters
8 8% covLINard - linear covariance function with ARD
9 9% covLINiso - linear covariance function
10 10% covLINone - linear covariance function with bias
11 11% covMaternard - Matern covariance function with nu=1/2, 3/2 or 5/2 with ARD
12 12% covMaterniso - Matern covariance function with nu=1/2, 3/2 or 5/2
13 13% covNNone - neural network covariance function
14 14% covNoise - independent covariance function (i.e. white noise)
15 15% covPeriodic - smooth periodic covariance function (1d)
16 16% covPeriodicNoDC - as above but with zero DC component and properly scaled
17 17% covPoly - polynomial covariance function
18 18% covPPard - piecewise polynomial covariance function (compact support)
19 19% covPPiso - piecewise polynomial covariance function (compact support)
20 20% covRQard - rational quadratic covariance function with ARD
21 21% covRQiso - isotropic rational quadratic covariance function
22 22% covSEard - squared exponential covariance function with ARD
23 23% covSEiso - isotropic squared exponential covariance function
24 24% covSEisoU - same as above but without latent scale
25 25% covSEvlen - spatially varying lengthscale squared exponential
26 26% covSEfact - factor analysis squared exponential covariance function
27 27% covSM - spectral mixture covariance function
28 28% covGaborard - Gabor covariance function with ARD
29 29% covGaboriso - isotropic Gabor covariance function
30 30% covDiscrete - precomputed covariance for discrete data
31 %
32 32% composite (meta) covariance functions (see explanation at the bottom):
33 33% covScale - scaled version of a covariance function
34 34% covProd - products of covariance functions
35 35% covSum - sums of covariance functions
36 36% covADD - additive covariance function
37 37% covMask - mask some dimensions of the data
38 38% covPERard - make ARD stationary covariance periodic
39 39% covPERiso - make isotropic stationary covariance periodic
40 40% covPref - difference covariance for preference learning
41 %
42 42% special purpose (wrapper) covariance functions
43 43% covFITC - to be used in conjunction with infFITC* for large scale
```
% inference problems; any covariance can be wrapped by covFITC such that the FITC approximation is applicable
% covGrid - to be used in conjunction with infGrid* for large scale inference problems on grids resulting Kronecker structure
% Naming convention: all covariance functions are named "cov/cov*.m". A trailing
% "iso" means isotropic, "ard" means Automatic Relevance Determination, and
% "one" means that the distance measure is parameterized by a single parameter.
% The covariance functions are written according to a special convention where
% the exact behaviour depends on the number of input and output arguments
% passed to the function. If you want to add new covariance functions, you
% should follow this convention if you want them to work with the function gp.
% There are four different ways of calling the covariance functions:
%
% 1) With no (or one) input argument(s):
%    s = cov
%
% The covariance function returns a string s telling how many hyperparameters it
% expects, using the convention that "D" is the dimension of the input space.
% For example, calling "covRQard" returns the string '(D+2)'.
%
% 2) With two input arguments:
%    K = cov(hyp, x) equivalent to K = cov(hyp, x, [])
%
% The function computes and returns the covariance matrix where hyp are
% the hyperparameters and x is an n by D matrix of cases, where
% D is the dimension of the input space. The returned covariance matrix is of
% size n by n.
%
% 3) With three input arguments:
%    Ks = cov(hyp, x, xs)
%    kss = cov(hyp, xs, 'diag')
%
% The function computes test set covariances; kss is a vector of self covariances
% for the test cases in xs (of length ns) and Ks is an (n by ns) matrix of cross
% covariances between training cases x and test cases xs.
%
% 4) With four input arguments:
%    dKi = cov(hyp, x, [], i)
%    dKsi = cov(hyp, x, xs, i)
%    dkssi = cov(hyp, xs, 'diag', i)
%
% The function computes and returns the partial derivatives of the
% covariance matrices with respect to hyp(i), i.e. with
% respect to the hyperparameter number i.
% Covariance functions can be specified in two ways: either as a string
% containing the name of the covariance function or using a cell array. For
% example:
% cov = 'covRQard';
% cov = {'covRQard'};
% cov = {@covRQard};
are supported. Only the second and third form using the cell array can be used for specifying composite covariance functions, made up of several contributions. For example:

cov = {'covScale', {'covRQiso'}};
cov = {'covSum', {'covRQiso', 'covSEard', 'covNoise'}};
cov = {'covProd', {'covRQiso', 'covSEard', 'covNoise'}};
cov = {'covMask', {mask, 'covSEiso'}}
q=1; cov = {'covPPiso', q};
d=3; cov = {'covPoly', d};
cov = {'covADD', {[1, 2], 'covSEiso'}};
cov = @covFITC, {@covSEiso}, u}; where u are the inducing inputs

covspecifies a covariance function which is the sum of three contributions. To find out how many hyperparameters this covariance function requires, we do:
feval(cov{:})

which returns the string '3+(D+1)+1' (i.e. the 'covRQiso' contribution uses 3 parameters, the 'covSEard' uses D+1 and 'covNoise' a single parameter).

See also doc/usageCov.m.
6.2 Implemented Covariance Functions

Similarly to the mean functions, we provide a whole algebra of covariance functions \( k : \mathbb{X} \times \mathbb{X} \to \mathbb{R} \) with the same generic name pattern \( \text{cov/cov<NAME>}.m \) as before.

Besides a long list of simple covariance functions, we also offer a variety of composite covariance functions as shown in the following table.

<table>
<thead>
<tr>
<th>Simple covariance functions ( k(x,x') )</th>
<th>Meaning</th>
<th>( k(x,x') = )</th>
<th>( \Psi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>covariance vanishes always</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Eye</td>
<td>unit additive measurement noise</td>
<td>( k(x,x') = x \cdot x' )</td>
<td>0</td>
</tr>
<tr>
<td>Noise</td>
<td>additive measurement noise</td>
<td>( \sigma^2(x-x') )</td>
<td>( \sigma^2 )</td>
</tr>
<tr>
<td>Const</td>
<td>constant</td>
<td>( \sigma^2 )</td>
<td>( \sigma^2 )</td>
</tr>
<tr>
<td>Lin</td>
<td>linear, ( x \in \mathbb{R}^n )</td>
<td>( x \cdot x' )</td>
<td>0</td>
</tr>
<tr>
<td>LIsard</td>
<td>linear with diagonal weighting, ( x \in \mathbb{R}^n )</td>
<td>( x \cdot x' )</td>
<td>( \Lambda_{gg}^{-1} )</td>
</tr>
<tr>
<td>LIsiso</td>
<td>linear with isotropic weighting, ( x \in \mathbb{R}^n )</td>
<td>( x \cdot x' )</td>
<td>( \Lambda_{gg}^{-1} )</td>
</tr>
<tr>
<td>LInsD</td>
<td>linear with bias, ( x \in \mathbb{R}^n )</td>
<td>( (x' + 1)^2 )</td>
<td>( \Lambda_{gg}^{-1} )</td>
</tr>
<tr>
<td>Poly</td>
<td>polynomial covariance, ( x \in \mathbb{R}^n )</td>
<td>( \sigma^2 x'^m )</td>
<td>( \sigma^2 )</td>
</tr>
<tr>
<td>Exp</td>
<td>exponential covariance, ( x \in \mathbb{R}^n )</td>
<td>( \exp(-a</td>
<td>x-x'</td>
</tr>
</tbody>
</table>
| Materniso | Matern, \( x \in \mathbb{R}^n \), | \( \frac{2^{1-\nu} \Gamma(1+\nu)}{\pi^2 \nu (\nu-1)} (\frac{\sqrt{2 \nu}}{\Lambda_{gg}}) \frac{1}{\Lambda_{gg}^{-1}} (|x-x'|)^
u \) | \( \Lambda_{gg}^{-1} \) |
| Maternard | Matern, | \( \frac{2^{1-\nu} \Gamma(1+\nu)}{\pi^2 \nu (\nu-1)} (\frac{\sqrt{2 \nu}}{\Lambda_{gg}}) \frac{1}{\Lambda_{gg}^{-1}} (|x-x'|)^
u \) | \( \Lambda_{gg}^{-1} \) |
| Bessel | square exponential, \( x \in \mathbb{R}^n \) | \( \exp(-\frac{|x-x'|^2}{2 \sigma^2}) \) | \( \sigma^2 \) |
| BEvlen | spatially varying lengthscale square exponential, \( x \in \mathbb{R}^n \) | \( \sigma^2 \exp(-\frac{|x-x'|^2}{2 \sigma^2}) \) | \( \sigma^2 \) |
| BEfact | factor analysis squared exponential, \( x \in \mathbb{R}^n \) | \( \sigma^2 \exp(-\sum_{i=1}^p (x_i-x_i')^2) \) | \( \sigma^2 \) |
| BQuad | quadratic, \( x \in \mathbb{R}^n \) | \( \sigma^2 \left( 1 + \sum_{i=1}^p (x_i-x_i')^2 \right) \) | \( \sigma^2 \) |
| BBQuad | quadratic, \( x \in \mathbb{R}^n \) | \( \sigma^2 \left( 1 + \sum_{i=1}^p (x_i-x_i')^2 \right) \) | \( \sigma^2 \) |
| Periodic | periodic, \( x \in \mathbb{R} \) | \( \sigma^2 \cos \left( \frac{\pi}{L} |x-x'| \right) \) | \( \sigma^2 \) |
| Periodicchio | periodic, \( x \in \mathbb{R} \), re-scale and DC component removed | \( \sigma^2 \cos \left( \frac{\pi}{L} |x-x'| \right) \) | \( \sigma^2 \) |
| Cos | cosine, \( x \in \mathbb{R} \) | \( \sigma^2 \cos \left( \frac{\pi}{L} |x-x'| \right) \) | \( \sigma^2 \) |
| Pard | compact support, piecewise polynomial, \( x \in \mathbb{R}^n \), \( m \in \mathbb{N} \), \( p \in \mathbb{N} \) | \( \sigma^2 \left( \sum_{i=1}^m |x_i-x'_i|^p \right) \) | \( \sigma^2 \) |
| PPard | compact support, piecewise polynomial, \( x \in \mathbb{R}^n \), \( m \in \mathbb{N} \), \( p \in \mathbb{N} \) | \( \sigma^2 \left( \sum_{i=1}^m |x_i-x'_i|^p \right) \) | \( \sigma^2 \) |
| ARD | spectral mixture, \( x \in \mathbb{R}^n \), \( w \in \mathbb{R}^n_+ \), \( M \in \mathbb{N} \), \( V \in \mathbb{R}^{n \times n} \) | \( \sigma^2 \sum_{i=1}^m \left( \sum_{j=1}^n w_j \left( \sum_{k=1}^m \left( \sum_{l=1}^n \cos \left( \frac{\pi}{L} |x_i-x'_i| \right) \right) \right) \right) \) | \( \sigma^2 \) |
| ARD | spectral mixture, \( x \in \mathbb{R}^n \), \( w \in \mathbb{R}^n_+ \), \( M \in \mathbb{N} \), \( V \in \mathbb{R}^{n \times n} \) | \( \sigma^2 \sum_{i=1}^m \left( \sum_{j=1}^n w_j \left( \sum_{k=1}^m \left( \sum_{l=1}^n \cos \left( \frac{\pi}{L} |x_i-x'_i| \right) \right) \right) \right) \) | \( \sigma^2 \) |
| Gabor | anisotropic Gabor function, \( x \in \mathbb{R}^n \), \( \lambda \in \mathbb{R}^n_+ \) | \( \sigma^2 \left( \prod_{i=1}^n \left( \sum_{j=1}^m \left( \sum_{k=1}^n \cos \left( \frac{\pi}{L} |x_i-x'_i| \right) \right) \right) \right) \) | \( \sigma^2 \) |
| Gaboriso | isotropic Gabor function, \( x \in \mathbb{R}^n \), \( \lambda \in \mathbb{R}^n_+ \) | \( \sigma^2 \left( \prod_{i=1}^n \left( \sum_{j=1}^m \left( \sum_{k=1}^n \cos \left( \frac{\pi}{L} |x_i-x'_i| \right) \right) \right) \right) \) | \( \sigma^2 \) |
| Discrete | precomputed covariance for discrete data \( x \in \mathbb{N} \) | \( \sigma^2 \) | \( \sigma^2 \) |

The spectral mixture covariance \( \text{covSM} \) was introduced by Wilson & Adams \( \text{Gaussian Process Kernels for Pattern Discovery and Extrapolation}, \) ICML, 2013.

The periodic covariance functions \( \text{covPERiso} \) and \( \text{covPERard} \) start from a stationary isotropic or ARD covariance function that depends on the data only through a distance \( r^2 = (x - x')^\top A^{-2} (x - x') \) such as \( \text{covMatern*}, \text{covPP*}, \text{covRQ*}, \text{covSE*} \) where ** is ard|iso and turn them into a periodic covariance function by embedding the data \( x \in \mathbb{R}^D \) into a periodic high-dimensional space \( x_p = u(x) \in \mathbb{R}^{2D} \) by a function \( u(x) = 2\pi\text{diag}(p^{-1})x \).

The additive covariance function \( \text{covADD} \) starts from a one-dimensional covariance function \( \kappa (x_1, x_1', \psi_1) \) acting on a single component \( i \in [1, ..., D] \) of \( x \). From that, we define covariance functions \( \kappa_i (x_i, x_i', \psi_i) = \prod_{i=1}^D \kappa (x_1, x_1', \psi_1) \) acting on vector-valued inputs \( x_1 \). The sums of exponential size can efficiently be computed using the Newton-Girard formulae. Samples functions drawn from a GP with additive covariance are additive functions. The number of interacting variables \( |I| \) is a measure of how complex the additive functions are.
6.3 Usage of Implemented Covariance Functions

Some code examples taken from `doc/usageCov.m` illustrate how to use simple and composite covariance functions to specify a GP model.

Syntactically, a covariance function `cf` is defined by

```plaintext
  cv := 'func' | @func  // simple
  cf := {cv} | {cv, {param, cf}} | {cv, {cf, ..., cf}}  // composite
```

i.e., it is either a string containing the name of a covariance function, a pointer to a covariance function or one of the former in combination with a cell array of covariance functions and an additional list of parameters.

```plaintext
[doc/usageCov.m] 41
% demonstrate usage of covariance functions
2 %
3 % See also covFunctions.m.
4 %
5 clear all, close all
6 n = 5; D = 3; x = randn(n,D); xs = randn(3,D);  % create a data set
7 % set up simple covariance functions
8 cn = {'covNoise'}; sn = .1; hypn = log(sn);  % one hyperparameter
9 cc = {@covConst}; sf = 2; hypc = log(sf);  % function handles OK
10 ce = {@covEye}; hype = [];  % identity
11 cla = {'covLINard'}; L = rand(D,1); hypla = log(L);  % linear (ARD)
12 cli = {'covLINiso'}; l = rand(1); hypli = log(l);  % linear iso
13 clo = {@covLINone}; ell = .9; hyplo = log(ell);  % linear with bias
14 cp = {@covPoly,3}; c = 2; hypp = log([c;sf]);  % third order poly
15 cga = {@covSEard}; hypga = log([L;sf]);  % Gaussian with ARD
16 cgl = {'covSEiso'}; hypgi = log([ell;sf]);  % isotropic Gaussian
17 cgu = {'covSEisoU'}; hypgu = log([ell;sf]);  % isotropic Gauss no scale
18 cra = {'covRQard'}; al = 2; hypra = log([ell;p;al]);  % ration. quad.
19 cri = {'covRQiso'}; hypri = log([ell;p;sf]);  % isotropic
20 cma = {'covMaternard',5}; hypma = log([ell;sf]);  % Matern class d=5
21 cmi = {'covMaterniso',3}; hypmi = log([ell;sf]);  % Matern class d=3
22 cnn = {'covNNone'}; hypnn = log([L;sf]);  % neural network
23 cpe = {'covPeriodic'}; p = 2; hypp = log([ell;p;sf]);  % periodic
24 cph = {'covPeriodicNoDC'}; p = 2; hypp = log([ell;p;sf]);  % w/o DC
25 cpc = {'covCos'}; p = 2; hypp = log([ell;p;sf]);  % cosine cov
26 cca = {'covPPard',3}; hypcc = hypgu;  % compact support poly degree 3
27 occ = {'covPPiso',2}; hypcc = hypgi;  % compact support poly degree 2
28 cgb = {'covGaboriso'}; ell = 1; p = 1.2; hypgb = log([ell;p]);  % Gabor
29 Q = 2; w = ones(Q,1)/Q; m = randn(D,Q); v = randn(D,Q);  % factor analysis
30 csm = {'covSM,Q'}; hypsm = log([w;v(:)]);  % Spectral Mixture
31 cvl = {'covSEVlen',meanLinear}; hypvl = [1 2 1];  % var lenscal
32 s = 12; cds = @covDiscrete,s;  % discrete covariance function
33 L = randn(s); L = chol(L*L); L([1:(s+1):end]) = log(diag(L));
34 hypds = L(triu(true(s))); xd = randn([1,s],[n,1]); xsd = [1:3;6];
35 cfa = {'covSEfact',2}; hypfa = randn(D*2,1);  % factor analysis
36 % set up composite i.e. meta covariance functions
37 csc = {'covScale',[cgu]}; hypsc = [log(3); hypgu];  % scale by 9
38 csu = {'covSum',[cn,cc,ccl]}; hypsu = [hypn, hypc, hyp1];  % sum
39 cpr = {'covProd',[cc,cci]}; hypp = [hypc, hypcc];  % product
40 mask = [0,1,0];  % binary mask excluding all but the 2nd component
```
7 Hyperpriors

A hyperprior \(p(\theta)\) with \(\theta = [\rho, \phi, \psi]\) is a joint probability distribution over the likelihood hyperparameters \(\rho\), the mean hyperparameters \(\phi\) and the covariance hyperparameters \(\psi\). We concentrate on factorial priors \(p(\theta) = \prod_j p_j(\theta_j)\). Hyperpriors can be used to regularise the optimisation of the hyperparameters via the marginal likelihood \(Z(\theta)\) so that \(p(\theta)Z(\theta)\) is maximised instead. As we wish to perform unconstrained optimisation, we require (mainly) smooth hyperpriors with infinite support.

7.1 Interface

In the GPML toolbox, a prior distribution \(p(\theta)\) needs to implement the evaluation of the log density \(\ln p(\theta)\) and its first derivative \(\frac{\partial}{\partial \theta} \ln p(\theta)\). In addition, we require sampling capabilities i.e. the generation of \(\theta \sim p(\theta)\).

42 \langle priorDistributions.m \rangle 42 
1 \% prior distributions to be used for hyperparameters of Gaussian processes
2 \% using infPrior.
3 \% There are two different kinds of prior distributions: simple and composite:
4 \%
5 \% simple prior distributions:
priorGauss - univariate Gaussian
priorLaplace - univariate Laplace
priorT - univariate Student’s t
priorSmoothBox1 - univariate interval (linear decay in log domain)
priorSmoothBox2 - univariate interval (quadr. decay in log domain)
priorGamma - univariate Gamma, IR+
priorWeibull - univariate Weibull, IR+
priorInvGauss - univariate Inverse Gaussian, IR+
priorLogNormal - univariate Log-normal, IR+
priorClamped or - fix hyperparameter to its current value by setting derivatives to zero, no effect on marginal likelihood
priorGaussMulti - multivariate Gauss
priorLaplaceMulti - multivariate Laplace
priorTMulti - multivariate Student’s t
priorClampedMulti or - fix hyperparameter to its current value by setting derivatives to zero, no effect on marginal likelihood
priorMix - nonnegative mixture of priors
priorTransform - prior on g(t) rather than t
Naming convention: all prior distributions are named "prior/prior*.m".
1) With only a fixed input arguments:
r = priorNAME(par1,par2,parN)
The function returns a random sample from the distribution for e.g. random restarts, simulations or optimisation initialisation.
2) With one additional input arguments:
[lp,dlp] = priorNAME(par1,par2,parN, t)
The function returns the log density at location t along with its first derivative.
See also doc/usagePrior.m, inf/infPrior.m.
7.2 Implemented Hyperpriors

All code files are named according to the pattern `prior/prior<NAME>.m` for simple identification. All currently available hyperpriors are summarised in the following table.

<table>
<thead>
<tr>
<th>Simple hyperpriors p(θ)</th>
<th>Univariate hyperpriors defined over the whole reals with mean µ and variance σ²</th>
<th>Multivariate hyperpriors supported all over ( \mathbb{R}^D ) with mean µ and covariance ( \Sigma )</th>
<th>Improper hyperpriors used to fix the value of a particular hyperparameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gauss</td>
<td>normally distributed hyperparameter ( \theta \in \mathbb{R} ) ( p(\theta) = 1/\sqrt{2\pi\sigma^2} \exp\left(-\frac{(\theta-\mu)^2}{2\sigma^2}\right) ) ( \mu \in \mathbb{R}, \sigma^2 \in \mathbb{R}_+ )</td>
<td>multivariate normal distribution ( \theta \in \mathbb{R}^D ) ( p(\theta) = 1/(\sqrt{2\pi}^D\sigma) \exp\left(-\frac{1}{2}(\theta-\mu)^\top \Sigma^{-1}(\theta-\mu)\right) ) ( \mu \in \mathbb{R}^D, \Sigma \in \mathbb{R}^{D\times D} )</td>
<td>Delta clamped hyperparameter ( \theta = 0_0 \in \mathbb{R} ) ( p(\theta) = \delta(\theta-0_0) )</td>
</tr>
<tr>
<td>Laplace</td>
<td>double exponentially distributed hyperparameter ( \theta \in \mathbb{R} ) ( p(\theta) = \frac{1}{\beta^\alpha \Gamma(\alpha)} \exp\left(-\frac{\beta</td>
<td>\theta</td>
<td>}{\alpha}\right), \beta = \sigma \sqrt{2} ) ( \mu \in \mathbb{R}, \sigma \in \mathbb{R}_+ )</td>
</tr>
<tr>
<td>T</td>
<td>Student's t-distributed hyperparameter ( \theta \in \mathbb{R} ) ( p(\theta) = \Gamma\left(1 + \frac{k}{2}\right)\Gamma\left(\frac{k}{2}\right)^{-1} \frac{1}{\sqrt{k\pi\sigma^2}} \left(1 +\frac{\theta-\mu}{k\sigma^2}\right)^{-\frac{k+1}{2}} ) ( \mu \in \mathbb{R}, \sigma^2 \in \mathbb{R}_+ )</td>
<td>multivariate Student's t distribution ( \theta \in \mathbb{R}^D ) ( p(\theta) = \Gamma\left(1 + \frac{k}{2}\right)\Gamma\left(\frac{k}{2}\right)^{-1} \frac{1}{\sqrt{k\pi\sigma^2}} \left(1 +\frac{\theta-\mu}{k\sigma^2}\right)^{-\frac{k+1}{2}} ) ( \mu \in \mathbb{R}^D, \sigma^2 \in \mathbb{R}_+ )</td>
<td></td>
</tr>
</tbody>
</table>

The priorSmoothBox2 is a Gauss-uniform sandwich obtained by complementing a uniform distribution on \([a, b]\) with two Gaussian halves at each side. The parameter \( \eta \) balances the probability mass between the constituents so that \( \eta/(\eta + 1) \) is used for the box and \( 1/(\eta + 1) \) for the Gaussian sides. Its brother priorSmoothBox1 is the product of two sigmoidal functions.

The priorDelta or equivalently priorClamped can be used to exclude some hyperparameters from the optimisation. Their values are clamped to \( 0_0 \) and the derivative vanishes. There are also multivariate counterparts priorDeltaMulti and priorClampedMulti.

7.3 Usage of Implemented Hyperpriors

Some code examples taken from `doc/usagePrior.m` illustrate how to use univariate, multivariate and composite priors on hyperparameters. Syntactically, a hyperprior \( hp \) is defined by

```plaintext
func := Dist            // prior distributions in prior/
   | Clamped | Delta // predefined for fixing the hyperparameter
pr  := 'func' | @func   // univariate hyperprior
   | 'funcMulti' | @funcMulti // multivariate hyperprior
```

45
hp := {pr} | {pr, {param, hp}} | {pr, {hp, ..., hp}} // composite

i.e., it is either a string containing the name of a hyperprior function, a pointer to a hyperprior function or one of the former in combination with a cell array of hyperprior functions and an additional list of parameters. Furthermore, we have multivariate hyperprior variants and 2 (equivalent) predefined hyperpriors allowing to exclude variables from optimisation.

```matlab
45 ⟨doc/usagePrior.m⟩45≡
1% demonstrate usage of prior distributions
2%
3% See also priorDistributions.m.
4%
5 ⟨gpml copyright⟩6a
6clear all, close all
7
88% 1) specify some priors
9% a) univariate priors
10mu = 1.0; s2 = 0.01^2; nu = 3;
11pg = {@priorGauss,mu,s2}; % Gaussian prior
12pl = {'priorLaplace',mu,s2}; % Laplace prior
13pt = {@priorT,mu,s2,nu}; % Student’s t prior
14p1 = {@priorSmoothBox1,0,3,15}; % smooth box constraints lin decay
15p2 = {@priorSmoothBox2,0,2,15}; % smooth box constraints qua decay
16pd = {'priorDelta'}; % fix value of prior exclude from optimisation
17pc = {@priorClamped}; % equivalent to above
18lam = 1.05; k = 2.5;
19pw = {@priorWeibull,lam,k}; % Weibull prior
20
21% b) meta priors
22pmx = {@priorMix,[0.5,0.5],{pg,pl}}; % mixture of two priors
23g = @exp; dg = @exp; ig = @log;
24ptr = {@priorTransform,g,dg,ig,pg}; % Gaussian in the exp domain
25
26% c) multivariate priors
27m = [1;2]; V = [2,1;1,2];
28pG = {@priorGaussMulti,m,V}; % 2d Gaussian prior
29pD = {'priorDeltaMulti'}; % fix value of prior exclude from optim
30pC = {@priorClampedMulti}; % equivalent to above
31
32% 2) evaluation
33% pri = pt; hp = randn(1,3);
34% pri = pmx; hp = randn(1,3);
35% pri = ptr; hp = randn(1,3);
36% pri = pG; hp = randn(2,3);
37
38% a) draw a sample from the prior
39feval(pri{:})
40
41% b) evaluate prior and derivative if requires
42[lp,dlp] = feval(pri{:},hp)
43
44% 3) comprehensive example
45x = (0:0.1:10)'; y = 2*x+randn(size(x)); % generate training data
46mean = {@meanSum,{@meanConst,@meanLinear}}; % specify mean function
47cov = {@covSEiso}; lik = {@likGauss}; % specify covariance and lik
48hyp.cov = [log(1);log(1.2)]; hyp.lik = log(0.9); hyp.mean = [2;3];
49par = {mean,cov,lik,x,y}; mfun = @minimize; % input for GP function
50```
% a) plain marginal likelihood optimisation (maximum likelihood)
im = @infExact; % inference method
hyp_plain = feval(mfun, hyp, @gp, -10, im, par{:}); % optimise

% b) regularised optimisation (maximum a posteriori) with 1d priors
prior.mean = {pg;pc}; % Gaussian prior for first, clamp second par
prior.cov = {p1;[]}; % box prior for first, nothing for second par
im = {@infPrior,@infExact,prior}; % inference method
hyp_p1 = feval(mfun, hyp, @gp, -10, im, par{:}); % optimise

% c) regularised optimisation (maximum a posteriori) with Nd priors
prior = [];
% multivariate Student’s t prior on the first and second mean hyper
prior.multi{1} = {@priorTMulti,mu,s2,nu,...
    struct('mean',[1,2])}; % use hyper struct
% Equivalent shortcut (same mu and s2 for all dimensions)
prior.multi{1} = {@priorTMulti,mu,s2,nu,struct('mean',[1,2])};
% multivariate Gaussian prior jointly on 1st and 3rd hyper
prior.multi{2} = {@priorGaussMulti,mu,s2,[1,3]}; % use unwrapped hyper vector
% Equivalent shortcut (same mu and s2 for all dimensions)
prior.multi{2} = {@priorGaussMulti,mu,s2,[1,3]};
im = {@infPrior,@infExact,prior}; % inference method
hyp_pN = feval(mfun, hyp, @gp, -10, im, par{:}); % optimise

[unwrap(hyp), unwrap(hyp_plain), unwrap(hyp_p1), unwrap(hyp_pN)]