GPML Toolbox by C.E. Rasmussen and H. Nickisch

- Toolbox web site:
  http://www.gaussianprocess.org/gpml/code
- Runs both on Octave 3.2.x and on Matlab 7.x.
- Originally used to demonstrate the main algorithms from book by Rasmussen and Williams
  Gaussian Processes for Machine Learning
  (book freely available for download)
GPML Toolbox

Overview

• A flexible framework for specifying GPs
  • Different mean function and covariance functions.
  • Modular design easily allowing extension for existing libraries.

• Allows different likelihood functions:
  • Gaussian (regression)
  • Laplace (regression)
  • Cumulative logistic (classification)

• Various inference methods:
  • Exact inference (regression)
  • Laplace approximation (both r&c)
  • Expectation propagation (both r&c)
  • Variational Bayes (both r&c)
GPML Toolbox

Supporting structures and functions

• The toolbox contains a single user function \texttt{gp}.

• A number of supporting structures and functions
  • \textbf{Inference Methods} - for a given model specification and a dataset
    • computes the (approximate) posterior,
    • the (approximate) negative log marginal likelihood and its partial derivatives w.r.t. the hyperparameters.
  • \textbf{Hyperparameters}
    • a struct with three fields controlling the properties of the model
      • Likelihood functions
      • Mean functions
      • Covariance functions
GPML Toolbox

Specifying model properties

- **Likelihood function**
  - specifies the form of the likelihood of the GP model
  - and computes terms needed for prediction and inference.

- **Mean function**
  - a cell array specifying the GP mean.
  - Computes the mean and its derivatives w.r.t. the part of the hyperparameters pertaining to the mean.

- **Covariance Function**
  - a cell array specifying the GP covariance function.
  - Computes the covariance and its derivatives w.r.t. the part of the hyperparameters pertaining to the covariance function.
Inference Methods

- **Exact** GP inference reduces to computing mean and covariance of a multivariate Gaussian.
  - for Gaussian likelihoods.

- **Laplace’s approximation** approximates the posterior by a Gaussian centered at its mode and matching its curvature.
  - for differentiable likelihoods.

- **Expectation Propagation (EP)** approximates the posterior by a Gaussian via moment matching.

- **Variational Bayes** constructs a joint lower bound on the marginal likelihood based on individual lower bounds to every likelihood function.
  - the maximization problem is concave for log-concave likelihoods.
Likelihood functions

- Implemented likelihood functions

| <NAME>   | classification $y_i \in \{\pm 1\}$ | $P_\rho(y_i|f_i) =$ | $\rho =$ |
|--------------|-----------------------------------|--------------------|---------|
| Gauss        | regression $y_i \in \mathbb{R}$   | $\mathbb{N}(y_i|f_i, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma}} \exp \left(-\frac{(y_i-f_i)^2}{2\sigma^2}\right)$ | $\ln \sigma$ |
| Sech2        | Sech-squared                      | $\frac{\tau}{2 \cosh^2(\tau(y_i-f_i))}, \tau = \frac{\pi}{2\sigma\sqrt{3}}$ | $\ln \sigma$ |
| Laplace      | Laplacian                         | $\frac{1}{2b} \exp \left(-\frac{|y_i-f_i|}{b}\right), b = \frac{\sigma}{\sqrt{2}}$ | $\ln \sigma$ |
| T            | Student’s t                       | $\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \frac{1}{\sqrt{\nu\pi\sigma}} \left(1 + \frac{(y_i-f_i)^2}{\nu\sigma^2}\right)^{-\frac{\nu+1}{2}}$ | $\ln(\nu - 1), \ln \sigma$ |

| <NAME> |                                     | $\mathbb{P}_\rho(y_i|f_i) =$ | $\rho =$ |
|--------------|-----------------------------------|--------------------|---------|
| Erf          | Error function                    | $\int_{-\infty}^{y_i} \mathbb{N}(t) dt$ | $\emptyset$ |
| Logistic     | Logistic function                 | $\frac{1}{1 + \exp(-y_i f_i)}$ | $\emptyset$ |
## GPML Toolbox

### Compatibility matrix

<table>
<thead>
<tr>
<th>Likelihood \ Inference</th>
<th>Exact</th>
<th>EP</th>
<th>Laplace</th>
<th>Variational Bayes</th>
<th>regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>regression</td>
</tr>
<tr>
<td>Sech-squared</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>regression</td>
</tr>
<tr>
<td>Laplacian</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>regression</td>
</tr>
<tr>
<td>Student’s t</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>regression</td>
</tr>
<tr>
<td>Error function</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>probit regression</td>
</tr>
<tr>
<td>Logistic function</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>logit regression</td>
</tr>
</tbody>
</table>
Mean functions

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

<table>
<thead>
<tr>
<th>&lt;NAME&gt;</th>
<th>Meaning</th>
<th>$m(x) =$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>mean vanishes always</td>
<td>0</td>
</tr>
<tr>
<td>One</td>
<td>mean equals 1</td>
<td>1</td>
</tr>
<tr>
<td>Const</td>
<td>mean equals a constant</td>
<td>$c$</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>
</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>
</tr>
</thead>
<tbody>
<tr>
<td>Scale</td>
<td>scale a mean</td>
<td>$\alpha \mu(x)$</td>
</tr>
<tr>
<td>Sum</td>
<td>add up mean functions</td>
<td>$\sum_j \mu_j(x)$</td>
</tr>
<tr>
<td>Prod</td>
<td>multiply mean functions</td>
<td>$\prod_j \mu_j(x)$</td>
</tr>
<tr>
<td>Pow</td>
<td>raise a mean to a power</td>
<td>$\mu(x)^d$</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>
</tr>
</tbody>
</table>
# Covariance functions

## Simple

<table>
<thead>
<tr>
<th>Simple covariance functions $k(x, x')$</th>
<th>Meaning</th>
<th>$k(x, x') =$</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;NAME&gt;</td>
<td>Mean vanishes always</td>
<td>0</td>
</tr>
<tr>
<td>Noise</td>
<td>Additive measurement noise</td>
<td>$\sigma_f^2 \delta(x - x')$</td>
</tr>
<tr>
<td>Const</td>
<td>Covariance equals a constant</td>
<td>$\sigma_f^2$</td>
</tr>
<tr>
<td>LIN</td>
<td>Linear, $\mathcal{X} \subseteq \mathbb{R}^D$</td>
<td>$x' x'$</td>
</tr>
<tr>
<td>LINard</td>
<td>Linear with diagonal weighting, $\mathcal{X} \subseteq \mathbb{R}^D$</td>
<td>$x' \Lambda^{-2} x'$</td>
</tr>
<tr>
<td>LINnone</td>
<td>Linear with bias, $\mathcal{X} \subseteq \mathbb{R}^D$</td>
<td>$(x' x' + 1)/\ell^2$</td>
</tr>
<tr>
<td>Poly</td>
<td>Polynomial covariance, $\mathcal{X} \subseteq \mathbb{R}^D$</td>
<td>$\sigma_f^2 (x' x' + c)^d$</td>
</tr>
<tr>
<td>SEard</td>
<td>Full squared exponential, $\mathcal{X} \subseteq \mathbb{R}^D$</td>
<td>$\sigma_f^2 \exp\left(-\frac{1}{2}(x - x')^T \Lambda^{-2} (x - x')\right)$</td>
</tr>
<tr>
<td>SEiso</td>
<td>Diagonal squared exponential, $\mathcal{X} \subseteq \mathbb{R}^D$</td>
<td>$\sigma_f^2 \exp\left(-\frac{1}{2\alpha}(x - x')^T (x - x')\right)$</td>
</tr>
<tr>
<td>SEisoU</td>
<td>Squared exponential, $\mathcal{X} \subseteq \mathbb{R}^D$</td>
<td>$\exp\left(-\frac{2}{\ell^2} x' x'\right)$</td>
</tr>
<tr>
<td>RQard</td>
<td>Rational quadratic, $\mathcal{X} \subseteq \mathbb{R}^D$</td>
<td>$\sigma_f^2 \left(1 + \frac{1}{2\alpha}(x - x')^T \Lambda^{-2} (x - x')\right)^{-\alpha}$</td>
</tr>
<tr>
<td>RQiso</td>
<td>Rational quadratic, $\mathcal{X} \subseteq \mathbb{R}^D$</td>
<td>$\sigma_f^2 \left(1 + \frac{1}{2\alpha}(x - x')^T (x - x')\right)^{-\alpha}$</td>
</tr>
<tr>
<td>Materniso</td>
<td>Matérn, $\mathcal{X} \subseteq \mathbb{R}^D$, $f_1(t) = 1$, $f_3(t) = 1 + t$, $f_5(t) = f_3(t) + \frac{t^2}{3}$</td>
<td>$\sigma_f^2 f_d(r_d) \exp(-r_d)$, $r_d = \sqrt{\frac{d}{\ell^2}} (x - x')^T (x - x')$</td>
</tr>
<tr>
<td>NNone</td>
<td>Neural net, $\mathcal{X} \subseteq \mathbb{R}^D$, $f(x) = 1 + x^T \Lambda^{-2} x$</td>
<td>$\sigma_f^2 \sin^{-1}\left(\frac{x^T \Lambda^{-2} x'}{\sqrt{f(x) f(x')}}\right)$</td>
</tr>
<tr>
<td>Periodic</td>
<td>Periodic, $\mathcal{X} \subseteq \mathbb{R}$</td>
<td>$\sigma_f^2 \exp\left(-\frac{2}{\ell^2 \sin^2\left[\frac{x}{2\pi}(x - x')\right]}\right)$</td>
</tr>
<tr>
<td>PPiso</td>
<td>Compact support, piecewise polynomial $f_v(r)$, $\mathcal{X} \subseteq \mathbb{R}$,</td>
<td>$\sigma_f^2 \max(0, 1 - r) \cdot f_v(r)$, $r = \frac{</td>
</tr>
</tbody>
</table>
# Covariance functions

## Composite

<table>
<thead>
<tr>
<th>&lt;NAME&gt;</th>
<th>Meaning</th>
<th>( k(x, x') = )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale</td>
<td>scale a covariance</td>
<td>( \alpha k(x, x') )</td>
</tr>
<tr>
<td>Sum</td>
<td>add up covariance functions</td>
<td>( \sum_j \kappa_j(x, x') )</td>
</tr>
<tr>
<td>Prod</td>
<td>multiply covariance functions</td>
<td>( \prod_j \kappa_j(x, x') )</td>
</tr>
<tr>
<td>Mask</td>
<td>act on components ( I \subseteq [1, 2, \ldots, D] ) of ( x \in \mathcal{X} \subseteq \mathbb{R}^D ) only</td>
<td>( \kappa(x_I, x_I') )</td>
</tr>
<tr>
<td>ADD</td>
<td>additive, ( \mathcal{X} \subseteq \mathbb{R}^D ), index degree set ( D = {1, \ldots, D} )</td>
<td>( \sum_{d \in D} \sigma_d^2 \prod_{i=1}^{|I|=d} \kappa(x_i, x_i'; \psi_i) )</td>
</tr>
</tbody>
</table>
The gp function
Overview

\[ \text{varargout} = \text{gp}(\text{hyp, inf, mean, cov, lik, x, y, xs, ys}) \]

- hyp  column vector of hyperparameters
- inf  function specifying the inference method
- cov  prior covariance function (see below)
- mean prior mean function
- lik  likelihood function
- x    n by D matrix of training inputs
- y    column vector of length n of training targets
- xs   ns by D matrix of test inputs
- ys   column vector of length nn of test targets
The gp function
Overview

\[ \text{varargout} = \text{gp}(\text{hyp}, \text{inf}, \text{mean}, \text{cov}, \text{lik}, x, y, xs, ys) \]

- \text{nlZ} returned value of the negative log marginal likelihood
- \text{dnlZ} column vector of partial derivatives of the negative log marginal likelihood w.r.t. each hyperparameter
- \text{ymu} column vector (of length ns) of predictive output means
- \text{ys2} column vector (of length ns) of predictive output variances
- \text{fmu} column vector (of length ns) of predictive latent means
- \text{fs2} column vector (of length ns) of predictive latent variances
- \text{lp} column vector (of length ns) of log predictive probs
- \text{post} struct representation of the (approximate) posterior
  3rd output in \text{training} and 6th output in \text{prediction} mode
The gp function

Overview

function [varargout] = gp(hyp, inf, mean, cov, lik, x, y, xs, ys)
    <gp function help >
    <initializations >
    <inference >
    if nargin==7 % if no test cases are provided
        varargout = {nlZ, dnlZ, post}; % report -log marg lik, derivs and post
    else
        <compute test predictions >
    end
The gp function

Process input arguments

if isempty(inf), inf = @infExact; else  % set default inf
    if iscell(inf), inf = inf{1}; end       % cell input is allowed
    if ischar(inf), inf = str2func(inf); end % convert into function handle
end

if isempty(mean), mean = {@meanZero}; end % set default mean
if ischar(mean) || isa(mean, 'function_handle'), mean = {mean}; end % make cell
if isempty(cov), error(' Covariance function cannot be empty'); end % no default
if ischar(cov) || isa(cov, 'function_handle'), cov = {cov}; end % make cell
cov1 = cov{1}; if isa(cov1, 'function_handle'), cov1 = func2str(cov1); end
if strcmp(cov1,'covFITC'), inf = @infFITC; end % only one possible inf alg
if isempty(lik), lik = @likGauss; else  % set default lik
    if iscell(lik), lik = lik{1}; end       % cell input is allowed
    if ischar(lik), lik = str2func(lik); end % convert into function handle
End
D = size(x,2);
The gp function
Check & initialize hyperparameters

if ~isfield(hyp,'mean'), hyp.mean = []; end
if eval(feval(mean{:})) ~= numel(hyp.mean)
    error('Number of mean function hyperparameters disagree with mean function')
end
if ~isfield(hyp,'cov'), hyp.cov = []; end
if eval(feval(cov{:})) ~= numel(hyp.cov)
    error('Number of cov function hyperparameters disagree with cov function')
end
if ~isfield(hyp,'lik'), hyp.lik = []; end
if eval(feval(lik)) ~= numel(hyp.lik)
    error('Number of lik function hyperparameters disagree with lik function')
end
The gp function

Do inference — issue a warning if it fails in training mode & try to recover

% issue a warning if a classification likelihood is used
% in conjunction with labels different from +1 and -1
if strcmp (func2str (lik), 'likErf') || strcmp (func2str (lik), 'likLogistic')
    uy = unique(y);
    if any ( uy~=+1 & uy~=-1 )
        warning('You attempt classification using labels different from \{+1,-1\\n\}');
    end
end
if nargin >7
    % compute marginal likelihood and its derivatives only if needed
    post = inf(hyp, mean, cov, lik, x, y);
else
    if nargout ==1
        [post nlZ] = inf(hyp, mean, cov, lik, x, y); dnlZ = {};
    else
        [post nlZ dnlZ] = inf(hyp, mean, cov, lik, x, y);
    end
end
The gp function
Compute test predictions

<handle sparse representations if applicable>
<compute necessary matrices if not provided>
<set mini-batch size> % prediction in mini-batches to avoid memory problems
<allocate memory>
<make predictions>
<assign output arguments>
The gp function
Make predictions – for all test points in a mini-batch

\texttt{kss = feval(cov{:}, hyp.cov, xs(id,:), 'diag');} \quad \% \text{self-variance}
\texttt{Ks = feval(cov{:}, hyp.cov, x(nz,:), xs(id,:));} \quad \% \text{cross-covariances}
\texttt{ms = feval(mean{:}, hyp.mean, xs(id,:));}
\texttt{fmu(id) = ms + Ks'*full(alpha(nz));} \quad \% \text{predictive means}
\textbf{if} \ Ltril \quad \% \text{L is triangular => use Cholesky parameters (alpha,sW,L)}
\quad \texttt{V = L'(repmat(sW,1,length(id)).*Ks);}
\quad \texttt{fs2(id) = kss - sum(V.*V,1)'}; \quad \% \text{predictive variances}
\textbf{else} \quad \% \text{L is not triangular => use alternative parametrisation}
\quad \texttt{fs2(id) = kss + sum(Ks.*(L*Ks),1)'}; \quad \% \text{predictive variances}
\textbf{end}
\texttt{fs2(id) = max(fs2(id),0);} \quad \% \text{remove numerical noise i.e. negative variances}
\textbf{if} \ nargin<9
\quad \texttt{[lp(id) ymu(id) ys2(id)] = lik(hyp.lik, [], fmu(id), fs2(id));} \quad \textbf{else}
\quad \texttt{[lp(id) ymu(id) ys2(id)] = lik(hyp.lik, ys(id), fmu(id), fs2(id));}
\textbf{end}
Inference Methods

infExact.m

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
L = chol(K/sn2+eye(n));  \% Cholesky of covariance with noise
alpha = solve_chol(L,y-m)/sn2;
post.alpha = alpha;  \% return the posterior parameters
post.sW = ones(n,1)/sqrt(sn2);  \% sqrt of noise precision vector
post.L = L;  \% L = chol(eye(n)+sW*sW'.*K)

if nargout>1,
    nI = (y-m)'*alpha/2 + sum(log(diag(L))) + n*log(2*pi*sn2)/2;  \% -log marg lik
    if nargout>2, dnlI = hyp;  \% derivatives
        Q = solve_chol(L,eye(n))/sn2 - alpha*alpha';  \% precompute for convenience
        for i = 1:numel(hyp.cov)
            dnlI.cov(i) = sum(sum(Q.*feval(cov{:}, hyp.cov, x, [], i)))/2;  end
        dnlI.lik = sn2*trace(Q);
        for i = 1:numel(hyp.mean),
            dnlI.mean(i) = -feval(mean{:}, hyp.mean, x, i)*alpha;  end
    end, end
Marginal Likelihood vs Cross Validation

\[ L = \sum_{i=1}^{n} \log p(y_i | \{y_j, j < i\}, \theta) \]

\[ L_{LOO} = \sum_{i=1}^{n} \log p(y_i | \{y_j, j \neq i\}, \theta) \]

- Marginal likelihood gives the probability of the data given the model assumptions.
- LOO-CV gives an estimate of the predictive log-probability regardless of the model assumptions being fulfilled.
  - More robust against model misspecification.