GAUSSIAN PROCESSES: PRACTICAL COURSE

Dilan Görür

Yahoo! Labs

April 2012 / MLSS

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 gp.
- A number of supporting structures and functions
  - 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.
  - 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 obund 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>	regression $y_i \in \mathbb{R}$	$\mathbb{P}_\rho(y_i\|f_i) =$	$\rho =$
Gauss	Gaussian	$\mathcal{N}(y_i\|f_i,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y_i-f_i)^2}{2\sigma^2}\right)$	$\{\ln\sigma\}$
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	$\frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})}\frac{1}{\sqrt{\nu\pi}\sigma}\left(1+\frac{(y_i-f_i)^2}{\nu\sigma^2}\right)^{-\frac{\nu+1}{2}}$	$\{\ln(\nu-1),\ln\sigma\}$
<NAME>	classification $y_i \in \{\pm 1\}$	$\mathbb{P}_\rho(y_i\|f_i) =$	$\rho =$
Erf	Error function	$\int_{-\infty}^{y_i f_i}\mathcal{N}(t)dt$	$\emptyset$
Logistic	Logistic function	$\frac{1}{1+\exp(-y_i f_i)}$	$\emptyset$

GPML Toolbox

Compatibility matrix

Likelihood \ Inference	Exact	EP	Laplace	Variational Bayes	regression
Gaussian	✓	✓	✓	✓	regression
Sech-squared		✓	✓	✓	regression
Laplacian		✓		✓	regression
Student's t			✓	✓	regression
Error function		✓	✓	✓	probit regression
Logistic function		✓	✓	✓	logit regression

Mean functions

Simple mean functions $m(\mathbf{x})$		
<NAME>	Meaning	$m(\mathbf{x}) =$
Zero	mean vanishes always	0
One	mean equals 1	1
Const	mean equals a constant	c
Linear	mean linearly depends on $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^D$	$\mathbf{a}^\top \mathbf{x}$

Composite mean functions $[\mu_1(\mathbf{x}), \mu_2(\mathbf{x}), ..] \mapsto m(\mathbf{x})$		
<NAME>	Meaning	$m(\mathbf{x}) =$
Scale	scale a mean	$\alpha\mu(\mathbf{x})$
Sum	add up mean functions	$\sum_j \mu_j(\mathbf{x})$
Prod	multiply mean functions	$\prod_j \mu_j(\mathbf{x})$
Pow	raise a mean to a power	$\mu(\mathbf{x})^d$
Mask	act on components $I \subseteq [1, 2, .., D]$ of $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^D$ only	$\mu(\mathbf{x}_I)$

Covariance functions

Simple

Simple covariance functions $k(x, x')$		
`<NAME>`	Meaning	$k(x, x') =$
`Zero`	mean vanishes always	0
`Noise`	additive measurement noise	$\sigma_f{}^2 \delta(x - x')$
`Const`	covariance equals a constant	$\sigma_f{}^2$
`LIN`	linear, $\mathcal{X} \subseteq \mathbb{R}^D$	$x^\top x'$
`LINard`	linear with diagonal weighting, $\mathcal{X} \subseteq \mathbb{R}^D$	$x^\top \Lambda^{-2} x'$
`LINone`	linear with bias, $\mathcal{X} \subseteq \mathbb{R}^D$	$(x^\top x' + 1)/\ell^2$
`Poly`	polynomial covariance, $\mathcal{X} \subseteq \mathbb{R}^D$	$\sigma_f^2 (x^\top x' + c)^d$
`SEard`	full squared exponential, $\mathcal{X} \subseteq \mathbb{R}^D$	$\sigma_f^2 \exp\left(-\frac{1}{2}(x - x')^\top \Lambda^{-2} (x - x')\right)$
`SEiso`	diagonal squared exponential, $\mathcal{X} \subseteq \mathbb{R}^D$	$\sigma_f^2 \exp\left(-\frac{1}{2\ell^2}(x - x')^\top (x - x')\right)$
`SEisoU`	squared exponential, $\mathcal{X} \subseteq \mathbb{R}^D$	$\exp(-\frac{1}{2\ell^2} x^\top x')$
`RQard`	rational quadratic, $\mathcal{X} \subseteq \mathbb{R}^D$	$\sigma_f^2 \left(1 + \frac{1}{2\alpha}(x - x')^\top \Lambda^{-2} (x - x')\right)^{-\alpha}$
`RQiso`	rational quadratic, $\mathcal{X} \subseteq \mathbb{R}^D$	$\sigma_f^2 \left(1 + \frac{1}{2\alpha\ell^2}(x - x')^\top (x - x')\right)^{-\alpha}$
`Materniso`	Matérn, $\mathcal{X} \subseteq \mathbb{R}^D$, $f_1(t) = 1$, $f_3(t) = 1 + t$, $f_5(t) = f_3(t) + \frac{t^2}{3}$	$\sigma_f^2 f_d(r_d) \exp(-r_d)$, $r_d = \sqrt{\frac{d}{\ell^2}(x - x')^\top (x - x')}$
`NNone`	neural net, $\mathcal{X} \subseteq \mathbb{R}^D$, $f(x) = 1 + x^\top \Lambda^{-2} x$	$\sigma_f^2 \sin^{-1}\left(\frac{x^\top \Lambda^{-2} x'}{\sqrt{f(x) f(x')}}\right)$
`Periodic`	periodic, $\mathcal{X} \subseteq \mathbb{R}$	$\sigma_f^2 \exp\left(-\frac{2}{\ell^2}\sin^2[\frac{\omega}{2\pi}(x - x')]\right)$
`PPiso`	compact support, piecewise polynomial $f_\nu(r)$, $\mathcal{X} \subseteq \mathbb{R}$,	$\sigma_f^2 \max(0, 1 - r) \cdot f_\nu(r)$, $r = \frac{\|x - x'\|}{\ell}$, $j = \lfloor \frac{D}{2} \rfloor + \nu + 1$

Covariance functions

Composite

Composite covariance functions $[\kappa_1(\mathbf{x},\mathbf{x}'), \kappa_2(\mathbf{x},\mathbf{x}'), ..] \mapsto k(\mathbf{x},\mathbf{x}')$		
<NAME>	Meaning	$k(\mathbf{x},\mathbf{x}') =$
Scale	scale a covariance	$\alpha\kappa(\mathbf{x},\mathbf{x}')$
Sum	add up covariance functions	$\sum_j \kappa_j(\mathbf{x},\mathbf{x}')$
Prod	multiply covariance functions	$\prod_j \kappa_j(\mathbf{x},\mathbf{x}')$
Mask	act on components $I \subseteq [1,2,..,D]$ of $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^D$ only	$\kappa(\mathbf{x}_I,\mathbf{x}'_I)$
ADD	additive, $\mathcal{X} \subseteq \mathbb{R}^D$, index degree set $\mathcal{D} = \{1,..,D\}$	$\sum_{d\in\mathcal{D}} \sigma^2_{f_d} \sum_{\lvert I\rvert=d} \prod_{i\in I} \kappa(x_i, x'_i; \psi_i)$

The gp function

Overview

[varargout] = gp(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

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

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

```
kss = feval(cov{:}, hyp.cov, xs(id,:), 'diag');     % self-variance
Ks = feval(cov{:}, hyp.cov, x(nz,:), xs(id,:));     % cross-covariances
ms = feval(mean{:}, hyp.mean, xs(id,:));
fmu(id) = ms + Ks'*full(alpha(nz));                 % predictive means
if Ltril     % L is triangular => use Cholesky parameters (alpha,sW,L)
  V = L'\(repmat(sW,1,length(id)).*Ks);
  fs2(id) = kss - sum(V.*V,1)';                     % predictive variances
else         % L is not triangular => use alternative parametrisation
  fs2(id) = kss + sum(Ks.*(L*Ks),1)';               % predictive variances
end
fs2(id) = max(fs2(id),0); % remove numerical noise i.e. negative variances
if nargin<9
  [lp(id) ymu(id) ys2(id)] = lik(hyp.lik, [], fmu(id), fs2(id)); else
  [lp(id) ymu(id) ys2(id)] = lik(hyp.lik, ys(id), fmu(id), fs2(id));
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,
   nlZ = (y-m)'*alpha/2 + sum(log(diag(L))) + n*log(2*pi*sn2)/2; % -log marg lik
   if nargout>2, dnlZ = hyp;      % derivatives
      Q = solve_chol(L,eye(n))/sn2 - alpha*alpha'; % precompute for convenience
      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
```

Marginal Likelihood vs Cross Validation

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

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

- 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.