Sparse Gaussian Process Toolbox

Lehel Csató

Max Planck Institute for Biological Cybernetics, Tübingen, Germany

Neural Computing Research Group, Aston University, Birmingham

MAX-PLANCK-GESELLSCHAFT

BIOLOGISCHE KYBERNETIK

Toolbox Overview

Uses:

- Gaussian process ($\mathcal{GP}$) latent model
- Sequential approximation to the posterior
- Sparsification of the resulting process
- *MATLAB* programming language
- *NETLAB* toolbox

Provides:

- GUI demos for teaching $\mathcal{GP}$s
- Variety of error or likelihood functions
- Bayesian hyperparameter selection

Freely available from:

- `http://www.tuebingen.mpg.de/~csatol`
- `http://www.ncrg.aston.ac.uk/Projects/SSGP`

Gaussian Process Inference

Gaussian process ($\mathcal{GP}$) models are *probabilistic* kernel methods. $\mathcal{GP}$s specify priors over a function space. Any finite sample from the random function has joint Gaussian distribution with covariance given by a kernel function. The *prior* is thus

$$p_0(\boldsymbol{f}) = \frac{1}{(2\pi)^{N/2}|\boldsymbol{K}_N|^{1/2}} \exp\left(-\frac{1}{2}\boldsymbol{f}^T\boldsymbol{K}_N^{-1}\boldsymbol{f}\right) \qquad (1)$$

where

$$\boldsymbol{f} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)]^T \quad \text{are samples from the function}$$

$$\boldsymbol{K}_N = \{K_0(\boldsymbol{x}_i, \boldsymbol{x}_j|\theta)\}_{i,j=1}^N \quad \text{is the sample covariance matrix}$$

where $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ are the inputs and θ is the set of parameters of K_0. For common $\mathcal{GP}$ models the likelihoods factorise:

$$P(\mathcal{D}) = \prod_{n=1}^N t(y_n|\boldsymbol{f}) \doteq \prod_{n=1}^N t_n(\boldsymbol{f})$$

The posterior *process* is

$$p_{\text{post}}(\boldsymbol{f}) = \frac{1}{Z} p_0(\boldsymbol{f}) P(\mathcal{D}|\boldsymbol{f})$$

For non-Gaussian likelihoods the posterior is not a $\mathcal{GP}$, thus it is analytically intractable.

There are **problems of:** *(solved by)*

- tractability for non-Gaussian likelihoods: *variational appr.*
- representation for large datasets: *sparse appr.*

Approximations to the Posterior

Representation of the posterior moments

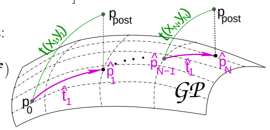
Using $\mathcal{GP}$ priors, the moments of the posterior process are:

$$\begin{aligned} \langle f_{\boldsymbol{x}} \rangle_{\text{post}} &= \langle f_{\boldsymbol{x}} \rangle_0 + \sum_{i=1}^N K_0(\boldsymbol{x}, \boldsymbol{x}_i)\alpha(i) \\ K(\boldsymbol{x}, \boldsymbol{x}')_{\text{post}} &= K_0(\boldsymbol{x}, \boldsymbol{x}') + \sum_{i,j=1}^N K_0(\boldsymbol{x}, \boldsymbol{x}_i) C(ij) K_0(\boldsymbol{x}_j, \boldsymbol{x}') \end{aligned} \qquad (2)$$

where $\alpha(i)$ and $C(ij)$ are parameters driven by the likelihood function.
The representation suggests an approximation: the posterior is approximated by the closest $\mathcal{GP}$ in a KL-sense. The process approximation is reduced to finding the parameters $\boldsymbol{\alpha} = [\alpha(1), \ldots, \alpha(N)]^T$ and $\boldsymbol{C} = \{C(i,j)\}_{i,j=1}^N$.

Approximation to the posterior process

The approximations are sequential, including a single case at each step. We use the *expectation-propagation (or TAP) algorithm* where the posterior is constructed from local Gaussian approximation $\hat{t}_i(\boldsymbol{f})$ to each factor in the likelihood [Minka 2000; Opper and Winther 2001].



We define the approximating process:

$$p_{\text{post}}(\boldsymbol{f}) \approx \hat{p}(\boldsymbol{f}) \propto p_0(\boldsymbol{f}) \prod_{n=1}^N \hat{t}_n(\boldsymbol{f})$$

and compute it using the algorithm:

- Initialise $\hat{t}_n(\boldsymbol{f}) = 1$ $i = 1, \ldots, N$.
- For each input n compute: $\quad p_0^{\setminus n}(\boldsymbol{f}) \propto \dfrac{\hat{p}(\boldsymbol{f})}{\hat{t}_n(\boldsymbol{f})}$
- Approximate the *local* posterior and substitute back $\hat{t}_n(\boldsymbol{f})$ based on:

$$\frac{1}{Z_n} p_0^{\setminus n}(\boldsymbol{f}) t_n(\boldsymbol{f}) \approx p_{\text{post}}^n(\boldsymbol{f}) = \frac{1}{\hat{Z}_n} p_0^{\setminus i}(\boldsymbol{f}) \hat{t}_n(\boldsymbol{f}) \quad \Rightarrow \quad t_n(\boldsymbol{f}) \approx \frac{Z_n}{\hat{Z}_n} \hat{t}_n(\boldsymbol{f}) \qquad (3)$$

At the equilibrium point we have an approximation to the *marginal likelihood* (or evidence) as:

$$Z = \int d\boldsymbol{f}\; p_0(\boldsymbol{f}) \prod_{n=1}^N t_n(\boldsymbol{f}) \approx \prod_{n=1}^N \frac{Z_n}{\hat{Z}_n} \int d\boldsymbol{f}\; p_0(\boldsymbol{f}) \hat{t}_n(\boldsymbol{f}) \qquad (4)$$

Sparsification

If $t_n(\boldsymbol{f})$ depends only on f_n, then $\boldsymbol{f}$ reduces to f_n. The random variable f_n can further be eliminated by

$$f_n \to \hat{f}_n = \pi_n \boldsymbol{f}_{\text{B}} \quad \text{, where B is a predefined set of inputs}$$

(B can be from the training/test data) The approximated posterior $\mathcal{GP}$ has the mean function and covariance kernel defined as:

$$\begin{aligned} \langle f_{\boldsymbol{x}} \rangle_{\text{post}} &= \langle f_{\boldsymbol{x}} \rangle_0 + \sum_{i \in \text{B}} K_0(\boldsymbol{x}, \boldsymbol{x}_i) \hat{\alpha}(i) \\ K(\boldsymbol{x}, \boldsymbol{x}')_{\text{post}} &= K_0(\boldsymbol{x}, \boldsymbol{x}') + \sum_{i,j \in \text{B}} K_0(\boldsymbol{x}, \boldsymbol{x}_i) \hat{C}(ij) K_0(\boldsymbol{x}_j, \boldsymbol{x}') \end{aligned}$$

Important: user control over the size of B. $\Rightarrow$ possible to use large datasets.

Matlab Implementation

Uses a *matlab structure* `net`. The initialisation of the structure with the default values for the fields is done using:

`net = ogp(` i_{dim}`,`o_{dim}`,covarfn,covpar);`

where i_{dim} is the dimension of inputs, o_{dim} is the dimension of outputs – one can define pairs of $\mathcal{GP}$s for more than a single latent variable in the likelihood. Associated to each input dimension there is an ARD [MacKay 1992] parameter:

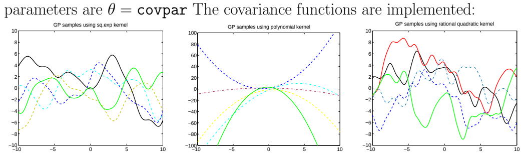
$\log(\nu)$`=net.inweights`

and the kernel functions depend on the weighted product:

$$\langle \boldsymbol{x}, \boldsymbol{x}' \rangle = \sum_i \gamma_i x_i x_i' \quad \text{which is used in the kernels below}$$

Kernel functions for the prior process

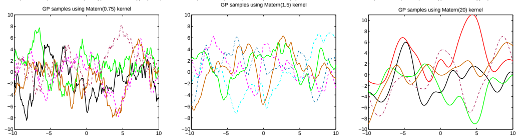
`covarfn,covpar` define the covariance function for the $\mathcal{GP}$. The kernel hyperparameters are θ = `covpar` The covariance functions are implemented:



`sqexp`: $K(\boldsymbol{x},\boldsymbol{x}') = \exp(-\|\boldsymbol{x}-\boldsymbol{x}'\|^2/2)$

`poly`: $K(\boldsymbol{x},\boldsymbol{x}') = (1+\langle \boldsymbol{x},\boldsymbol{x}' \rangle)^d$

`ratquad`: $K(\boldsymbol{x},\boldsymbol{x}') = (1+\|\boldsymbol{x}-\boldsymbol{x}'\|^2)^{-d}$



`user` - extensibility of the toolbox with user–defined covariance function **(example: Matèrn kernel)** [Stein 1999]:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \frac{A}{\Gamma(\nu) 2^{(\nu-1)}} \left(\sqrt{2\nu}d\right) K_\nu \left(\sqrt{2\nu}d\right)$$

where K_ν is the modified Bessel function of the second kind. Allows transition from rough covariances to squared exponentials by $\nu \to \infty$.

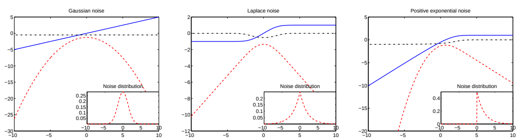
Likelihood models

The toolbox requires a function which returns the local update coefficients from eq. (3). It is specified by

`net = ogpinit(net,@likfn,likpar)`

For Gaussian noise model (`c_reg_gauss`) no approximation is needed.

Implemented likelihood models

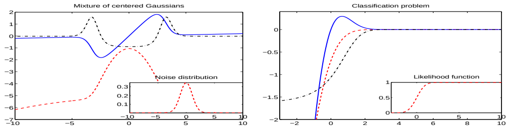


GAUSSIAN: $t_n(f_n) \propto e^{-\frac{\lambda(y_n - f_n)^2}{2}}$

LAPLACE: $t_n(f_n) \propto e^{-\lambda|y_n - f_n|}$

POS. EXP: Laplace + $y_n > f_n$

Obs: Nonstandard likelihood models are difficult to deal with using conventional kernel methods.



MIXTURES: $t_n(f_n) = \sum_i \pi_i N_i(y_n - f_n, \sigma_i^2)$

CLASSIFICATION: $t_n(f_n) \propto \int_{-\infty}^{y_n f_n} dz\; e^{-z^2/2}$

Inference and prediction

Iterating the following two-steps (EM algorithm):

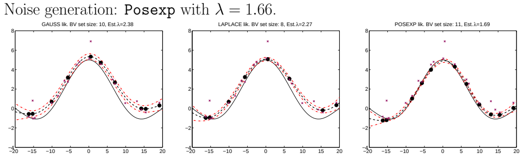
- Given the set of kernel-, and likelihood parameters, find $\mathcal{GP}_{opt}$ using the *EP* algorithm.
  `net = ogptrain(net,xTrain,yTrain,foptions);`
- Fixing the $\mathcal{GP}$, optimising the *evidence from eq.* (4) with respect to hyperparameters. For any kernel parameter θ

$$\frac{\partial \ln Z}{\partial \theta} = \text{tr}\left[\frac{\partial \ln Z}{\partial \boldsymbol{K}_B} \frac{\partial \boldsymbol{K}_B}{\partial \theta}\right]$$

`net = ogphyplearn(net,its)`

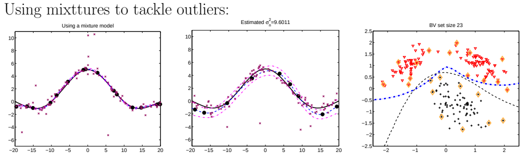
Examples

Noise generation: **Posexp** with $\lambda = 1.66$.



GAUSSIAN — LAPLACE — POS. EXP

Using mixttures to tackle outliers:



MIXTURE — GAUSSIAN LIKELIHOOD — CLASSIFICATION

Acknowledgements

Comments and discussions with Dan Cornford (NCRG), Ian Nabney (NCRG), Carl Rasmussen (MPI), codes from Ian Nabney (NCRG) and Anton Schwaighofer (TU Graz) are acknowledged.

References

MacKay, D. J. C. (1992). Bayesian interpolation. *Neural Computation* **4**, 415–447.

Minka, T. P. (2000). *Expectation Propagation for Approximate Bayesian Inference*. Ph. D. thesis, Dep. of El. Eng. & Comp. Sci.; MIT, vismod.www.media.mit.edu/~tpminka.

Opper, M. and O. Winther (2001). Adaptive and self-averaging TAP mean field theory for probabilistic modeling. *Physical Review E* **64**(056131), 1–14.

Stein, M. L. (Ed.) (1999). *Interpolation of Spatial Data: Some theory for kriging*. New York: Springer.