# Variational Pitman-Yor process infinite mixture of GPs

Laurent de Vito

April 14, 2015

This document presents a slightly different infinite mixtures of GPs than the one described by Sun and Xu [1] based partly on the PYP-GP regression model of Chatzis and Demiris [2]. Some useful tricks are borrowed from Titsias and Lazaro-Gredilla [6].

## 1 Theory

### 1.1 The model

Let us consider a regression problem with input variables $\mathbf{x} \in \mathbb{R}^D$ and output variables $y \in \mathbb{R}$. Let $f(\mathbf{x})$ be a latent function modeling $y$ as a function of the model input $\mathbf{x}$.

Let us consider a set of input/output regression pairs $\{\mathbf{x}_n, y_n\}_{n=1}^N$, comprising $N$ samples. Let us also introduce the set of variables $\{z_{nc}\}_{n,c=1}^{N,\infty}$ with $z_{nc} = 1$ if the function relating $\mathbf{x}_n$ to $y_n$ is considered to be expressed by $f_c(\mathbf{x}_n)$; $z_{nc} = 0$ otherwise. As a result:

$$p(y_n|\mathbf{x}_n, z_{nc} = 1) = \mathcal{N}(y_n|f_c(\mathbf{x}_n), \sigma_c^2) \tag{1}$$

We consider a Pitman-Yor mixture model for clustering the samples into components. Each component is modeled as a bag of samples drawn from a distribution over the input space. The mixture components are the distributions over the input space and the mixture proportions $\omega_c$ are represented with a stick-breaking process:

$$p(z_{nc} = 1|\boldsymbol{v}) = \omega_c(\boldsymbol{v}) \tag{2}$$

$$\omega_c(\boldsymbol{v}) = v_c \prod_{c'=1}^{C-1} (1 - v_{c'}) \ \in [0:1] \tag{3}$$

with

$$p(v_c) = \text{Beta}(1 - \delta, \alpha + \delta c) \tag{4}$$

This construction implies

$$\sum_{c=1}^{\infty} \omega_c(\boldsymbol{v}) = 1 \tag{5}$$

$\boldsymbol{f}_c$ is assigned a Gaussian process prior:

$$p(\boldsymbol{f}_c|\mathbf{X}) = \mathcal{N}(\boldsymbol{f}_c|\mathbf{a}_c, \mathbf{K}_c(\mathbf{X}, \mathbf{X})) \tag{6}$$

One can put a prior (e.g., inverse Gamma distribution) on $\sigma_c^2$. In this work, we treat them as fixed hyperparameters. Due to the effect of the innovation parameter $\alpha$ on the number of effective mixture components, a Gamma prior is imposed on it:

$$p(\alpha) = \mathcal{G}(\alpha|\eta_1, \eta_2) \tag{7}$$

The distribution over the input space for a mixture component is given by a Gaussian distribution with a full covariance matrix:

$$p(\mathbf{x}_n|z_{nc} = 1) = \mathcal{N}(\mathbf{x}|\mathbf{m}_c, \mathbf{R}_c^{-1}) \tag{8}$$

where $\mathbf{R_c}$ is the inverse covariance. Parameters $\mathbf{m}_c$ and $\mathbf{R_c}$ are given a Gaussian distribution and Wishart distribution prior respectively:

$$\mathbf{m}_c \sim \mathcal{N}(\mathbf{m}_c|\mathbf{g}_0, \mathbf{G}_0^{-1}) \ , \ \ \mathbf{R}_c \sim \mathcal{W}(\mathbf{R}_c|\mathbf{W}_0, \nu_0) \tag{9}$$

The sphere of influence of a given component is represented through a Gaussian distribution. This representation is more flexible that those based on axis-aligned cuts, but it might be limiting for some data sets. To further finely model the input space, one could adopt a mixture of Gaussian distributions as done by Yuan and Neubauer [26]. For even further flexibility in segmenting the input space, Duan et al.[25] proposed to model the components through Gaussian processes.

## 1.2 Variational algorithm

Learning and inference are carried out in a variational Bayesian approach. The variational distribution $q$ approximate the true posterior distribution over the infinite sets $\mathbf{Z}$, $\boldsymbol{v} = \{v_c\}_{c=1}^{\infty}$, $\{\mathbf{m}_c\}_{c=1}^{\infty}$, $\{\mathbf{R}_c\}_{c=1}^{\infty}$, and $\{\mathbf{f}_c\}_{c=1}^{\infty}$ and the innovation parameter $\alpha$. Bayesian inference is not tractable under this setting since we are dealing with an infinite number of parameters.

Chatzis and Demiris [2] employed the strategy elaborated by Blei and Jordan [7] on the basis of the stick-breaking representation: They fixed a value $C$ and let the variational posterior over the $v_c$ have the property $q(v_C = 1) = 1$. In other words, they set $\omega_c(\boldsymbol{v})$ equal to zero for $c > C$. As put in by Blei and Jordan [7], the truncation is not imposed on the model itself, but only on the variational distribution to allow for tractable inference. Hence, the truncation level $C$ is a variational parameter that can be freely set (depending on prior knowledge or computational budget for instance), and not part of the prior model specification. Remark that this restriction is not serious since there cannot be more clusters than samples.

We decompose the log marginal likelihood as

$$\log p(\mathbf{Y}) = \mathcal{L}(q(\mathbf{W})) + \mathrm{KL}(q(\mathbf{W})||p(\mathbf{W}|\mathbf{Y})) \tag{10}$$

where

$$\mathcal{L}(q(\mathbf{W})) = \int q(\mathbf{W}) \log \frac{p(\mathbf{Y}, \mathbf{W})}{q(\mathbf{W})} d\mathbf{W} \tag{11}$$

$$\mathrm{KL}(q(\mathbf{W})||p(\mathbf{W}|\mathbf{Y})) = \int q(\mathbf{W}) \log \frac{q(\mathbf{W})}{p(\mathbf{W}|\mathbf{Y})} d\mathbf{W} \tag{12}$$

and $\mathbf{W}$ denotes the set of all parameters of the model: $\mathbf{W} = \{\mathbf{Z}, \{\boldsymbol{f}_c\}_{c=1}^{C}, \alpha, \boldsymbol{v}, \{\boldsymbol{m}_c, \boldsymbol{R}_c\}_{c=1}^{C}\}$. All distributions are implicitly conditioned on the hyperparameters $\boldsymbol{\Xi}$ of the model: $\boldsymbol{\Xi} = \{\eta_1, \eta_2, \delta, \{\sigma_c^2\}_{c=1}^{C}, \{\boldsymbol{\theta}_c\}_{c=1}^{C}, \mathbf{m}_0, \mathbf{R}_0, \mathbf{W}_0, \nu_0\}$ The reader is referred to Chapter 10 of Bishop [4] for instance for an introduction to variational methods. $\mathbf{W}$ is the set of random variables that need to be marginalized out to compute the marginal likelihood. We approximate the true posterior distribution $p(\mathbf{W}|\mathbf{Y})$ by introducing a variational distribution $q(\mathbf{W})$ and minimizing the KL divergence (12).

For the variational framework to yield a computationally effective inference method, it is necessary to break some of the dependencies between latent variables that make the true posterior difficult to compute [7]. We consider fully-factorized variational distributions which break all of the dependencies:

$$q(\mathbf{W}) = \prod_{n=1}^{N} q(\mathbf{z}_n) \prod_{c=1}^{C-1} q(v_c) q(\alpha) \prod_{c=1}^{C-1} q(\mathbf{m}_c) \prod_{c=1}^{C-1} q(\mathbf{R}_c) \prod_{c}^{C} q(\boldsymbol{f}_c) \tag{13}$$

Remark that $q(\mathbf{z}_n)$ is a multivariate Bernoulli distribution. It implies that $\sum_{c=1}^{C} q(z_{nc} = 1) = 1$. Among all the distributions $q(\mathbf{W})$ having this factorization, the distributions $q^*$ for which the lower bound is largest are given by

$$\log q^*(\mathbf{z}_n) = \mathrm{E}_{\neq \mathbf{z}_n} [\log p(\mathbf{Y}, \mathbf{W})] + \text{const.} \tag{14}$$

$$\log q^*(v_c) = \mathrm{E}_{\neq v_c} [\log p(\mathbf{Y}, \mathbf{W})] + \text{const.} \tag{15}$$

$$\log q^*(\alpha) = \mathrm{E}_{\neq \alpha} [\log p(\mathbf{Y}, \mathbf{W})] + \text{const.} \tag{16}$$

$$\log q^*(\mathbf{m}_c) = \mathrm{E}_{\neq \mathbf{m}_c} [\log p(\mathbf{Y}, \mathbf{W})] + \text{const.} \tag{17}$$

$$\log q^*(\mathbf{R}_c) = \mathrm{E}_{\neq \mathbf{R}_c} [\log p(\mathbf{Y}, \mathbf{W})] + \text{const.} \tag{18}$$

$$\log q^*(\boldsymbol{f}_c) = \mathrm{E}_{\neq \boldsymbol{f}_c} [\log p(\mathbf{Y}, \mathbf{W})] + \text{const.} \tag{19}$$

The joint probability density function is:

$$p(\mathbf{Y}, \mathbf{W}|\mathbf{X}) = p(\mathbf{Y}, \mathbf{Z}, \{\boldsymbol{f}_c, \boldsymbol{m}_c, \boldsymbol{R}_c\}_{c=1}^{C}, \alpha, \boldsymbol{v}|\mathbf{X}) \tag{20}$$

$$= p(\mathbf{Y}|\mathbf{X}, \mathbf{Z}, \{\boldsymbol{f}_c, \boldsymbol{m}_c, \boldsymbol{R}_c\}_{c=1}^{C})\ p(\mathbf{Z}|\mathbf{X}, \boldsymbol{v}, \{\boldsymbol{m}_c, \boldsymbol{R}_c\}_{c=1}^{C})\ p(\boldsymbol{v}|\alpha)\ p(\alpha)\ p(\{\boldsymbol{f}_c\}_{c=1}^{C}|\mathbf{X})\ p(\{\boldsymbol{m}_c\}_{c=1}^{C})\ p(\{\boldsymbol{R}_c\}_{c=1}^{C}) \tag{21}$$

$$= \prod_{n=1}^{N} p(y_n|z_{nc} = 1, \boldsymbol{f}_c) \prod_{n=1}^{N} p(\mathbf{z}_n|\mathbf{x}_n, \boldsymbol{v}, \mathbf{m}_c, \mathbf{R}_c) \prod_{c=1}^{C-1} p(v_c|\alpha)\ p(\alpha) \prod_{c=1}^{C} p(\boldsymbol{f}_c|\mathbf{X}) \prod_{c=1}^{C} p(\boldsymbol{m}_c) \prod_{c=1}^{C} p(\boldsymbol{R}_c) \tag{22}$$

We have

$$p(\mathbf{z}_n) = \prod_{c=1}^{C} \omega_c^{z_{nc}} \tag{23}$$

and

$$p(\mathbf{x}_n|\mathbf{z}_n) = \prod_{c=1}^{C} \mathcal{N}(\mathbf{x}_n|\mathbf{m}_c, \mathbf{R}_c^{-1})^{z_{nc}} \tag{24}$$

Applying Bayes' theorem, we find the responsability of the mixture components for the input $\mathbf{x}_n$:

$$p(\mathbf{z}_n|\mathbf{x}_n) = \frac{p(\mathbf{x}_n|\mathbf{z}_n) p(\mathbf{z}_n)}{p(\mathbf{x}_n)} \tag{25}$$

Thus

$$
\begin{aligned}
p(\mathbf{Y}, \mathbf{W}|\mathbf{X}) = & \prod_{n,c=1}^{N,C} \left[\mathcal{N}(y_n|f_c(\mathbf{x}_n), \sigma_c^2)\right]^{z_{nc}} \\
& \prod_{n,c=1}^{N,C} \left[\omega_c \, \mathcal{N}(\mathbf{x}_n|\mathbf{m}_c, \mathbf{R}_c^{-1})\right]^{z_{nc}} \\
& \prod_{c=1}^{C-1} \mathcal{B}(v_c|1-\delta, \alpha+\delta c) \\
& \mathcal{G}(\alpha|\eta_1, \eta_2) \\
& \prod_{c=1}^{C} \mathcal{N}(\boldsymbol{f}_c|\mathbf{a}_c, \mathbf{K}_c) \\
& \prod_{c=1}^{C} \mathcal{N}(\boldsymbol{m}_c|\mathbf{g}_0, \mathbf{G}_0^{-1}) \\
& \prod_{c=1}^{C} \mathcal{W}(\boldsymbol{R}_c|\mathbf{W}_0, \nu_0)
\end{aligned} \tag{26}
$$

Taking the log, we have:

$$
\begin{aligned}
\log p(\mathbf{Y}, \mathbf{W}) = & \sum_{n,c=1}^{N,C} z_{nc} \log \mathcal{N}(y_n|f_c(\mathbf{x}_n), \sigma_c^2) \\
& + \sum_{n,c=1}^{N,C} z_{nc} \log(\omega_c) \\
& + \sum_{n,c=1}^{N,C} z_{nc} \log \mathcal{N}(\mathbf{x}_n|\mathbf{m}_c, \mathbf{R}_c^{-1}) \\
& + \sum_{c=1}^{C-1} \log \Gamma(1-\delta+\alpha+\delta c) - \log \Gamma(\alpha+\delta c) - \log \Gamma(1-\delta) - \delta \log(v_c) + (\alpha+\delta c-1) \log(1-v_c) \\
& + \eta_1 \log(\eta_2) - \log \Gamma(\eta_1) + (\eta_1 - 1) \log(\alpha) - \eta_2 \alpha \\
& + \sum_{c=1}^{C} \log \mathcal{N}(\boldsymbol{f}_c|\mathbf{a}_c, \mathbf{K}_c) \\
& + \sum_{c=1}^{C} \log \mathcal{N}(\boldsymbol{m}_c|\mathbf{g}_0, \mathbf{G}_0^{-1}) \\
& + \sum_{c=1}^{C} \log \mathcal{W}(\boldsymbol{R}_c|\mathbf{W}_0, \nu_0)
\end{aligned} \tag{27}
$$

Notice the abuse of notations: For the joint distribution, the sets $\boldsymbol{v} = \{v_c\}_{c=1}^{\infty}$, $\mathbf{Z} = \{z_{nc}\}_{n,c=1}^{N,\infty}$ $\{\mathbf{m}_c, \mathbf{R}_c\}_{c=1}^{\infty}$ and $\{\boldsymbol{f}^c\}_{d,c=1}^{D,\infty}$, are infinite and so we ought to replace the occurrence of $C$ by $\infty$ in (26) and (27). They are truncated so that the mixing proportions $\omega_c = 0$ for $c > C$ only for the variational distribution. Hence, for the derivation of the optimal variational distributions, the terms $v_c$, $z_{nc}$, $\mathbf{m}_c$, $\mathbf{R}_c$ and $\boldsymbol{f}_c$ for $c > C$ are treated as constant and thus can be ignored.

### 1.2.1 E-Step

#### 1.2.1.1 $\log q^*(\boldsymbol{f}_c)$

In the following, we omit the constant terms.

$$\log q^*(\boldsymbol{f}_c) = \mathrm{E}_{\neq \boldsymbol{f}_c}\left[\log p(\mathbf{Y}, \mathbf{W})\right] \tag{28}$$

$$= \sum_{n,c=1}^{N,C} \mathrm{E}_{\neq \boldsymbol{f}_c}\left[z_{nc} \log \mathcal{N}(y_n | f_c(\mathbf{x}_n), \sigma_c^2)\right] \quad : \mathrm{F}_1 \tag{29}$$

$$+ \sum_{c=1}^{C} \mathrm{E}_{\neq \boldsymbol{f}_c}\left[\log \mathcal{N}(\boldsymbol{f}_c | \mathbf{a}_c, \mathbf{K}_c)\right] \quad : \mathrm{F}_2 \tag{30}$$

$$\mathrm{F}_2 = -\frac{1}{2}(\mathbf{a}_c - \boldsymbol{f}_c)^T \mathbf{K}_c^{-1}(\mathbf{a}_c - \boldsymbol{f}_c) \tag{31}$$

$$\mathrm{F}_1 = -\frac{1}{2\sigma_c^2}(\mathbf{y} - \boldsymbol{f}_c)^T \mathrm{diag}(\mathrm{E}\left[z_{nc}\right]_{n=1}^N)(\mathbf{y} - \boldsymbol{f}_c) \tag{32}$$

$\mathrm{F}_1$ and $\mathrm{F}_2$ are quadratic in $\boldsymbol{f}_c$ and thus $q^*(\boldsymbol{f}_c)$ is Gaussian distributed. We readily see that

$$\mathbf{K}_{\mathrm{F}_1}^{-1} = \frac{1}{\sigma_c^2}\mathrm{diag}\left(\mathrm{E}\left[z_{nc}\right]_{n=1}^N\right) \tag{33}$$

$$\boldsymbol{\mu}_{\mathrm{F}_1} = \mathbf{y} \tag{34}$$

Using the formulae from Appendix A, we arrive at the result:

$$q^*(\boldsymbol{f}_c) = \mathcal{N}(\boldsymbol{f}_c | \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \tag{35}$$

with

$$\boldsymbol{\Sigma}_c = \left(\mathbf{K}_c^{-1} + \mathbf{B}_c^{-1}\right)^{-1} \tag{36}$$

$$\boldsymbol{\mu}_c = \boldsymbol{\Sigma}_c\left(\mathbf{B}_c^{-1}\mathbf{y} + \mathbf{K}_c^{-1}\mathbf{a}_c\right) \tag{37}$$

where we set

$$\mathbf{B}_c^{-1} = \frac{1}{\sigma_c^2}\mathrm{diag}\left(\mathrm{E}\left[z_{nc}\right]_{n=1}^N\right) \tag{38}$$

for convenience.

Notice that the update for $\boldsymbol{\Sigma}_c$ depends on the inverse $\mathbf{K}_c^{-1}$ which is not numerically stable as $\mathbf{K}_c$ in computer precision might not be invertible. This is however easily resolved by re-writing $\boldsymbol{\Sigma}_c$ as

$$\boldsymbol{\Sigma}_c = \mathbf{K}_c\left(\mathbf{K}_c + \mathbf{B}_c\right)^{-1}\mathbf{B}_c \tag{39}$$

or

$$\boldsymbol{\Sigma}_c = \mathbf{B}_c\left(\mathbf{K}_c + \mathbf{B}_c\right)^{-1}\mathbf{K}_c \tag{40}$$

Both are advantageous for computing $\boldsymbol{\mu}_c$ as

$$\boldsymbol{\mu}_c = \mathbf{K}_c\left(\mathbf{K}_c + \mathbf{B}_c\right)^{-1}\mathbf{y} + \mathbf{B}_c\left(\mathbf{K}_c + \mathbf{B}_c\right)^{-1}\mathbf{a}_c \tag{41}$$

It is also worth mentioning that we do not have to store all those big matrices $\boldsymbol{\Sigma}_c$: The update of all other variational distributions depends only on the diagonal elements as will be clear in the following.

#### 1.2.1.2 $\log\, q^*(\mathbf{z}_n)$

$$\begin{aligned}
\log\, q^*(\mathbf{z}_n) =& \mathrm{E}_{\neq \mathbf{z}_n}\left[\log\, p(\mathbf{Y}, \mathbf{W})\right] && (42)\\
=& \sum_{n,c=1}^{N,C} \mathrm{E}_{\neq \mathbf{z}_n}\left[z_{nc}\, \log\, \mathcal{N}(y_n|f_c(\mathbf{x}_n), \sigma_c^2)\right]\\
&+ \sum_{n,c=1}^{N,C} \mathrm{E}_{\neq \mathbf{z}_n}\left[z_{nc}\, \log(\omega_c)\right]\\
&+ \sum_{n,c=1}^{N,C} \mathrm{E}_{\neq \mathbf{z}_n}\left[z_{nc}\, \log\, \mathcal{N}(\mathbf{x}_n|\mathbf{m}_c, \mathbf{R}_c^{-1})\right] && (43)\\
=& \sum_{c=1}^{C} z_{nc} u_{nc} && (44)
\end{aligned}$$

where we introduced

$$u_{nc} = -\frac{1}{2}\log(2\pi\sigma_c^2) - \frac{1}{2\sigma_c^2}\mathrm{E}\left[(y_n - f_c(\mathbf{x}_n))^2\right] + \mathrm{E}\left[\log(\omega_c)\right] - \frac{D}{2}\log(2\pi) + \frac{1}{2}\mathrm{E}\left[\log|\mathbf{R}_c|\right] - \frac{1}{2}\mathrm{E}\left[(\mathbf{x}_n - \mathbf{m}_c)^T\mathbf{R}_c(\mathbf{x}_n - \mathbf{m}_c)\right]$$

Hence

$$\log(\gamma_{nc}) = \log\, q^*(z_{nc} = 1) = u_{nc} \tag{46}$$

From the constraint

$$\sum_{c=1}^{C} \gamma_{nc} = 1 \tag{47}$$

it follows that

$$\gamma_{nc} = \frac{\mathrm{e}^{u_{nc}}}{\sum_{c'=1}^{C} \mathrm{e}^{u_{nc'}}} \tag{48}$$

From (3), we deduce

$$\mathrm{E}\left[\log(\omega_c)\right] = \mathrm{E}\left[\log(v_c)\right] + \sum_{c'=1}^{c-1} \mathrm{E}\left[\log(1 - v_{c'})\right] \tag{49}$$

We will see later that $q^*(v_c)$ is a Beta distribution with shape parameters $\beta_{c,1}$ and $\beta_{c,2}$. As a result [8]

$$\begin{aligned}
\mathrm{E}\left[\log(v_c)\right] &= \psi(\beta_{c,1}) - \psi(\beta_{c,1} + \beta_{c,2}) && (50)\\
\mathrm{E}\left[\log(1 - v_c)\right] &= \psi(\beta_{c,2}) - \psi(\beta_{c,1} + \beta_{c,2}) && (51)
\end{aligned}$$

where $\psi(.)$ designates the Digamma function.
From (35), we have that

$$\mathrm{E}\left[(y_n - f_c(\mathbf{x}_n))^2\right] = (y_n - [\boldsymbol{\mu}_c]_n)^2 + [\boldsymbol{\Sigma}_c]_{nn} \tag{52}$$

where $[\boldsymbol{\mu}_c]_n$ designates the $n$-th component of the vector $\boldsymbol{\mu}_c$.

Further, $\mathbf{R}_c$ will be Wishart-distributed, $\mathbf{R}_c \sim \mathcal{W}(\mathbf{R}_c|\mathbf{W}_c, \nu_c)$, and so:

$$\mathrm{E}\left[\mathbf{R}_c\right] = \nu_c \mathbf{W}_c \tag{53}$$

$$\mathrm{E}\left[\log|\mathbf{R}_c|\right] = \sum_{d=1}^{D} \psi\left(\frac{1}{2}\left(\nu_c + 1 - d\right)\right) + D\,\log(2) + \log|\mathbf{W}_c| \tag{54}$$

$\mathbf{m}_c$ will be Gaussian-distributed, $\mathbf{m}_c \sim \mathcal{N}(\mathbf{m}_c|\mathbf{g}_c, \mathbf{G}_c)$, and so:

$$\mathrm{E}\left[(\mathbf{x}_n - \mathbf{m}_c)^T \mathbf{R}_c (\mathbf{x}_n - \mathbf{m}_c)\right] = (\mathbf{x}_n - \mathbf{g}_c)^T \nu_c \mathbf{W}_c (\mathbf{x}_n - \mathbf{g}_c) + \mathrm{tr}(\mathbf{G}_c \nu_c \mathbf{W}_c) \tag{55}$$

Notice that $\mathrm{E}[z_{nc}] = \gamma_{nc}$.

#### 1.2.1.3 $\log\, q^*(v_c)$

$$\log\, q^*(v_c) = \mathrm{E}_{\neq v_c}\left[\log\, p(\mathbf{Y}, \mathbf{W})\right] \tag{56}$$

$$= \sum_{n,c'=1}^{N,C} \mathrm{E}_{\neq v_c}\left[z_{nc'}\, \log(\omega_{c'})\right]$$

$$+ \sum_{c'=1}^{C-1} \mathrm{E}_{\neq v_c}\left[\log\, \Gamma(1 - \delta + \alpha + \delta c')\right] - \mathrm{E}_{\neq v_c}\left[\log\, \Gamma(\alpha + \delta c')\right] + \mathrm{E}_{\neq v_c}\left[-\delta\, \log(v_{c'}) + (\alpha + \delta c' - 1)\, \log(1 - v_{c'})\right] \tag{57}$$

$$= \sum_{n,c'=1}^{N,C} \mathrm{E}_{\neq v_c}\left[z_{nc'}\left(\log(v_{c'}) + \sum_{c''=1}^{c'-1} \log(1 - v_{c''})\right)\right]$$

$$-\delta\, \log(v_c) + (\mathrm{E}\left[\alpha\right] + \delta c - 1)\, \log(1 - v_c) \tag{58}$$

$$= \log(v_c) \sum_{n=1}^{N} \mathrm{E}\left[z_{nc}\right] + \sum_{n,c'=1}^{N,C} \mathrm{E}_{\neq v_c}\left[z_{nc'} \sum_{c''=1}^{c'-1} \log(1 - v_{c''})\right]$$

$$-\delta\, \log(v_c) + (\mathrm{E}\left[\alpha\right] + \delta c - 1)\, \log(1 - v_c) \tag{59}$$

$$= \log(v_c) \sum_{n=1}^{N} \mathrm{E}\left[z_{nc}\right] + \log(1 - v_c) \sum_{n,c'=c+1}^{N,C} \mathrm{E}\left[z_{nc'}\right]$$

$$-\delta\, \log(v_c) + (\mathrm{E}\left[\alpha\right] + \delta c - 1)\, \log(1 - v_c) \tag{60}$$

We readily see that $q^*(v_c)$ has a Beta distribution, $q^*(v_c) = \mathcal{B}(v_c|\beta_{c,1}, \beta_{c,2})$, with

$$\beta_{c,1} = 1 - \delta + \sum_{n=1}^{N} \mathrm{E}\left[z_{nc}\right] \tag{61}$$

$$\beta_{c,2} = \mathrm{E}\left[\alpha\right] + \delta c + \sum_{n,c'=c+1}^{N,C} \mathrm{E}\left[z_{nc'}\right] \tag{62}$$

We will see in the next section that $q^*(\alpha) = \mathcal{G}(\alpha|\hat{\eta}_1, \hat{\eta}_2)$, and thus $\mathrm{E}\left[\alpha\right] = \hat{\eta}_1/\hat{\eta}_2$.

#### 1.2.1.4 $\log\, q^*(\alpha)$

$$\begin{aligned}
\log\, q^*(\alpha) &= \mathrm{E}_{\neq\alpha}\left[\log\, p(\mathbf{Y},\mathbf{W})\right] && (63)\\
&= \sum_{c=1}^{C-1} \mathrm{E}_{\neq\alpha}\left[\log\,\Gamma(1-\delta+\alpha+\delta c) - \log\,\Gamma(\alpha+\delta c) - \delta\,\log(v_c) + (\alpha+\delta c-1)\,\log(1-v_c)\right]\\
&+ \mathrm{E}_{\neq\alpha}\left[(\eta_1-1)\,\log(\alpha) - \eta_2\alpha\right] && (64)\\
&= \sum_{c=1}^{C-1} \log\,\Gamma(1-\delta+\alpha+\delta c) - \log\,\Gamma(\alpha+\delta c) + \alpha\;\mathrm{E}\left[\log(1-v_c)\right]\\
&+ (\eta_1-1)\,\log(\alpha) - \eta_2\alpha && (65)
\end{aligned}$$

The Gamma function enjoys the property $\Gamma(1+z) = z\Gamma(z)$ for $\mathrm{Re}(z) > 0$, and so

$$\log\,\Gamma(1-\delta+\alpha+\delta c) = \log(\alpha+\delta(c-1)) + \log\,\Gamma(\alpha+\delta(c-1)) \tag{66}$$

Summing over $c$, we get for $C \geq 2$

$$\begin{aligned}
\sum_{c=1}^{C-1} \log\,\Gamma(1-\delta+\alpha+\delta c) - \log\,\Gamma(\alpha+\delta c) &= \sum_{c=1}^{C-1} \log(\alpha+\delta(c-1))\\
&+ \sum_{c=1}^{C-1} \log\,\Gamma(\alpha+\delta(c-1)) - \sum_{c=1}^{C-1} \log\,\Gamma(\alpha+\delta c) && (67)\\
&= \sum_{c=1}^{C-1} \log(\alpha+\delta(c-1)) + \log\,\Gamma(\alpha) - \log\,\Gamma(\alpha+\delta(C-1)) && (68)\\
&= \log(\alpha) + \sum_{c=1}^{C-2} \log(\alpha+\delta c)) + \log\,\Gamma(\alpha) - \log\,\Gamma(\alpha+\delta(C-1)) && (69)
\end{aligned}$$

For a Dirichlet process, we have $\delta = 0$ and so the summation equals $(C-1)\log(\alpha)$. As a result, $q^*(\alpha)$ follows nicely a Gamma distribution. But for a Pitman-Yor process, $\delta \neq 0$ and so we cannot proceed further analytically. We wish (65) had also the same form as the logarithmic of a Gamma density function. To solve this issue we will judiciously lower bound $\log[p(\mathbf{v}|\alpha)\; p(\alpha)]$. To see that this is a valid option, we recall that by maximizing the lower bound (11) w.r.t. the factor $q(\alpha)$ we get

$$\mathcal{L}(q(\alpha)) = \int q(\alpha)\,\log\, q^*(\alpha)\; d\alpha - \int q(\alpha)\,\log\, q(\alpha)\; d\alpha + \text{const.} \tag{70}$$

where

$$\log\, q^*(\alpha) = \int q(\boldsymbol{\Theta})\,\log[p(\mathbf{v}|\alpha)\; p(\alpha)]\; d\boldsymbol{\Theta} + \text{const.} \tag{71}$$

$\boldsymbol{\Theta}$ designates all all random variables excluding $\alpha$. We would recognize that (70) is the negative Kullback-Leibler divergence between $q(\alpha)$ and $q^*(\alpha)$ and thus that the optimal $q(\alpha)$ factor is given by $q^*(\alpha)$.

By lower bounding $\log[p(\mathbf{v}|\alpha)\; p(\alpha)]$, we lower bound $\log\, q^*(\alpha)$, and so we define $q^{**}(\alpha)$ such that $\log\, q^*(\alpha) \geq \log\, q^{**}(\alpha)$. This gives

$$\mathcal{L}(q(\alpha)) \geq \int q(\alpha)\,\log\, q^{**}(\alpha)\; d\alpha - \int q(\alpha)\,\log\, q(\alpha)\; d\alpha + \text{const.} \tag{72}$$

By setting the factor $q(\alpha)$ to the (suboptimal) solution $q^{**}(\alpha)$, the lower bound $\mathcal{L}(q(\alpha))$ is maximized. Even though we cannot maximize $\mathcal{L}(q(\alpha))$ directly, by maximizing the lower bound of $\mathcal{L}(q(\alpha))$ we can still reach the maximum value of $\mathcal{L}(q(\alpha))$ asymptotically [9].

How suboptimal it is, depends on how tight the lower bound is. Additional parameters $\boldsymbol{\zeta}$ can be introduced to make it indeed tight and those parameters will be in turn optimized as any other hyperparameters.

All we have to do is thus lower bound (69). There are at least to ways to do this.

- Since the Gamma function is log convex, it is lower bounded by its first-order Taylor series expansion:

$$\log \Gamma(\alpha) - \log \Gamma(\alpha + \delta(C-1)) \geq -\delta(C-1)\ \psi(\alpha + \delta(C-1)) \tag{73}$$

  The bound is tight if $\delta C \ll 1$. Qi [18] presented some Keckic-Vasic type inequalities (see Appendix C), notably

$$\psi(\alpha + \delta(C-1)) < \log(\delta(C-1)) + \frac{\alpha}{\delta(C-1)} \tag{74}$$

  This bound is not particularly tight. As a result:

$$\log \Gamma(\alpha) - \log \Gamma(\alpha + \delta(C-1)) \geq -\delta(C-1)\log(\delta(C-1)) - \alpha \tag{75}$$

- We use Wendell's double inequality, Qi [18], also listed in Appendix C:

$$\log \Gamma(\alpha) - \log \Gamma(\alpha + \delta(C-1)) \geq -\delta(C-1)\log(\alpha) \tag{76}$$

  that is valid for $\delta(C-1) < 1$. The bound is tight for $\delta C \ll 1$.

Since the log is monotone increasing, we have:

$$\log(\alpha + \delta c) \geq \log(\alpha) \tag{77}$$

This inequality is tight if $\delta C \ll 1$. By exploiting the concavity of the log, we could also derived that

$$\log(\alpha + \delta c) \geq \log(\delta c) + \frac{\alpha}{\delta c} \tag{78}$$

but this bound is not particularly tight.

Eventually, using the latter option:

$$\sum_{c=1}^{C-1} \log \Gamma(1 - \delta + \alpha + \delta c) - \log \Gamma(\alpha + \delta c) \geq (C-1)(1-\delta)\log(\alpha) \tag{79}$$

As a result, $q^*(\alpha)$ has a Gamma distribution, $q^*(\alpha) = \mathcal{G}(\alpha|\hat{\eta}_1, \hat{\eta}_2)$, with

$$\hat{\eta}_1 = \eta_1 + (C-1)(1-\delta) \tag{80}$$

$$\hat{\eta}_2 = \eta_2 - \sum_{c=1}^{C-1} \mathrm{E}\left[\log(1 - v_c)\right] \tag{81}$$

We recall that

$$\mathrm{E}[\alpha] = \frac{\hat{\eta}_1}{\hat{\eta}_2} \quad \text{and} \quad \mathrm{E}[\log(\alpha)] = \psi(\hat{\eta}_1) - \log(\hat{\eta}_2) \tag{82}$$

#### 1.2.1.5 $\log\, q^*(\mathbf{m}_c)$

$$\begin{aligned}
\log\, q^*(\mathbf{m}_c) &= \mathrm{E}_{\neq \mathbf{m}_c}\left[\log\, p(\mathbf{Y}, \mathbf{W})\right] && (83)\\
&= \sum_{n,c=1}^{N,C} \mathrm{E}_{\neq \mathbf{m}_c}\left[z_{nc}\, \log\, \mathcal{N}(\mathbf{x}_n|\mathbf{m}_c, \mathbf{R}_c^{-1})\right] \\
&+ \sum_{c=1}^{C} \mathrm{E}_{\neq \mathbf{m}_c}\left[\log\, \mathcal{N}(\mathbf{m}_c|\mathbf{g}_0, \mathbf{G}_0^{-1})\right] \\
&= -\frac{1}{2}\sum_{n=1}^{N} \mathrm{E}[z_{nc}](\mathbf{x}_n - \mathbf{m}_c)^T \mathrm{E}[\mathbf{R}_c](\mathbf{x}_n - \mathbf{m}_c) \quad : \mathrm{F}_1 && (84)\\
&- \frac{1}{2}(\mathbf{g}_0 - \mathbf{m}_c)^T \mathbf{G}_0 (\mathbf{g}_0 - \mathbf{m}_c) \quad : \mathrm{F}_2 && (85)\\
& && (86)
\end{aligned}$$

$\mathrm{F}_1$ and $\mathrm{F}_2$ are quadratic in $\mathbf{m}_c$ and thus $q^*(\mathbf{m}_c)$ is Gaussian distributed. We readily see that

$$\mathbf{K}_{\mathrm{F}_1}^{-1} = \left(\sum_{n=1}^{N} \gamma_{nc}\right) \mathrm{E}[\mathbf{R}_c] \tag{87}$$

$$\mathbf{K}_{\mathrm{F}_1}^{-1} \boldsymbol{\mu}_{\mathrm{F}_1} = \mathrm{E}[\mathbf{R}_c] \sum_{n=1}^{N} \gamma_{nc} \mathbf{x}_n \tag{88}$$

and thus

$$\boldsymbol{\mu}_{\mathrm{F}_1} = \frac{\sum_{n=1}^{N} \gamma_{nc} \mathbf{x}_n}{\sum_{n=1}^{N} \gamma_{nc}} \tag{89}$$

Using the formulae from Appendix A, we arrive at the result:

$$q^*(\mathbf{m}_c) = \mathcal{N}(\mathbf{m}_c|\mathbf{g}_c, \mathbf{G}_c) \tag{90}$$

with

$$\mathbf{G}_c = \left(\mathbf{G}_0 + \left(\sum_{n=1}^{N} \gamma_{nc}\right) \mathrm{E}[\mathbf{R}_c]\right)^{-1} \tag{91}$$

$$\mathbf{g}_c = \mathbf{G}_c \left(\mathbf{G}_0 \mathbf{g}_0 + \mathrm{E}[\mathbf{R}_c] \sum_{n=1}^{N} \gamma_{nc} \mathbf{x}_n\right) \tag{92}$$

#### 1.2.1.6 $\log\, q^*(\mathbf{R}_c)$

$$\log q^*(\mathbf{R}_c) = \mathrm{E}_{\neq \mathbf{R}_c}\left[\log p(\mathbf{Y}, \mathbf{W})\right] \tag{93}$$

$$\begin{aligned}
&= \sum_{n,c=1}^{N,C} \mathrm{E}_{\neq \mathbf{R}_c}\left[z_{nc} \log \mathcal{N}(\mathbf{x}_n | \mathbf{m}_c, \mathbf{R}_c^{-1})\right] \\
&+ \sum_{c=1}^{C} \mathrm{E}_{\neq \mathbf{R}_c}\left[\log \mathcal{W}(\mathbf{R}_c | \mathbf{W}_0, \nu_0)\right] \\
&= \frac{1}{2}\log|\mathbf{R}_c| \left(\sum_{n=1}^{N} \mathrm{E}[z_{nc}]\right) - \frac{1}{2}\sum_{n=1}^{N} \mathrm{E}[z_{nc}] \mathrm{E}\left[(\mathbf{x}_n - \mathbf{m}_c)^T \mathbf{R}_c (\mathbf{x}_n - \mathbf{m}_c)\right] \\
&+ \frac{\nu_0 - D - 1}{2}\log|\mathbf{R}_c| - \frac{1}{2}\mathrm{tr}(\mathbf{W}_0^{-1}\mathbf{R}_c)
\end{aligned} \tag{94}$$

Since

$$\mathrm{E}\left[(\mathbf{x}_n - \mathbf{m}_c)^T \mathbf{R}_c (\mathbf{x}_n - \mathbf{m}_c)\right] = (\mathbf{x}_n - \mathbf{g}_c)^T \mathbf{R}_c (\mathbf{x}_n - \mathbf{g}_c) + \mathrm{tr}(\mathbf{G}_c \mathbf{R}_c) = \mathrm{tr}((\mathbf{G}_c + (\mathbf{x}_n - \mathbf{g}_c)(\mathbf{x}_n - \mathbf{g}_c)^T)\mathbf{R}_c) \tag{95}$$

we readily see that

$$q^*(\mathbf{R}_c) = \mathcal{W}(\mathbf{R}_c | \mathbf{W}_c, \nu_c) \tag{96}$$

with

$$\mathbf{W}_c = \left(\mathbf{W}_0^{-1} + \left(\sum_{n=1}^{N} \gamma_{nc}\right)\mathbf{G}_c + \sum_{n=1}^{N} \gamma_{nc}(\mathbf{x}_n - \mathbf{g}_c)(\mathbf{x}_n - \mathbf{g}_c)^T\right)^{-1} \tag{97}$$

$$\nu_c = \nu_0 + \sum_{n=1}^{N} \gamma_{nc} \tag{98}$$

### 1.2.2 M-Step

In the M-step, the bound is maximized w.r.t. the hyperparameters $\{\eta_1, \eta_2, \delta, \sigma_c^2\}$ and the kernel hyperparameters $\{\boldsymbol{\theta}\}_{c=1}^{C}$. For the target variance $\sigma_c^2$ and $\eta_2$, a closed-form update can be derived. On the other hand, kernel hyperparameters and $\{\eta_1, \delta\}$ require non-linear gradient-based optimization.

Using point estimates for hyperparameters means that the posterior distribution of the hyperparameters is assumed to be sharply peaked. According to Rasmussen and Williams [23], this assumption works well for GP models in practice.

Following Yuan and Neubauer [26] and Sun and Xu [1], we do not infer some hyperparameters that can be set to a reasonable default: They are fixed. $\mathbf{g}_0$ and $\mathbf{G}_0$ are set to the mean $\boldsymbol{\mu}_\mathbf{x}$ and inverse covariance $\mathbf{R}_\mathbf{x}$ of the training data respectively. Parameter $\nu_0$, the number of degrees of freedom under a Wishart distribution, is set to the dimensionality $D$ of the inputs (least informative proper prior). $\mathbf{W}_0$ is set to $\mathbf{R}_\mathbf{x}/D$ such that the mean of $\mathbf{R}_c$ under the Wishart distribution is $\mathbf{R}_\mathbf{x}$.

The variational lower bound is given by

$$\mathcal{L} = \int q(\mathbf{W}) \log p(\mathbf{Y}, \mathbf{W}) \, d\mathbf{W} - \int q(\mathbf{W}) \log q(\mathbf{W}) \, d\mathbf{W} \tag{99}$$

or

$$\mathcal{L} = \mathrm{E}_q[\log p(\mathbf{Y}, \mathbf{W})] + \mathrm{H}_q \tag{100}$$

where $\mathrm{H}_q$ is the entropy of $q$. Using the decomposition (13), it can also be written as:

$$\begin{aligned}\mathcal{L}=&\mathrm{E}[\log\, p(\mathbf{Y}|\mathbf{X},\mathbf{Z},\{\boldsymbol{f}_c,\mathbf{m}_c,\mathbf{R}_c\}_{c=1}^C)]\\ &+\mathrm{E}[\log\, p(\mathbf{Z}|\mathbf{X},\boldsymbol{v},\{\mathbf{m}_c,\mathbf{R}_c\}_{c=1}^C)]+\mathrm{E}[\log\, p(\boldsymbol{v}|\alpha)]+\mathrm{E}[\log\, p(\alpha)]+\mathrm{E}[\log\, p(\{\boldsymbol{f}_c\}_{c=1}^C)]+\mathrm{E}[\log\, p(\{\boldsymbol{m}_c\}_{c=1}^C)]+\mathrm{E}[\log\, p(\{\boldsymbol{R}_c\}\\ &-\mathrm{E}[\log\, q(\mathbf{Z})]-\mathrm{E}[\log\, q(\boldsymbol{v})]-\mathrm{E}[\log\, q(\alpha)]-\mathrm{E}[\log\, q(\{\boldsymbol{f}_c\}_{c=1}^C)]-\mathrm{E}[\log\, q(\{\boldsymbol{m}_c\}_{c=1}^C)]-\mathrm{E}[\log\, q(\{\boldsymbol{R}_c\}_{c=1}^C)]\end{aligned}$$

We consider each term in turn.

#### 1.2.2.1 $\mathbf{E}[\log\, p(\mathbf{Y}|\mathbf{X},\mathbf{Z},\{\boldsymbol{f}_c,\mathbf{m}_c,\mathbf{R}_c\}_{c=1}^C)]$

$$\begin{aligned}\mathrm{E}[\log\, p(\mathbf{Y}|\mathbf{X},\mathbf{Z},\{\boldsymbol{f}_c,\mathbf{m}_c,\mathbf{R}_c\}_{c=1}^C)]&=\sum_{n,c=1}^{N,C}\int z_{nc}\,\log\,\mathcal{N}(y_n|f_c(\mathbf{x}_n),\sigma_c^2)\;q(z_{nc})\;q(f_c(\mathbf{x}_n))\;dz_{nc}\;df_c(\mathbf{x}_n) &(102)\\ &=\sum_{n,c=1}^{N,C}\mathrm{E}[z_{nc}]\int\left(-\frac{1}{2}\log(2\pi\sigma_c^2)-\frac{1}{2\sigma_c^2}(y_n-f_c(\mathbf{x}_n))^2\right)\;q(f_c(\mathbf{x}_n))\;df_c(\mathbf{x}_n) &(103)\\ &=-\frac{1}{2}\sum_{n,c=1}^{N,C}\log(2\pi\sigma_c^2)\;\mathrm{E}[z_{nc}]-\frac{1}{2}\sum_{n,c=1}^{N,C}\frac{\mathrm{E}[z_{nc}]}{\sigma_c^2}\left((y_n-[\boldsymbol{\mu}_c]_n)^2+[\boldsymbol{\Sigma}_c]_{nn}\right) &(104)\\ &=-\frac{1}{2}\sum_{n,c=1}^{N,C}\log(2\pi\sigma_c^2)\;\gamma_{nc}-\frac{1}{2}\sum_{n,c=1}^{N,C}\frac{\gamma_{nc}}{\sigma_c^2}((y_n-[\boldsymbol{\mu}_c]_n)^2+[\boldsymbol{\Sigma}_c]_{nn}) &(105)\end{aligned}$$

#### 1.2.2.2 $\mathbf{E}[\log\, p(\mathbf{Z}|\mathbf{X},\boldsymbol{v},\{\mathbf{m}_c,\mathbf{R}_c\}_{c=1}^C)]$

$$\begin{aligned}\mathrm{E}[\log\, p(\mathbf{Z}|\mathbf{X},\boldsymbol{v},\{\mathbf{m}_c,\mathbf{R}_c\}_{c=1}^C)]=&\sum_{n,c=1}^{N,C}\int z_{nc}\left(\log(v_c)+\sum_{c'=1}^{c-1}\log(1-v_{c'})\right)\;q(z_{nc})\;q(v_c)\ldots q(v_1)\;dz_{nc}\;dv_c\ldots\;dv_1\\ &+\sum_{n,c=1}^{N,C}\int z_{nc}\,\log\,\mathcal{N}(\mathbf{x}_n|\mathbf{m}_c,\mathbf{R}_c^{-1})\;q(z_{nc})\;q(\mathbf{m}_c)\;q(\mathbf{R}_c)\;dz_{nc}\;d\mathbf{m}_c\;d\mathbf{R}_c &(106)\end{aligned}$$

where the definition (3) of $\omega_c$ was used. Further:

$$\begin{aligned}
\mathrm{E}[\log p(\mathbf{Z}|\mathbf{X},\boldsymbol{v},\{\mathbf{m}_c,\mathbf{R}_c\}_{c=1}^C)] =& \sum_{c=1}^{C}\left(\mathrm{E}[\log(v_c)]\sum_{n=1}^{N}\mathrm{E}[z_{nc}]\right) + \sum_{n,c=1}^{N,C}\left(\mathrm{E}[z_{nc}]\sum_{c'=1}^{c-1}\mathrm{E}[\log(1-v_{c'})]\right) \\
&+ \sum_{n,c=1}^{N,C}\mathrm{E}[z_{nc}]\left(-\frac{D}{2}\log(2\pi)+\frac{1}{2}\mathrm{E}[\log|\mathbf{R}_c|]-\frac{1}{2}\mathrm{E}[(\mathbf{x}_n-\mathbf{m}_c)^T\mathbf{R}_c(\mathbf{x}_n-\mathbf{m}_c)]\right) \quad (107) \\
=& \sum_{c=1}^{C}\left(\mathrm{E}[\log(v_c)]\sum_{n=1}^{N}\gamma_{nc}\right) + \sum_{c=1}^{C}\left(\sum_{n=1}^{N}\gamma_{nc}\right)\left(\sum_{c'=1}^{c-1}\mathrm{E}[\log(1-v_{c'})]\right) \\
&-\frac{D}{2}\log(2\pi)\left(\sum_{n,c=1}^{N,C}\gamma_{nc}\right)+\frac{1}{2}\sum_{c=1}^{C}\left(\mathrm{E}[\log|\mathbf{R}_c|]\sum_{n=1}^{N}\gamma_{nc}\right) \\
&-\frac{1}{2}\sum_{n,c=1}^{N,C}\gamma_{nc}(\mathbf{x}_n-\mathbf{g}_c)^T\nu_c\mathbf{W}_c(\mathbf{x}_n-\mathbf{g}_c)-\frac{1}{2}\sum_{c=1}^{C}\left(\mathrm{tr}(\nu_c\mathbf{W}_c\mathbf{G}_c)\sum_{n=1}^{N}\gamma_{nc}\right) \quad (108)
\end{aligned}$$

#### 1.2.2.3 $\mathbf{E}[\log\ p(\boldsymbol{v}|\alpha)]$

$$\begin{aligned}
\mathrm{E}[\log\ p(\boldsymbol{v}|\alpha)] =& \sum_{c=1}^{C-1}\int (\log\ \Gamma(1-\delta+\alpha+\delta c)-\log\ \Gamma(\alpha+\delta c)-\log\ \Gamma(1-\delta) \\
&-\delta\ \log(v_c)+(\alpha+\delta c-1)\ \log(1-v_c)) \\
&q(\alpha)\ q(v_c)\ d\alpha\ dv_c \quad (109)
\end{aligned}$$

We have derived the lower bound (79) for the term $\sum_{c=1}^{C-1}\log\ \Gamma(1-\delta+\alpha+\delta c)-\log\ \Gamma(\alpha+\delta c)$, and so we have:

$$\begin{aligned}
\mathrm{E}[\log\ p(\boldsymbol{v}|\alpha)] \geq& (C-1)(1-\delta)\mathrm{E}[\log(\alpha)]-(C-1)\log\ \Gamma(1-\delta) \\
&+\sum_{c=1}^{C-1}-\delta\ \mathrm{E}[\log(v_c)]+(\mathrm{E}[\alpha]+\delta c-1)\ \mathrm{E}[\log(1-v_c)] \quad (110)
\end{aligned}$$

#### 1.2.2.4 $\mathbf{E}[\log\ p(\alpha)]$

$$\begin{aligned}
\mathrm{E}[\log\ p(\alpha)] =& \int(\eta_1\log(\eta_2)-\log\ \Gamma(\eta_1)+(\eta_1-1)\ \log(\alpha)-\eta_2\alpha)\ \ q(\alpha)\ d\alpha \quad (111) \\
=& \eta_1\log(\eta_2)-\log\ \Gamma(\eta_1)+(\eta_1-1)\ \mathrm{E}[\log(\alpha)]-\eta_2\ \mathrm{E}[\alpha] \quad (112)
\end{aligned}$$

#### 1.2.2.5 $\mathbf{E}[\log\ p(\{\boldsymbol{f}_c\}_{c=1}^C)]$

$$\mathrm{E}[\log\, p(\{\boldsymbol{f}_c\}_{c=1}^C)] = \sum_{c=1}^{C} \int \log\, \mathcal{N}(\boldsymbol{f}_c|\mathbf{a}_c, \mathbf{K}_c)\; q(\boldsymbol{f}_c)\; d\boldsymbol{f}_c \tag{113}$$

$$= \sum_{c=1}^{C} \int \left( -\frac{N}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{K}_c| - \frac{1}{2}(\boldsymbol{f}_c - \mathbf{a}_c)^T \mathbf{K}_c^{-1}(\boldsymbol{f}_c - \mathbf{a}_c) \right)\; q(\boldsymbol{f}_c)\; d\boldsymbol{f}_c \tag{114}$$

$$= -\frac{NC}{2}\log(2\pi) - \frac{1}{2}\sum_{c}^{C}\log|\mathbf{K}_c| - \frac{1}{2}\sum_{c=1}^{C}\left[(\boldsymbol{\mu}_c - \mathbf{a}_c)^T \mathbf{K}_c^{-1}(\boldsymbol{\mu}_c - \mathbf{a}_c) + \mathrm{tr}(\boldsymbol{\Sigma}_c \mathbf{K}_c^{-1})\right] \tag{115}$$

$$= -\frac{NC}{2}\log(2\pi) - \frac{1}{2}\sum_{c}^{C}\log|\mathbf{K}_c| - \frac{1}{2}\sum_{c=1}^{C}\mathrm{tr}(\mathbf{K}_c^{-1}(\boldsymbol{\Sigma}_c + (\boldsymbol{\mu}_c - \mathbf{a}_c)(\boldsymbol{\mu}_c - \mathbf{a}_c)^T)) \tag{116}$$

#### 1.2.2.6 E[log $p(\{\boldsymbol{m}_c\}_{c=1}^C)$]

$$\mathrm{E}[\log\, p(\{\boldsymbol{m}_c\}_{c=1}^C)] = \sum_{c=1}^{C} \int \log\, \mathcal{N}(\boldsymbol{m}_c|\mathbf{g}_0, \mathbf{G}_0^{-1})\; q(\boldsymbol{m}_c)\; d\boldsymbol{m}_c \tag{117}$$

$$= \sum_{c=1}^{C} \int \left( -\frac{D}{2}\log(2\pi) + \frac{1}{2}\log|\mathbf{G}_0| - \frac{1}{2}(\boldsymbol{m}_c - \mathbf{g}_0)^T \mathbf{G}_0(\boldsymbol{m}_c - \mathbf{g}_0) \right)\; q(\boldsymbol{m}_c)\; d\boldsymbol{m}_c \tag{118}$$

$$= -\frac{DC}{2}\log(2\pi) + \frac{C}{2}\log|\mathbf{G}_0| - \frac{1}{2}\mathrm{tr}\left( \mathbf{G}_0 \sum_{c=1}^{C} \left( \mathbf{G}_c + (\mathbf{g}_c - \mathbf{g_0})(\mathbf{g}_c - \mathbf{g_0})^T \right) \right) \tag{119}$$

#### 1.2.2.7 E[log $p(\{\boldsymbol{R}_c\}_{c=1}^C)$]

$$\mathrm{E}[\log\, p(\{\boldsymbol{R}_c\}_{c=1}^C)] = \sum_{c=1}^{C} \int \log\, \mathcal{W}(\boldsymbol{R}_c|\mathbf{W}_0, \nu_0)\; q(\boldsymbol{R}_c)\; d\boldsymbol{R}_c$$

$$= \sum_{c=1}^{C} \int \left( -\frac{\nu_0 D}{2}\log(2) - \frac{\nu_0}{2}\log|\mathbf{W}_0| - \log\,\Gamma_D\left(\frac{\nu_0}{2}\right) + \frac{\nu_0 - D - 1}{2}\log|\mathbf{R}_c| - \frac{1}{2}\mathrm{tr}(\mathbf{W}_0^{-1}\mathbf{R}_c) \right)\; q(\boldsymbol{R}_c)\; d\boldsymbol{R}$$

$$= -\frac{\nu_0 CD}{2}\log(2) - \frac{\nu_0 C}{2}\log|\mathbf{W}_0| - C\,\log\,\Gamma_D\left(\frac{\nu_0}{2}\right) + \frac{\nu_0 - D - 1}{2}\sum_{c=1}^{C}\mathrm{E}[\log|\mathbf{R}_c|] - \frac{1}{2}\mathrm{tr}\left( \mathbf{W}_0^{-1}\sum_{c=1}^{C}\nu_c \mathbf{W}_c \right.$$

where $\Gamma_D$ designates the multivariate Gamma function:

$$\log\,\Gamma_d(n) = \frac{d(d-1)}{4}\log(\pi) + \sum_{j=1}^{d}\log\,\Gamma\left(n + \frac{1-j}{2}\right) \tag{123}$$

#### 1.2.2.8 E[log $q(\mathbf{Z})$]

$$\mathrm{E}[\log\, q(\mathbf{Z})]=\mathrm{E}\left[\log \prod_{n=1}^{N} q(\mathbf{z}_n)\right] = \mathrm{E}\left[\log \prod_{n,c=1}^{N,C} \gamma_{nc}^{z_{nc}}\right] = \sum_{n,c=1}^{N,C} \mathrm{E}[z_{nc}]\, \log(\gamma_{nc}) = \sum_{n,c=1}^{N,C} \gamma_{nc}\, \log(\gamma_{nc}) \tag{124}$$

#### 1.2.2.9 $\mathbf{E}[\log\, q(\boldsymbol{v})]$

$$\mathrm{E}[\log\, q(\boldsymbol{v})]=\mathrm{E}\left[\log \prod_{c=1}^{C-1} q(v_c)\right] = \sum_{c=1}^{C-1} \mathrm{E}[\log\, q(v_c)] \tag{125}$$

We know that $q(v_c)$ is a Beta distribution:

$$q(v_c) = \frac{\Gamma(\beta_{c,1}+\beta_{c,2})}{\Gamma(\beta_{c,1})\Gamma(\beta_{c,2})} v_c^{\beta_{c,1}-1}(1-v_c)^{\beta_{c,2}-1} \tag{126}$$

Hence

$$\mathrm{E}[\log\, q(\boldsymbol{v})]=\sum_{c=1}^{C-1} \log\, \frac{\Gamma(\beta_{c,1}+\beta_{c,2})}{\Gamma(\beta_{c,1})\Gamma(\beta_{c,2})} + \sum_{c=1}^{C-1} (\beta_{c,1}-1)\mathrm{E}[\log(v_c)] + (\beta_{c,2}-1)\mathrm{E}[\log(1-v_c)] \tag{127}$$

#### 1.2.2.10 $\mathbf{E}[\log\, q(\alpha)]$

We have seen that $q^*(\alpha) = \mathcal{G}(\alpha|\hat{\eta}_1, \hat{\eta}_2)$, thus

$$\mathrm{E}[\log\, q(\alpha)]=\hat{\eta}_1\, \log(\hat{\eta}_2) - \log\, \Gamma(\hat{\eta}_1) + (\hat{\eta}_1 - 1)\, \mathrm{E}[\log(\alpha)] - \hat{\eta}_2 \mathrm{E}[\alpha] \tag{128}$$

#### 1.2.2.11 $\mathbf{E}[\log\, q(\{\boldsymbol{f}_c\}_{c=1}^C)]$

$$\mathrm{E}[\log\, q(\{\boldsymbol{f}_c\}_{c=1}^C)]=\mathrm{E}\left[\log \prod_{c=1}^{C} q(\boldsymbol{f}_c)\right] = \sum_{c=1}^{C} \mathrm{E}[\log\, q(\boldsymbol{f}_c)] \tag{129}$$

Since $q^*(\boldsymbol{f}_c) = \mathcal{N}(\boldsymbol{f}_c|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$, we have:

$$\mathrm{E}[\log\, q(\{\boldsymbol{f}_c\}_{c=1}^C)]=\sum_{c=1}^{C} \mathrm{E}\left[-\frac{N}{2}\log(2\pi) - \frac{1}{2}\log\, |\boldsymbol{\Sigma}_c| - \frac{1}{2}(\boldsymbol{f}_c - \boldsymbol{\mu}_c)^T \boldsymbol{\Sigma}_c^{-1}(\boldsymbol{f}_c - \boldsymbol{\mu}_c)\right] \tag{130}$$

$$=-\frac{NC}{2}\log(2\pi) - \frac{1}{2}\sum_{c=1}^{C} \log\, |\boldsymbol{\Sigma}_c| - \frac{CN}{2} \tag{131}$$

#### 1.2.2.12 $\mathbf{E}[\log\ q(\{\boldsymbol{m}_c\}_{c=1}^C)]$

$$\mathrm{E}[\log\ q(\{\boldsymbol{m}_c\}_{c=1}^C)]=\mathrm{E}\left[\log\prod_{c=1}^C q(\boldsymbol{m}_c)\right]=\sum_{c=1}^C \mathrm{E}[\log\ q(\boldsymbol{m}_c)] \tag{132}$$

Since $q^*(\boldsymbol{m}_c)=\mathcal{N}(\boldsymbol{m}_c|\mathbf{g}_c,\mathbf{G}_c)$, we have:

$$\mathrm{E}[\log\ q(\{\boldsymbol{m}_c\}_{c=1}^C)]=\sum_{c=1}^C \mathrm{E}\left[-\frac{D}{2}\log(2\pi)-\frac{1}{2}\log\ |\mathbf{G}_c|-\frac{1}{2}(\boldsymbol{m}_c-\mathbf{g}_c)^T\mathbf{G}_c^{-1}(\boldsymbol{m}_c-\mathbf{g}_c)\right] \tag{133}$$

$$=-\frac{CD}{2}\log(2\pi)-\frac{1}{2}\sum_{c=1}^C\log\ |\mathbf{G}_c|-\frac{CD}{2} \tag{134}$$

#### 1.2.2.13 $\mathbf{E}[\log\ q(\{\boldsymbol{R}_c\}_{c=1}^C)]$

$$\mathrm{E}[\log\ q(\{\boldsymbol{R}_c\}_{c=1}^C)]=\mathrm{E}\left[\log\prod_{c=1}^C q(\boldsymbol{R}_c)\right]=\sum_{c=1}^C \mathrm{E}[\log\ q(\boldsymbol{R}_c)] \tag{135}$$

Since $q^*(\boldsymbol{R}_c)=\mathcal{W}(\boldsymbol{R}_c|\mathbf{W}_c,\nu_c)$, we have:

$$\mathrm{E}[\log\ q(\{\boldsymbol{R}_c\}_{c=1}^C)]=\sum_{c=1}^C \mathrm{E}\left[-\frac{\nu_c D}{2}\log(2)-\frac{\nu_c}{2}\log\ |\mathbf{W}_c|-\log\ \Gamma_D\left(\frac{\nu_c}{2}\right)+\frac{\nu_c-D-1}{2}\log\ |\mathbf{R}_c|-\frac{1}{2}\mathrm{tr}(\mathbf{W}_c^{-1}\mathbf{R}_c)\right] \tag{136}$$

$$=-\frac{D}{2}\log(2)\sum_{c=1}^C\nu_c-\frac{1}{2}\sum_{c=1}^C\nu_c\log\ |\mathbf{W}_c|-\sum_{c=1}^C\log\ \Gamma_D\left(\frac{\nu_c}{2}\right)+\sum_{c=1}^C\frac{\nu_c-D-1}{2}\mathrm{E}[\log\ |\mathbf{R}_c|]-\frac{D}{2}\sum_{c=1}^C\nu_c \tag{137}$$

To recap:

$$
\begin{aligned}
\mathcal{L} \geq & -\frac{1}{2}\sum_{n,c=1}^{N,C}\log(2\pi\sigma_c^2)\ \gamma_{nc} - \frac{1}{2}\sum_{n,c=1}^{N,C}\frac{\gamma_{nc}}{\sigma_c^2}((y_n - [\boldsymbol{\mu}_c]_n)^2 + [\boldsymbol{\Sigma}_c]_{nn}) && \%\mathcal{F}_1 \\
& + \sum_{c=1}^{C}\left(\mathrm{E}[\log(v_c)]\sum_{n=1}^{N}\gamma_{nc}\right) + \sum_{c=1}^{C}\left(\sum_{n=1}^{N}\gamma_{nc}\right)\left(\sum_{c'=1}^{c-1}\mathrm{E}[\log(1-v_{c'})]\right) \\
& - \frac{D}{2}\log(2\pi)\sum_{n,c=1}^{N,C}\gamma_{nc} + \frac{1}{2}\sum_{c=1}^{C}\left(\mathrm{E}[\log|\mathbf{R}_c|]\sum_{n=1}^{N}\gamma_{nc}\right) \\
& - \frac{1}{2}\sum_{n,c=1}^{N,C}\gamma_{nc}(\mathbf{x}_n - \mathbf{g}_c)^T\nu_c\mathbf{W}_c(\mathbf{x}_n - \mathbf{g}_c) - \frac{1}{2}\sum_{c=1}^{C}\left(\mathrm{tr}(\nu_c\mathbf{W}_c\mathbf{G}_c)\sum_{n=1}^{N}\gamma_{nc}\right) && \%\mathcal{F}_2 \\
& - (C-1)\ \log\ \Gamma(1-\delta) \\
& + (C-1)(1-\delta)\mathrm{E}[\log(\alpha)] \\
& + \sum_{c=1}^{C-1} -\delta\ \mathrm{E}[\log(v_c)] + (\mathrm{E}[\alpha] + \delta c - 1)\ \mathrm{E}[\log(1-v_c)] && \%\mathcal{F}_3 \\
& + \eta_1\log(\eta_2) - \log\ \Gamma(\eta_1) + (\eta_1 - 1)\ \mathrm{E}[\log(\alpha)] - \eta_2\ \mathrm{E}[\alpha] && \%\mathcal{F}_4 \\
& - \frac{NC}{2}\log(2\pi) - \frac{1}{2}\sum_{c}^{C}\log|\mathbf{K}_c| - \frac{1}{2}\sum_{c=1}^{C}\mathrm{tr}(\mathbf{K}_c^{-1}(\boldsymbol{\Sigma}_c + (\boldsymbol{\mu}_c - \boldsymbol{a}_c)(\boldsymbol{\mu}_c - \boldsymbol{a}_c)^T)) && \%\mathcal{F}_5 \\
& - \frac{DC}{2}\log(2\pi) + \frac{C}{2}\log|\mathbf{G}_0| - \frac{1}{2}\mathrm{tr}\left(\mathbf{G}_0\sum_{c=1}^{C}\left(\mathbf{G}_c + (\mathbf{g}_c - \mathbf{g_0})(\mathbf{g}_c - \mathbf{g_0})^T\right)\right) && \%\mathcal{F}_6 \\
& - \frac{\nu_0 CD}{2}\log(2) - \frac{\nu_0 C}{2}\log|\mathbf{W}_0| - C\ \log\ \Gamma_D\left(\frac{\nu_0}{2}\right) \\
& + \frac{\nu_0 - D - 1}{2}\sum_{c=1}^{C}\mathrm{E}[\log|\mathbf{R}_c|] - \frac{1}{2}\mathrm{tr}\left(\mathbf{W}_0^{-1}\sum_{c=1}^{C}\nu_c\mathbf{W}_c\right) && \%\mathcal{F}_7 \\
& - \sum_{n,c=1}^{N,C}\gamma_{nc}\ \log(\gamma_{nc}) && \%\mathcal{E}_2 \\
& - \sum_{c=1}^{C-1}\log\frac{\Gamma(\beta_{c,1}+\beta_{c,2})}{\Gamma(\beta_{c,1})\Gamma(\beta_{c,2})} - \sum_{c=1}^{C-1}(\beta_{c,1}-1)\ \mathrm{E}[\log(v_c)] + (\beta_{c,2}-1)\ \mathrm{E}[\log(1-v_c)] && \%\mathcal{E}_3 \\
& - \hat{\eta}_1\ \log(\hat{\eta}_2) + \log\ \Gamma(\hat{\eta}_1) - (\hat{\eta}_1 - 1)\ \mathrm{E}[\log(\alpha)] + \hat{\eta}_2\mathrm{E}[\alpha] && \%\mathcal{E}_4 \\
& + \frac{NC}{2}\log(2\pi) + \frac{1}{2}\sum_{c=1}^{C}\log\ |\boldsymbol{\Sigma}_c| + \frac{CN}{2} && \%\mathcal{E}_5 \\
& + \frac{CD}{2}\log(2\pi) + \frac{1}{2}\sum_{c=1}^{C}\log\ |\mathbf{G}_c| + \frac{CD}{2} && \%\mathcal{E}_6 \\
& + \frac{D}{2}\log(2)\sum_{c=1}^{C}\nu_c + \frac{1}{2}\sum_{c=1}^{C}\nu_c\log\ |\mathbf{W}_c| + \sum_{c=1}^{C}\log\ \Gamma_D\left(\frac{\nu_c}{2}\right) \\
& - \sum_{c=1}^{C}\frac{\nu_c - D - 1}{2}\mathrm{E}[\log\ |\mathbf{R}_c|] + \frac{D}{2}\sum_{c=1}^{C}\nu_c && \%\mathcal{E}_7
\end{aligned}
\tag{138}
$$

Notice that the hyperparameters $\{\eta_1, \eta_2, \delta, \sigma_c^2\}$ are also hidden behind $\mathrm{E}[\alpha]$, $\mathrm{E}[\log(\alpha)]$, $\mathrm{E}[\log(v_c)]$, $\mathrm{E}[\log(1-v_c)]$, $\gamma_{nc}$, $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$. We ignore this dependency when we maximize the lower bound w.r.t $\{\eta_1, \eta_2, \delta, \sigma_c^2\}$ otherwise the maximization is not tractable.

The terms involving $\sigma_c^2$ are:

$$\mathcal{L}_{\sigma_c^2} = -\frac{1}{2}\log(2\pi\sigma_c^2)\sum_{n=1}^{N}\gamma_{nc} - \frac{1}{2\sigma_c^2}\sum_{n=1}^{N}\gamma_{nc}((y_n - [\boldsymbol{\mu}_c]_n)^2 + [\boldsymbol{\Sigma}_c]_{nn}) \tag{139}$$

Taking the derivative of $\mathcal{L}_{\sigma_c^2}$ against $\sigma_c^2$ and setting it to 0 gives the following update:

$$\sigma_c^2 = \frac{\sum_{n=1}^{N}\gamma_{nc}((y_n - [\boldsymbol{\mu}_c]_n)^2 + [\boldsymbol{\Sigma}_c]_{nn})}{\sum_{n=1}^{N}\gamma_{nc}} \tag{140}$$

The terms involving $\delta$ are:

$$\mathcal{L}_\delta = -(C-1)\ \log\Gamma(1-\delta) + (C-1)(1-\delta)\mathrm{E}[\log(\alpha)] + \delta\sum_{c=1}^{C-1} -\mathrm{E}[\log(v_c)] + c\ \mathrm{E}[\log(1-v_c)] \tag{141}$$

The derivative of $\mathcal{L}_\delta$ w.r.t $\delta$ is thus given by

$$\frac{\partial\mathcal{L}_\delta}{\partial\delta} = (C-1)\ \psi(1-\delta) - (C-1)\mathrm{E}[\log(\alpha)] + \sum_{c=1}^{C-1} -\mathrm{E}[\log(v_c)] + c\ \mathrm{E}[\log(1-v_c)] \tag{142}$$

The terms involving $\eta_1$ are:

$$\mathcal{L}_{\eta_1} = \eta_1\log(\eta_2) - \log\Gamma(\eta_1) + \eta_1\ \mathrm{E}[\log(\alpha)] \tag{143}$$

The derivative of $\mathcal{L}_{\eta_1}$ w.r.t $\eta_1$ is thus given by

$$\frac{\partial\mathcal{L}_{\eta_1}}{\partial\eta_1} = \log(\eta_2) - \psi(\eta_1) + \mathrm{E}[\log(\alpha)] \tag{144}$$

The terms involving $\eta_2$ are:

$$\mathcal{L}_{\eta_2} = \eta_1\log(\eta_2) - \eta_2\ \mathrm{E}[\alpha] \tag{145}$$

The derivative of $\mathcal{L}_{\eta_2}$ w.r.t $\eta_2$ is thus given by

$$\frac{\partial\mathcal{L}_{\eta_2}}{\partial\eta_2} = \frac{\eta_1}{\eta_2} - \mathrm{E}[\alpha] \tag{146}$$

Setting it to zero, we obtain the update

$$\eta_2 = \frac{\eta_1\hat{\eta}_2}{\hat{\eta}_1} \tag{147}$$

Eventually, for the hyperparameters of the GPs:

$$\boldsymbol{\theta}_c = \underset{\boldsymbol{\theta}_c}{\mathrm{argmax}}\left[-\frac{1}{2}\log|\mathbf{K}_c| - \frac{1}{2}\mathrm{tr}(\mathbf{K}_c^{-1}(\boldsymbol{\Sigma}_c + (\boldsymbol{\mu}_c - \mathbf{a}_c)(\boldsymbol{\mu}_c - \mathbf{a}_c)^T))\right] \tag{148}$$

### 1.2.3 A joint update for $q(\boldsymbol{f}_c)$ and $\boldsymbol{\theta}_c$

As explained by Titsias and Lazaro-Gredilla [5], the update for the hyperparameters $\boldsymbol{\theta}_c$ which parameterize $\mathbf{K}_c$ is problematic for two reasons. Firstly, it requires the inverse of $\mathbf{K}_c$ and this is numerically unstable in computer precision. Such a problem can be partially overcome by adding a small amount of jitter into the diagonal of $\mathbf{K}_c$, but this is not ideal. Secondly, the update of the hyperparameters $\boldsymbol{\theta}_c$ strongly depends on $\boldsymbol{\mu}_c\boldsymbol{\mu}_c^T$ and $\boldsymbol{\Sigma}_c$. This update can be slow because $\boldsymbol{\mu}_c\boldsymbol{\mu}_c^T$ and $\boldsymbol{\Sigma}_c$ depends on the kernel matrix $\mathbf{K}_c^{\text{old}}$ evaluated at the old values of the hyperparameters $\boldsymbol{\theta}_c^{\text{old}}$. To resolve this, we would like to update simultaneously somehow $\boldsymbol{\theta}_c$ and both $\boldsymbol{\mu}_c\boldsymbol{\mu}_c^T$ and $\boldsymbol{\Sigma}_c$, i.e. the factor $q(\boldsymbol{f}_c)$. This can be done in an elegant and efficient way using a Marginalized Variational step.

We would like to perform a joint optimization update for $(q(\boldsymbol{f}_c), \boldsymbol{\theta}_c)$ in a way that the factor $q(\boldsymbol{f}_c)$ is marginalized / removed optimally from the optimization problem. We write the variational lower bound as follows:

$$\mathcal{L}(\boldsymbol{\theta}_c) = \int q(\boldsymbol{f}_c)q(\boldsymbol{\Theta})\log \frac{p(\mathbf{Y}|\boldsymbol{f}_c, \boldsymbol{\Theta})p(\boldsymbol{\Theta})p(\boldsymbol{f}_c)}{q(\boldsymbol{f}_c)q(\boldsymbol{\Theta})}\, d\boldsymbol{f}_c d\boldsymbol{\Theta} \tag{149}$$

where $\boldsymbol{\Theta}$ are all random variables excluding $\boldsymbol{f}_c$ and $q(\boldsymbol{\Theta})$ their variational distribution. Given that we wish to update the factor $q(\boldsymbol{f}_c)$ and the kernel matrix $\mathbf{K}_c$, the above is written as

$$\mathcal{L}(\boldsymbol{\theta}_c) = \int q(\boldsymbol{f}_c)q(\boldsymbol{\Theta})\log \frac{p(\mathbf{Y}|\boldsymbol{f}_c, \boldsymbol{\Theta})p(\boldsymbol{f}_c)}{q(\boldsymbol{f}_c)}\, d\boldsymbol{f}_c d\boldsymbol{\Theta} + \text{const.} \tag{150}$$

$$= \int q(\boldsymbol{f}_c) \left( \int q(\boldsymbol{\Theta}) \log[p(\mathbf{Y}|\boldsymbol{f}_c, \boldsymbol{\Theta})p(\boldsymbol{f}_c)]\, d\boldsymbol{\Theta} \right) d\boldsymbol{f}_c - \int q(\boldsymbol{f}_c) \log q(\boldsymbol{f}_c)\, d\boldsymbol{f}_c + \text{const.} \tag{151}$$

$$= \int q(\boldsymbol{f}_c) \log \tilde{q}(\boldsymbol{f}_c)\, d\boldsymbol{f}_c - \int q(\boldsymbol{f}_c) \log q(\boldsymbol{f}_c)\, d\boldsymbol{f}_c + \text{const.} \tag{152}$$

where we have defined a new function $\tilde{q}(\boldsymbol{f}_c)$ such that

$$\log \tilde{q}(\boldsymbol{f}_c) = \int q(\boldsymbol{\Theta}) \log[p(\mathbf{Y}|\boldsymbol{f}_c, \boldsymbol{\Theta})p(\boldsymbol{f}_c)]\, d\boldsymbol{\Theta} = \int q(\boldsymbol{\Theta}) \log p(\mathbf{Y}|\boldsymbol{f}_c, \boldsymbol{\Theta})\, d\boldsymbol{\Theta} + \log p(\boldsymbol{f}_c) \tag{153}$$

It is strictly positive and by normalizing it, we in fact define a new distribution $q^*(\boldsymbol{f}_c)$:

$$q^*(\boldsymbol{f}_c) = \frac{\tilde{q}(\boldsymbol{f}_c)}{\int \tilde{q}(\boldsymbol{f}_c)\, d\boldsymbol{f}_c} \tag{154}$$

and thus

$$\log q^*(\boldsymbol{f}_c) = \log \tilde{q}(\boldsymbol{f}_c) - \log \int \tilde{q}(\boldsymbol{f}_c)\, d\boldsymbol{f}_c \tag{155}$$

Notice that the last term on the right-hand side is a constant since $\boldsymbol{f}_c$ is integrated out. As a result, the lower bound takes the form:

$$\mathcal{L}(\boldsymbol{\theta}_c) = \log \int \tilde{q}(\boldsymbol{f}_c)\, d\boldsymbol{f}_c + \int q(\boldsymbol{f}_c) \log q^*(\boldsymbol{f}_c)\, d\boldsymbol{f}_c - \int q(\boldsymbol{f}_c) \log q(\boldsymbol{f}_c)\, d\boldsymbol{f}_c + \text{const.} \tag{156}$$

We recognize the negative Kullback-Leibler divergence between $q(\boldsymbol{f}_c)$ and $q^*(\boldsymbol{f}_c)$:

$$\mathcal{L}(\boldsymbol{\theta}_c) = \log \int \tilde{q}(\boldsymbol{f}_c)\, d\boldsymbol{f}_c - \text{KL}\left(q(\boldsymbol{f}_c), q^*(\boldsymbol{f}_c)\right) + \text{const.} \tag{157}$$

Thus the optimal $q(\boldsymbol{f}_c)$ is simply given by $q^*(\boldsymbol{f}_c)$:

$$q^*(\boldsymbol{f}_c) \propto \mathrm{e}^{\mathrm{E}[\log\, p(\mathbf{Y}|\boldsymbol{f}_c, \boldsymbol{\Theta})]_{q(\boldsymbol{\Theta})}} p(\boldsymbol{f}_c) \tag{158}$$

The constant of proportionality is found by normalizing the distribution.

Notice that those are the classical steps that leads to (19) since for the factor $q(\boldsymbol{f}_c)$, the lower bound is:

$$\mathcal{L}(q(\boldsymbol{f}_c))=\int q(\boldsymbol{f}_c)q(\boldsymbol{\Theta})\log \frac{p(\mathbf{Y},\mathbf{W})}{q(\boldsymbol{f}_c)}\, d\boldsymbol{f}_c d\boldsymbol{\Theta} + \text{const.} \tag{159}$$

and from there we would arrive at

$$\log q^*(\boldsymbol{f}_c) = \int q(\boldsymbol{\Theta}) \log p(\mathbf{Y},\mathbf{W})\, d\boldsymbol{\Theta} \tag{160}$$

From the factorization of the joint probability density function (21), we would recover (158). The optimal distribution $q^*(\boldsymbol{f}_c)$ is still given by (35). The main difference in the treatment lies in that we have explicitly found the value reached by the lower bound by optimizing $q(\boldsymbol{f}_c)$:

$$\mathcal{L}(\boldsymbol{\theta}_c)=\log \int \tilde{q}(\boldsymbol{f}_c)\, d\boldsymbol{f}_c + \text{const.} \tag{161}$$

$$=\log \int \mathrm{e}^{\mathrm{E}[\log\, p(\mathbf{Y}|\boldsymbol{f}_c,\boldsymbol{\Theta})]_{q(\boldsymbol{\Theta})}}\, p(\boldsymbol{f}_c)\, d\boldsymbol{f}_c + \text{const.} \tag{162}$$

$$=\log \int \mathrm{e}^{\mathrm{E}[\log\, p(\mathbf{Y}|\boldsymbol{f}_c,\boldsymbol{\Theta})]_{q(\boldsymbol{\Theta})}}\, \mathcal{N}(\boldsymbol{f}_c|\mathbf{a}_c,\mathbf{K}_c)\, d\boldsymbol{f}_c + \text{const.} \tag{163}$$

In the classical derivation, the term on the right-hand side is simply put in the constant term since it does not depends on $\boldsymbol{f}_c$. But this is the only term that contains $\mathbf{K}_c$. Hence we can tune in $\boldsymbol{\theta}_c$ so that this term is maximized and we have the guarantee that the optimal distribution $q^*(\boldsymbol{f}_c)$ is still given by (35).

This lower bound is analytically tractable:

$$\mathcal{L}(\boldsymbol{\theta}_c)=\log \int \mathcal{N}(\boldsymbol{f}_c|\boldsymbol{\mu}_{F_1},\mathbf{K}_{F_1})\, \mathcal{N}(\boldsymbol{f}_c|\mathbf{a}_c,\mathbf{K}_c)\, d\boldsymbol{f}_c + \text{const.} \tag{164}$$

where $\boldsymbol{\mu}_{F_1}$ and $\mathbf{K}_{F_1}$ refer to (34) and (33) respectively. The product of two Gaussian distributions is an unnormalized Gaussian. The normalization factor defines another Gaussian distribution and thus:

$$\mathcal{L}(\boldsymbol{\theta}_c)=\log \mathcal{N}(\boldsymbol{\mu}_{F_1}|\mathbf{a}_c,\mathbf{K}_c+\mathbf{K}_{F_1}) + \text{const.} \tag{165}$$

It is the same as:

$$\mathcal{L}(\boldsymbol{\theta}_c)=\log \mathcal{N}(\mathbf{y}|\mathbf{a}_c,\mathbf{K}_c+\mathbf{B}_c) + \text{const.} \tag{166}$$

where the diagonal matrix $\mathbf{B}_c$ is given by

$$\mathbf{B}_c^{-1} = \frac{1}{\sigma_c^2}\mathrm{diag}\,[\gamma_{nc}]_{n=1}^N \tag{167}$$

This was done for a given factor $q(\boldsymbol{f}_c)$. The above is now optimized wrt $\boldsymbol{\theta}_c$ and this can be done using any standard GP implementation for maximizing the marginal likelihood, keeping fixed the noise variance $\mathbf{B}_c$. The optimization requires the inverse of $\mathbf{K}_c+\mathbf{B}_c$ which often will be numerically stable due to the addition of $\mathbf{B}_c$ in the diagonal of $\mathbf{K}_c$.

Once the optimization is completed, we evaluate the final value of the factor $q(\boldsymbol{f}_c)$ and then continue with other variational EM updates.

### 1.3 Predictive density

The predictive density distribution at an unseen input $\mathbf{x}^*$ is given by

$$p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{Y}) = \sum_{c=1}^{C} p(z_{*c} = 1|\mathbf{x}_*, \mathbf{X}, \mathbf{Y})\ p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{Y}, z_{*c} = 1) \tag{168}$$

We have

$$p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{Y}, z_{*c} = 1) = \int p(y_*|\mathbf{x}_*, \boldsymbol{f}_c)\ p(\boldsymbol{f}_c|\mathbf{X}, \mathbf{Y})\ d\boldsymbol{f}_c \tag{169}$$

We approximate it by replacing the true posterior distribution $p(\boldsymbol{f}_c|\mathbf{X}, \mathbf{Y})$ with its variational approximation $q(\boldsymbol{f}_c)$ given by (35). Furthermore

$$p(y_*|\mathbf{x}_*, \boldsymbol{f}_c) = \mathcal{N}(y_*|a_c + \mathbf{k}_c(\mathbf{x}_*, \mathbf{X})^T \mathbf{K}_c^{-1}(\boldsymbol{f}_c - \mathbf{a}_c), k_c(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_c(\mathbf{x}_*, \mathbf{X})^T \mathbf{K}_c^{-1} \mathbf{k}_c(\mathbf{x}_*, \mathbf{X}) + \sigma_c^2) \tag{170}$$

Using the formulae from Appendix A, we arrive at [1] :

$$p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{Y}, z_{*c} = 1) = \mathcal{N}(y_*|\mu_*^c, \sigma_{*c}^2) \tag{172}$$

where

$$\mu_*^c = a_c + \mathbf{k}_c(\mathbf{x}_*, \mathbf{X})^T((\mathbf{K}_c + \mathbf{B}_c)^{-1}\mathbf{y} - \mathbf{K}_c^{-1}\mathbf{a}_c) \tag{173}$$

$$\sigma_{*c}^2 = k_c(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_c(\mathbf{x}_*, \mathbf{X})^T(\mathbf{K}_c + \mathbf{B}_c)^{-1}\mathbf{k}_c(\mathbf{x}_*, \mathbf{X}) + \sigma_c^2 \tag{174}$$

The term $\mathbf{K}_c^{-1}\mathbf{a}_c$ poses problem as $\mathbf{K}_c$ is badly conditioned. We have found numerically that to a good approximation we could replace it by $(\mathbf{K}_c + \mathbf{B}_c)^{-1}\mathbf{a}_c$ since it will later be weighted by the responsabilities. We also have

$$p(z_{*c} = 1|\mathbf{x}_*, \mathbf{X}, \mathbf{Y}) = \int p(z_{*c} = 1|\mathbf{x}_*, \mathbf{X}, \boldsymbol{v}, \mathbf{m}_c, \mathbf{R}_c)\ p(\boldsymbol{v}, \mathbf{m}_c, \mathbf{R}_c|\mathbf{X}, \mathbf{Y})\ d\boldsymbol{v}\ d\mathbf{m}_c\ d\mathbf{R}_c \tag{175}$$

We replace the marginal posterior $p(\boldsymbol{v}, \mathbf{m}_c, \mathbf{R}_c|\mathbf{X}, \mathbf{Y})$ by its variational approximation $q(\boldsymbol{v})\ q(\mathbf{m}_c)\ q(\mathbf{R}_c)$. This does not suffice to make the derivation tractable. Hence we also replace the average of $p(z_{*c} = 1|\mathbf{x}_*, \mathbf{X}, \boldsymbol{v}, \mathbf{m}_c, \mathbf{R}_c)$ with respect to $q(\boldsymbol{v})\ q(\mathbf{m}_c)\ q(\mathbf{R}_c)$ by its value at the variational posterior mean:

$$p(z_{*c} = 1|\mathbf{x}_*, \mathbf{X}, \mathbf{Y}) \approx p(z_{*c} = 1|\mathbf{x}_*, \mathbf{X}, \mathrm{E}[\boldsymbol{v}], \mathrm{E}[\mathbf{m}_c], \mathrm{E}[\mathbf{R}_c]) \tag{176}$$

Consequently, the posterior mixing proportions are no longer random variables: They are replaced by their expectation under the variational distribution:

$$\omega_c \approx \mathrm{E}[\omega_c] = \mathrm{E}\left[v_c \prod_{c'=1}^{c-1}(1 - v_{c'})\right] = \mathrm{E}[v_c] \prod_{c'=1}^{c-1}(1 - \mathrm{E}[v_{c'}]) \tag{177}$$

since the $v_c$ are independent under the variational posterior. So we have

$$p(z_{*c} = 1|\mathbf{x}_*, \mathbf{X}, \mathbf{Y}) = \frac{\mathrm{E}[\omega_c]\ \mathcal{N}(\mathbf{x}_*|\mathbf{g}_c, \mathbf{W}_c^{-1}/\nu_c)}{\sum_{c'=1}^{C} \mathrm{E}[\omega_{c'}]\ \mathcal{N}(\mathbf{x}_*|\mathbf{g}_{c'}, \mathbf{W}_{c'}^{-1}/\nu_{c'})} \tag{178}$$

Eq. (168) clearly shows that the predictive density for the model output is a weighted sum of Gaussian process predictions. The weights, given by (178), are function of the input $\mathbf{x}_*$.

We recall that

$$\mathrm{E}[v_c] = \frac{\beta_{c,1}}{\beta_{c,1} + \beta_{c,2}} \tag{179}$$

[1] For the derivation, we also made use of the Woodbury formula

$$\mathbf{B}_c^{-1} - (\mathbf{K}_c + \mathbf{B}_c)^{-1} = (\mathbf{B}_c + \mathbf{B}_c\mathbf{K}_c^{-1}\mathbf{B}_c)^{-1} \tag{171}$$

# 2 Implementation

The implementation builds upon the package **gplm** of Rasmussen when it goes to building the covariance matrices and optimizing the hyperparameters of the kernels. It is also no coincidence that our implementation looks a lot like the code developed by Titsias and Lazaro-Gredilla [5]. Although their model was quite different, our bottom algorithm shares the same structure (EM updates), and so we could adopt and adapt the structure of their code.

K-means and Gaussian Mixture Model (GMM) clustering are part of NETLAB. GMM are trained using the EM algorithm [29].

The gamma function is available in Matlab / Octave natively. In addition, Matlab supports also natively the Digamma function, Octave does not. To make things worse, we also make use of the Trigamma function that is available neither in Matlab nor in Octave. We borrowed the Matlab/Octave routine **psin** written by Godfrey [24] that computes any polygamma function.

## 2.1 Model creation, initialization and training

The model is created, initialized and trained in **imgpTrain**:

```
1  function [ model vardist lb ] = imgpTrain(X,Y,kernel,C,noise,delta,options)
2
3  model.type = 'imgpmodel';
4
5  [ N D ] = size(X);
6  [ N Q ] = size(Y);
7  model.X = X;
8  model.Y = Y;
9  model.N = N;  % no. of samples
10 model.Q = 1;  % output dimension
11 assert( model.Q == 1 );
12 model.D = D;  % input dimension
13 model.Likelihood.type = 'Gaussian';
14 model.Likelihood.noise = 'heterosc';  % all features have the same target noise
15 %model.Likelihood.sigma2 is initialized a bit later
16 model.Likelihood.minSigma2 = 1e-10;
17 if length(kernel) == 1
18   kernel = repmat(kernel,1,C);
19 else
20   if length(kernel) ~= C
21     error('specify either a single kernel or as many kernel as C');
22   end
23 end
24 model.GP = {};
25 for c = 1:C
26   model.GP{c}.covfunc = kernel{c};
27   switch model.GP{c}.covfunc
28     case 'covSEard'
29       dd = log((max(X)-min(X))'/2); % log of length-scales, column vector
30       dd(dd == -Inf) = 0;
31       model.GP{c}.logtheta(1:D,1) = (1+0.1*(rand()-0.5))*(dd+0.1*(dd==0)) - mod(C,2)*rand();
32       model.GP{c}.logtheta(D+1,1) = 0;  % the Y are supposed to be re-scaled as normally distributed,
33                                         % so v0 = 1 is a good guess
34       model.GP{c}.nParams = length(model.GP{c}.logtheta);
35
36     otherwise
37       char(model.GP{c}.covfunc)
38       error('Unknown covariance type')
39   end
40   model.GP{c}.mean = 0.;
41 end
```

```
42
43  dispLB = options(1);  % display lower bound during training
44  dispEvery = 1;
45  kernLearn = options(2); % learn kernel hyperparameters (0 for not learning)
46  meanLearn = options(16); % learn the mean of the GPs
47  sigmaLearn = options(4);  % learn target noise (0 for not learning)
48  if sigmaLearn
49    model.Likelihood.sigma2 = noise;
50    if noise == 0
51      model.Likelihood.sigma2 = mean(var(Y))*(5e-2)^2;  % 5pc
52    end
53  else
54    model.Likelihood.sigma2 = noise;
55    if model.Likelihood.sigma2 < model.Likelihood.minSigma2
56      fprintf(1,'resetting target noise to %g\n', model.Likelihood.minSigma2);
57      model.Likelihood.sigma2 = model.Likelihood.minSigma2;
58    end
59  end
60  deltaLearn = options(6);  % learn delta (0 for not learning)
61  if deltaLearn
62    vardist.delta = 0.1;
63    if debug
64      fprintf(1,'delta will be learned: initial delta = %g\n', vardist.delta);
65    end
66  else
67    vardist.delta = delta;
68    if vardist.delta == 0
69      fprintf(1,'resetting delta to 1e-8\n');
70      vardist.delta = 1e-8;
71    end
72    if debug
73      fprintf(1,'delta is frozen to %g\n', vardist.delta);
74    end
75  end
76  nu0Learn = options(8);  % learn nu0 (0 for not learning)
77  W0Learn = options(9); % learn W0 (0 for not learning)
78  labelReordering = options(10);
79  iter = options(11); % number of variational EM iterations
80
81  learnKernEvery = 4;
82
83  % initialize factors
84
85  vardist.C = C;  % truncation threshold
86  assert( C <= N );
87  if options(15) == 0 % uniform initialization
88    vardist.gamma = ones(N,C)/C;
89    if meanLearn
90      meanY = mean(Y);
91      varY = var(Y);
92      for c = 1:C
93        model.GP{c}.mean = meanY + sqrt(varY)/10*randn();
94      end
95    end
96  else
97    p = randperm(N);
98    kmeans_options = foptions();
99    if options(15) == 1 % initialization using kmeans clustering in dimension D
100     [mix.centres, kmeans_options, post] = kmeans(X(p(1:C),:), X, kmeans_options);
101     vardist.gamma = post; % hard assignments
102   elseif options(15) == 2 % initialization using kmeans clustering in dimension D+1
103     [mix.centres, kmeans_options, post] = kmeans([ X(p(1:C),:) Y(p(1:C)) ], [ X Y ], kmeans_options);
```

```
104       vardist.gamma = post; % hard assignments
105     else % initialization using a GMM in dimension D+1
106       mix = gmm(D+1,C,'full');
107       kmeans_options(14) = 5; % just 5 iterations of K-means
108       mix = gmminit(mix, [ X Y ], options);
109       kmeans_options(14) = 15; % Max. number of iterations
110       [mix, kmeans_options] = gmmem(mix, [ X Y ], kmeans_options);
111       vardist.gamma = gmmpost(mix, [ X Y ]); % soft assignments
112       [ m mid ] = max(vardist.gamma, [], 2);
113     end
114
115     if options(15) ~= 3
116       post = logical(post);
117     end
118
119     if meanLearn
120       for c = 1:C
121         if options(15) ~= 3
122           model.GP{c}.mean = mean(Y(post(:,c)));
123         else
124           model.GP{c}.mean = mix.centres(c,D+1);
125         end
126         fprintf(1,'mean of component %i: %f\n', c, model.GP{c}.mean);
127       end
128     end
129
130     vardist.gamma = vardist.gamma ./ repmat( sum(vardist.gamma,2), 1, C );
131     vardist.gamma( vardist.gamma < 1e-40 ) = 1e-40;
132   end
133   vardist.mu = zeros(N,C);
134   vardist.diagSigma = zeros(N,C);
135   vardist.B = model.Likelihood.sigma2./vardist.gamma;
136   vardist.eta1 = 1e-3;  % vague Gamma prior if a << 1
137   vardist.eta2 = 1e-3;
138   vardist.beta1 = (1-vardist.delta+N/C)*ones(C-1,1);
139   vardist.beta2 = vardist.eta1/vardist.eta2*ones(C-1,1);
140   vardist.etahat1 = vardist.eta1;
141   vardist.etahat2 = vardist.eta2;
142   mu_x = mean(X); % row vector
143   R_x = zeros(D,D);
144   for n = 1:N
145     R_x = R_x + (X(n,:)-mu_x)'*(X(n,:)-mu_x);
146   end
147   R_x = R_x / (N-1) / 10;
148   vardist.g0 = mu_x';
149   vardist.invG0 = R_x;
150   vardist.G0 = inv(R_x);
151   vardist.nu0 = D*ones(C,1);
152   vardist.W0 = repmat(inv(R_x)/D, [ 1 1 C ]);
153   for c = 1:C
154     vardist.invW0(:,:,c) = inv(vardist.W0(:,:,c));
155   end
156   vardist.g = gsamp(vardist.g0, vardist.invG0, C)'; % D x C
157   vardist.G = repmat(vardist.invG0, [ 1 1 C]);
158   vardist.nu = vardist.nu0;
159   vardist.W = vardist.W0; % D x D x C
160   u = zeros(N,C);
161
162   % iterate
163
164   lb = zeros(iter,1);
165   LBold = -Inf; % at the onset, the lower bound cannot be computed;
```

```
166         % the reasong is that for the calculation of the lower bound
167         % we assume that Sigmac = Kc inv(Kc+Bc) Bc and muc = Sigmac (inv(B) y + inv(Kc) ac)
168         % but right at the onset muc = Sigmac = 0
169
170 for niter = 1:iter
171
172   gammaSum = sum(vardist.gamma);
173
174   % E step
175
176   vardist.B = model.Likelihood.sigma2./vardist.gamma;
177
178   for c = 1:C
179     Kc = feval(model.GP{c}.covfunc, model.GP{c}.logtheta, model.X);
180     [ Cc L ] = solve_chol_zeros(Kc, vardist.B(:,c), diag(vardist.B(:,c)), max(diag(Kc))+model.Likelihood.sig
181
182     % update factor q(f_c)
183     % we never store all the big matrices Sigma !
184
185     Sigma = Kc*Cc;
186     vardist.diagSigma(:,c) = diag(Sigma);
187     vardist.mu(:,c) = Sigma*((1./vardist.B(:,c)).*Y);
188     if meanLearn
189       cc = solve_chol_zeros(Kc, vardist.B(:,c), model.GP{c}.mean*ones(N,1), max(diag(Kc))+model.Likelihood.
190       vardist.mu(:,c) = vardist.mu(:,c) + vardist.B(:,c).*cc;
191     end
192
193     % construct u for the update of factor q(z_nc)
194     u(:,c) = -1/(2*model.Likelihood.sigma2)* ...
195               ( vardist.diagSigma(:,c) + (Y-vardist.mu(:,c)).^2 );
196   end
197
198   % update factor q(alpha)
199   log_omvc = digamma(vardist.beta2) - digamma(vardist.beta1+vardist.beta2);
200   vardist.etahat1 = vardist.eta1 + (C-1)*(1-vardist.delta);
201   vardist.etahat2 = vardist.eta2 - sum(log_omvc);
202
203   % update factor q(v_c)
204   vardist.beta1 = 1-vardist.delta+gammaSum(1:C-1)';
205   vardist.beta2 = vardist.etahat1/vardist.etahat2 + vardist.delta*[1:C-1]';
206   for c = 1:C-1
207     vardist.beta2(c) = vardist.beta2(c) + sum(gammaSum(c+1:C));
208   end
209
210   % update factor q(z_nc)
211   % for c=C, E[log(v_c)] = 0 since v_C = 1
212   for c = 1:C-1
213     u(:,c) = u(:,c) + ...
214               digamma(vardist.beta1(c)) - digamma(vardist.beta1(c)+vardist.beta2(c));
215   end
216   for c = 2:C
217     u(:,c) = u(:,c) + ...
218               sum( digamma(vardist.beta2(1:c-1)) - digamma(vardist.beta1(1:c-1)+vardist.beta2(1:c-1)) );
219   end
220   u = u - 1/2*log(2*pi*model.Likelihood.sigma2) - D/2*log(2*pi);
221   for c = 1:C
222     E_Rc = vardist.nu(c)*vardist.W(:,:,c);
223     E_log_Rc = log(det(vardist.W(:,:,c))) + D*log(2)+sum(digamma(0.5*(vardist.nu(c)+1-[1:D])));
224     Xc = X' - repmat(vardist.g(:,c),1,N);
225     u(:,c) = u(:,c) + ...
226               0.5*E_log_Rc - 0.5*( trace(vardist.G(:,:,c)*E_Rc) + sum(Xc.*(E_Rc*Xc),1)' );
227   end
```

```
228     for n = 1:N
229       [ mx idmx ] = max(u(n,:));
230       vardist.gamma(n,:) = exp(u(n,:)-mx)./sum(exp(u(n,:)-mx));
231     end
232     vardist.gamma( vardist.gamma<1e-30 ) = 1e-30;
233
234     % label re-ordering
235
236     if labelReordering
237       [ dummy I ] = sort(sum(vardist.gamma),'descend');
238
239       vardist.gamma = vardist.gamma(:,I);
240       vardist.diagSigma = vardist.diagSigma(:,I);
241       vardist.mu = vardist.mu(:,I);
242       vardist.G = vardist.G(:,:,I);
243       vardist.g = vardist.g(:,I);
244       vardist.W = vardist.W(:,:,I);
245       vardist.nu = vardist.nu(I);
246
247       vardist.B = model.Likelihood.sigma2./vardist.gamma;
248     end
249
250     % update factor q(m_c)
251     for c = 1:C
252       E_Rc = vardist.nu(c)*vardist.W(:,:,c);
253       vardist.G(:,:,c) = inv( vardist.G0 + sum(vardist.gamma(:,c))*E_Rc);
254       vardist.g(:,c) = vardist.G(:,:,c)*(vardist.G0*vardist.g0 + E_Rc*sum(repmat(vardist.gamma(:,c)',D,1
255     end
256
257     % update factor q(R_c)
258     for c = 1:C
259       vardist.nu(c) = vardist.nu0(c) + sum(vardist.gamma(:,c));
260       vardist.W(:,:,c) = vardist.invW0(:,:,c);
261       for n = 1:N
262         vardist.W(:,:,c) = vardist.W(:,:,c) + vardist.gamma(n,c)*(X(n,:)'-vardist.g(:,c))*(X(n,:)'-vardi
263       end
264       vardist.W(:,:,c) = inv( vardist.W(:,:,c) + sum(vardist.gamma(:,c))*vardist.G(:,:,c) );
265     end
266
267
268     % M step
269
270     if sigmaLearn
271       s = 0.;
272       for c = 1:C
273         s = s + (vardist.diagSigma(:,c) + (Y-vardist.mu(:,c)).^2)'*vardist.gamma(:,c);
274       end
275       model.Likelihood.sigma2 = s/sum(vardist.gamma(:));
276     end
277
278     if deltaLearn
279       [ tfdelta fX ] = minimize(log(vardist.delta/(1-vardist.delta)), 'imgp_delta', 5, vardist);
280       vardist.delta = exp(tfdelta)/(1+exp(tfdelta));
281     end
282
283     [ logeta1  fX ] = minimize(log(vardist.eta1) , 'imgp_eta1' , 5, vardist);
284     vardist.eta1 = exp(logeta1);
285
286     vardist.eta2 = vardist.eta1*vardist.etahat2/vardist.etahat1;
287
288     if nu0Learn
289       [ tfnu0  fX ] = minimize(log(vardist.nu0-(D-1)) , 'imgp_nu0' , 5, vardist, model);
```

```
290       vardist.nu0 = exp(tfnu0)+(D-1);
291     end
292
293     if ( D == 1 ) && W0Learn
294       S = zeros(D,D);
295       for c = 1:C
296         vardist.W0(:,:,c) = vardist.nu(c)/vardist.nu0(c)*vardist.W(:,:,c);
297         vardist.invW0(:,:,c) = inv(vardist.W0(:,:,c));
298       end
299     end
300
301     if kernLearn && ( mod(niter,learnKernEvery) == 0 )
302
303       fprintf(1,'optimizing the hyperparameters\n');
304
305       vardist.B = model.Likelihood.sigma2./vardist.gamma;
306
307       for c = 1:C
308         if model.GP{c}.nParams > 0
309
310           % kernel hyperparameters
311           if meanLearn
312             logtheta = [ model.GP{c}.logtheta(:) ; model.GP{c}.mean ];
313             [logtheta fX] = minimize(logtheta, 'gpr_fn_my', 5, model.GP{c}.covfunc, model.X, model.Y, vardist
314             model.GP{c}.logtheta = logtheta(1:end-1);
315             model.GP{c}.mean = logtheta(end);
316           else
317             [logtheta fX] = minimize(model.GP{c}.logtheta(:), 'gpr_fn_my', 5, model.GP{c}.covfunc, model.X, m
318             model.GP{c}.logtheta = logtheta;
319           end
320
321           % update q(f_c)
322           Kc = feval(model.GP{c}.covfunc, model.GP{c}.logtheta, model.X);
323           Cc = solve_chol_zeros(Kc, vardist.B(:,c), diag(vardist.B(:,c)), max(diag(Kc))+model.Likelihood.sigm
324           Sigma = Kc*Cc;
325           vardist.diagSigma(:,c) = diag(Sigma);
326           vardist.mu(:,c) = Sigma*((1./vardist.B(:,c)).*Y);
327           if meanLearn
328             cc = solve_chol_zeros(Kc, vardist.B(:,c), model.GP{c}.mean*ones(N,1), max(diag(Kc))+model.Likeliho
329             vardist.mu(:,c) = vardist.mu(:,c) + vardist.B(:,c).*cc;
330           end
331         end
332       end
333
334     end
335
336     % print lower bound
337
338     if dispLB == 1
339       LBnew = imgpLowerBound(model, vardist);
340       fprintf(1,'Iteration%4d/%4d  Fnew %11.6f Fold %11.6f Diffs %20.12f\n', ...
341               niter, iter, LBnew, LBold, LBnew-LBold);
342       if ( LBnew < LBold ) && ~labelReordering
343         error('non-increasing lower bound !');
344       end
345       if ( LBnew >= LBold ) && ( LBnew < LBold + 1e-4 ) && ( niter > 1 )
346         fprintf(1,'relative increase of lower bound below 1e-4\n');
347         return;
348       end
349       LBold = LBnew;
350       lb(niter) = LBnew;
351     else
```

```
352       if mod(niter,dispEvery) == 0
353         fprintf(1,'Iteration%4d/%4d\n',niter,iter);
354       end
355     end
356   end
```

Without proper initialization, variational methods can be easily trapped into local optima. So it pays off to initialize the model in a sensible way to overcome this issue.

The uniform initialization of $\gamma_{nc}$ at line 88 is typical but troublesome. Indeed, for large data sets, the noise defined by $\sigma_c^2/\gamma_{nc}$ ($= N\sigma_c^2$ at the onset) is also large. As a result, the mean predictions given by (173) will be over-smoothed. Generally, we acquire more samples to resolve fine scales and so the initialization should favor signals with a high-frequency content. The uniform initialization does the opposite. Furthermore, for large data sets, inversion of $\mathbf{K}_c + \mathbf{B}_c$ is computationally intensive.

So we prefer the initialization based on K-means clustering. The effort for the Cholesky decomposition of $\mathbf{K}_c + \mathbf{B}_c$ is reduced from a $O(N^3)$ to a $O(N_c^3)$ operation, where $N_c$ designates the approximate number of data that belongs to cluster $c$. K-means clustering is declined in two versions: We can cluster the input data $\mathbf{x}_n$, line 83, but if the samples are uniformly spaced, this initialization might be poor. In the second version advocated by Yuan and neubauer [26], the initialization is based on clusters of the combined data $\mathbf{x}_n$ and $y_n$ in the combined space of $D+1$-dimensions, line 86.

We can do even better, namely use a Gaussian Mixture Model with C clusters in the combined dimension, line 89-99.

Notice that the initial clusters might stronly overlap. If the output is multi-modal, this is welcome, otherwise it can be harmful. One avenue for circumventing this issue would be to resort to methods that oversegment pictures by producing super-pixels.

At line 31, for a standard GP, we would initialize the length-scales to one half of the range spanned by the training data as done by Lazaro-Gredilla and Figueiras-Vidal [28]. For the nonstationary GP, the correlation length-scales can be up to 2 orders of magnitude smaller.

At line 236-246, we reorder the labels as advocated by Kurihara et al. [27]. The stick-breaking prior assumes a certain ordering of the clusters. Since a permutation of the cluster labels changes the probability of the data, we should choose the optimal permutation resulting in the highest probability for the data. The optimal relabelling of the clusters is given by the one that orders the cluster sizes in decreasing order. Since the steak-breaking weights $v_c$ are left unchanged, the lower bound is not guaranteed to increase after this operation.

At line 186 and 187, we update $\mathbf{\Sigma}_c$ and $\boldsymbol{\mu}_c$ respectively. The old implementation for the update of $\mathbf{K}_c$ involved the Cholesky decomposition $\mathbf{L}\mathbf{L}^T = \mathbf{B}_c^{-\frac{1}{2}}\mathbf{K}_c\mathbf{B}_c^{-\frac{1}{2}} + \mathbf{I}$. We set $\mathbf{T} = \mathbf{L}^{-1}\mathbf{B}_c^{-\frac{1}{2}}\mathbf{K}_c$ and so $\mathbf{T}^T\mathbf{T} = \mathbf{K}_c(\mathbf{K}_c + \mathbf{B}_c)^{-1}\mathbf{K}_c$. Now we make use of the Kailath variant of the Woodbury identity (see eq. (148) of Petersen and Petersen [22]):

$$(\mathbf{A} + \mathbf{B}\mathbf{C})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{I} + \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} \tag{180}$$

Plugging in $\mathbf{A} = \mathbf{K}_c^{-1}$, $\mathbf{B} = \mathbf{C} = \mathbf{B}_c^{-\frac{1}{2}}$, we obtain:

$$\mathbf{\Sigma}_c = (\mathbf{K}_c^{-1} + \mathbf{B}_c^{-1})^{-1} = \mathbf{K}_c - \mathbf{K}_c(\mathbf{B}_c + \mathbf{K}_c)^{-1}\mathbf{K}_c = \mathbf{K}_c - \mathbf{T}^T\mathbf{T} \tag{181}$$

Notice that since $\mathbf{B}_c$ is diagonal, we have $\mathbf{B}_c^{-\frac{1}{2}}\mathbf{K}_c\mathbf{B}_c^{-\frac{1}{2}} = \mathbf{K}_c \otimes (\mathrm{diag}(\mathbf{B}_c^{-\frac{1}{2}})\mathrm{diag}(\mathbf{B}_c^{-\frac{T}{2}}))$ where diag(.) transforms a diagonal matrix into a vector and vice-versa.

This old implementation is efficient, the matrix $\mathbf{B}_c^{-\frac{1}{2}}\mathbf{K}_c\mathbf{B}_c^{-\frac{1}{2}} + \mathbf{I}$ is well conditioned but of size $N$-x-$N$. To make the computation feasible for large data sets, we solve for $\mathbf{\Sigma}_c = \mathbf{K}_c\left(\mathbf{K}_c + \mathbf{B}_c\right)^{-1}\mathbf{B}_c$. For a data $\mathbf{x}_n$ from a cluster $c' \neq c$, we have $z_{nc} \approx 0$ and so the corresponding element $\sigma_c^2/z_{nc}$ in the diagonal matrix $\mathbf{B}_c$ is almost infinite. It is easy to exploit this fact to derive a $O(N_c^3)$ method by considering the blockwise

inversion:

$$\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix}^{-1} = \begin{bmatrix}\mathbf{A}^{-1}+\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\\ -(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} & (\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\end{bmatrix} \tag{182}$$

The data are re-ordered so that $\mathbf{A}$ is the covariance between data points such that $\sigma_c^2/z_{nc}$ does not exceed a given threshold (set arbitrarily large to $10^{10}$). As a result, $\mathbf{D}$ has all its diagonal elements greater that this threshold and so

$$\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix}^{-1} \approx \begin{bmatrix}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}\mathbf{D}^{-1}\\ -\mathbf{D}^{-1}\mathbf{C}\mathbf{A}^{-1} & \mathbf{D}^{-1}\end{bmatrix} \tag{183}$$

$\mathbf{D}^{-1}$ is well approximated by a diagonal matrix whose elements are the inverse of the diagonal elements of $\mathbf{D}$. The reason that the diagonal matrices $\mathbf{A}^{-1}\mathbf{B}\mathbf{D}^{-1}$ and its transpose $\mathbf{D}^{-1}\mathbf{C}\mathbf{A}^{-1}$ are not set to $\mathbf{0}$ is that the inverse multiplies $\mathbf{B}_c$ whose elements might be large as well. This operation is done in the function **solve_chol_zeros**:

```
function [ X L logdet r ] = solve_chol_zeros(M, D, Y, e, L)

p = (D < e);
r = sum(p);
s = n-r;

invD = 1./D(~p);  % of size s
A = M(p,p) + diag(D(p));
B = M(p,~p);  % of size r x s
if nargin < 5
  L = chol(A,'lower');
end
Y1 = Y(p,:);
Y2 = Y(~p,:);
U = L'\(L\Y1);
diag_invD_B_t = bsxfun(@times,B',invD); % = diag(invD)*B';
V1 = -L'\(L\(diag_invD_B_t'*Y2));
V2 = -diag_invD_B_t*U;  % = -diag(invD)*B'*U
W = bsxfun(@times,Y2,invD); % = diag(invD)*Y2;
X = zeros(size(Y));
if size(X,2) ~= 1
  X(p,:) = U+V1;
  X(~p,:) = V2 + W;
else
  X(p) = U+V1;
  X(~p) = V2+W;
end

logdet = sum(log(D(~p))) + 2*sum(log(diag(L)));
```

For the optimization of the hyperparameters, we modify the function **gpr_fn** from Rasmussen that returns for our purpose only the negative log likelihood and its partial derivatives w.r.t. the hyperparameters for a given vanilla GP. To be precise, we modify a clever hack of it done by Titsias and Lazaro-Gredilla [6]. Our task is barely different, namely, to minimize the negative of (166):

$$-\mathcal{L}(\boldsymbol{\theta}_c)=\frac{N}{2}\log(2\pi)+\frac{1}{2}\log|\mathbf{K}_c+\mathbf{B}_c|+\frac{1}{2}(\mathbf{y}-\mathbf{a}_c)^T\boldsymbol{\alpha}_c \tag{184}$$

where, following the notations of Rasmussen and Williams [23], we introduced $\boldsymbol{\alpha}_c=(\mathbf{K}_c+\mathbf{B}_c)^{-1}(\mathbf{y}-\mathbf{a}_c)$. The derivative w.r.t. a hyperparameter $\theta$ is given by

$$-\frac{\partial\mathcal{L}(\boldsymbol{\theta}_c)}{\partial\theta}=\frac{1}{2}\mathrm{tr}\left(\frac{\partial(\mathbf{K}_c+\mathbf{B}_c)}{\partial\theta}\left((\mathbf{K}_c+\mathbf{B}_c)^{-1}-\boldsymbol{\alpha}_c\boldsymbol{\alpha}_c^T\right)\right) \tag{185}$$

By setting the derivative w.r.t. the GP constant mean $\mathbf{a}_c(=a_c\mathbf{I})$ to zero, we get

$$a_c=\frac{\mathbf{1}^T(\mathbf{K}_c+\mathbf{B}_c)^{-1}\mathbf{y}}{\mathbf{1}^T(\mathbf{K}_c+\mathbf{B}_c)^{-1}\mathbf{1}} \tag{186}$$

```
function [out1, out2] = gpr_fn_my(logtheta, covfunc, x, y, Bc, nonzero_m)

if ischar(covfunc), covfunc = cellstr(covfunc); end % convert to cell if needed

[n, D] = size(x);
if nargin < 6 || ~nonzero_m
  nParams = length(logtheta);
  ac = zeros(n,1);
else
  nParams = length(logtheta)-1;
  ac = logtheta(end);
end

if eval(feval(covfunc{:})) ~= nParams
  error('Error: Number of parameters do not agree with covariance function')
end

Kc = feval(covfunc{:}, logtheta(1:nParams), x);
[ alpha Lc logdet ] = solve_chol_zeros(Kc, Bc, y-ac, max(diag(Kc))+1e10);

out1 = 0.5*sum((y-ac).*alpha) + 0.5*logdet + 0.5*n*log(2*pi);

if nargout == 2                % ... and if requested, its partial derivatives
  out2 = zeros(nParams,1);       % set the size of the derivative vector
  W = solve_chol_zeros(Kc, Bc, eye(n), max(diag(Kc))+1e10, Lc); % precompute for convenience
  W = W - alpha*alpha';
  for i = 1:nParams
    out2(i) = sum(sum(W.*feval(covfunc{:}, logtheta, x, i)))/2;
  end
  if nargin > 5 && nonzero_m % derivative w.r.t. the mean
    cc = solve_chol_zeros(Kc, Bc, ones(n,1), max(diag(Kc))+1e10, Lc);
    out2 = [ out2 ; cc'*(ac-y) ];
  end

end
```

The hyperparameter $\eta_1$ must be strictly positive, and so we proceed classically by maximizing the lower bound w.r.t. $\log(\eta_1)$. The gradient is given by

$$\frac{\partial\mathcal{L}_{\eta_1}}{\partial\log(\eta_1)}=\eta_1\frac{\partial\mathcal{L}_{\eta_1}}{\partial\eta_1} \tag{187}$$

```
function [f, df] = imgp_eta1(logeta1, vardist)

eta1 = exp(logeta1);
C = vardist.C;
E_log_alpha = digamma(vardist.etahat1)-log(vardist.etahat2);

f = eta1*log(vardist.eta2) ...
    -lgamma(eta1) ...
    +eta1*E_log_alpha;
f = -f;

if nargout > 1
  df = log(vardist.eta2)-digamma(eta1)+E_log_alpha;
```

```
14    df = df*eta1;
15    df = -df;
16  end
```

The hyperparameter $\delta$ is restricted to $[0:1)$ and so we consider the transformation $\log(\delta/(1-\delta))$, that is, the inverse transformation of the sigmoid $\frac{1}{1+e^{-\delta}}$ that maps $(-\infty,+\infty)$ to $(0,1)$. The gradient of the lower bound w.r.t. to this parameter is given by

$$\frac{\partial \mathcal{L}_\delta}{\partial \log(\delta/(1-\delta))} = \delta(1-\delta)\frac{\partial \mathcal{L}_\delta}{\partial \delta} \tag{188}$$

```
1   function [f, df] = imgp_delta(tfdelta, vardist)
2
3   delta = exp(tfdelta)/(1+exp(tfdelta));  % in (0,1)
4   C = vardist.C;
5   E_log_alpha = digamma(vardist.etahat1)-log(vardist.etahat2);
6   E_log_vc = digamma(vardist.beta1)-digamma(vardist.beta1+vardist.beta2);
7   E_log_omvc = digamma(vardist.beta2)-digamma(vardist.beta1+vardist.beta2);
8
9   f = -(C-1)*lgamma(1-delta) ...
10      +(C-1)*(1-delta)*E_log_alpha ...
11      +delta*(-sum(E_log_vc) + [1:C-1]*E_log_omvc);
12  f = -f;
13
14  if nargout > 1
15    df = (C-1)*digamma(1-delta) ...
16         -(C-1)*E_log_alpha ...
17         +(-sum(E_log_vc) + [1:C-1]*E_log_omvc);
18    df = df*(delta*(1-delta));
19    df = -df;
20  end
```

The hyperparameter $\nu_0$ is restricted to $(D-1:\infty)$ where $D$ is the dimension of the input space, and so we consider the transformation $\log(\nu_0-(D-1))$. The gradient of the lower bound w.r.t. to this parameter is given by

$$\frac{\partial \mathcal{L}_{\nu_0}}{\partial \log(\nu_0-(D-1))} = (\nu_0-(D-1))\frac{\partial \mathcal{L}_{\nu_0}}{\partial \nu_0} \tag{189}$$

```
1   function [f, df] = imgp_nu0(tfnu0, vardist, model)
2
3   D = model.D;
4   C = vardist.C;
5   E_log_Rc = zeros(C,1);
6   log_det_W = zeros(C,1);
7   for c = 1:C
8     log_det_W(c) = log(det(vardist.W(:,:,c)));
9     E_log_Rc(c) = log_det_W(c) + D*log(2)+sum(digamma(0.5*(vardist.nu(c)+1-[1:D])));
10  end
11
12  nu0 = exp(tfnu0)+(D-1); % in (D-1: \infty);
13
14  f = -nu0*C*D/2*log(2) ...
15      -nu0*C/2*log(det(vardist.W0)) ...
16    -C*lmvgamma(D,nu0/2) ...
17    +(nu0-D-1)/2*sum(E_log_Rc);
18  f = -f;
19
20  if nargout > 1
21    df = -C*D/2*log(2) ...
```

```
22          -C/2*log(det(vardist.W0)) ...
23        -1/2*C*mvdigamma(D,nu0/2) ...
24        +1/2*sum(E_log_Rc);
25    df = df*(nu0-(D-1));
26    df = -df;
27  end
```

The lower bound is computed as follows:

```
1   function lb = imgpLowerBound(model,vardist)
2
3   N = model.N;
4   D = model.D;
5   C = vardist.C;
6
7   E_alpha = vardist.etahat1/vardist.etahat2;
8   E_log_alpha = digamma(vardist.etahat1)-log(vardist.etahat2);
9   E_log_vc = digamma(vardist.beta1)-digamma(vardist.beta1+vardist.beta2);
10  E_log_omvc = digamma(vardist.beta2)-digamma(vardist.beta1+vardist.beta2);
11  gammaSum = sum(vardist.gamma);
12  delta = vardist.delta;
13  eta1 = vardist.eta1;
14  eta2 = vardist.eta2;
15  sig2 = model.Likelihood.sigma2;
16  E_log_Rc = zeros(C,1);
17  log_det_W = zeros(C,1);
18  for c = 1:C
19    log_det_W(c) = log(det(vardist.W(:,:,c)));
20    E_log_Rc(c) = log_det_W(c) + D*log(2)+sum(digamma(0.5*(vardist.nu(c)+1-[1:D])));
21  end
22
23  %F1
24  F1 = -1/2*log(2*pi*sig2)*sum(gammaSum);
25  for c = 1:C
26    F1 = F1 - 1/(2*sig2)* ...
27              vardist.gamma(:,c)'*( vardist.diagSigma(:,c) + (model.Y-vardist.mu(:,c)).^2 );
28  end
29
30  %F2
31  F2 = gammaSum(1:C-1)*E_log_vc;
32  for c = 2:C
33    F2 = F2 + gammaSum(c)*sum(E_log_omvc(1:c-1));
34  end
35  F2 = F2 - D/2*log(2*pi)*sum(gammaSum) ...
36          + 0.5*gammaSum*E_log_Rc;
37  for c = 1:C
38    E_Rc = vardist.nu(c)*vardist.W(:,:,c);
39    Xc = model.X' - repmat(vardist.g(:,c),1,N);
40    F2 = F2 - 0.5*sum(Xc.*(E_Rc*Xc),1)*vardist.gamma(:,c) ...
41            - 0.5*trace(E_Rc*vardist.G(:,:,c))*gammaSum(c);
42  end
43
44  %F3
45  F3 = - (C-1)*gammaln(1-delta) ...
46       + (C-1)*(1-delta)*E_log_alpha ...
47     - delta*sum(E_log_vc) ...
48     + (E_alpha+delta*[1:C-1]-1)*E_log_omvc;
49
50  %F4
51  F4 = eta1*log(eta2) - gammaln(eta1) + (eta1-1)*E_log_alpha - eta2*E_alpha;
52
53  %F6
54  F6 = -D*C/2*log(2*pi) + C/2*log(det(vardist.G0));
```

```
F6s = zeros(D,D);
for c = 1:C
  F6s = F6s + vardist.G(:,:,c) + (vardist.g(:,c)-vardist.g0)*(vardist.g(:,c)-vardist.g0)';
end
F6 = F6 - 0.5*trace(vardist.G0*F6s);

%F7
F7 = -sum(vardist.nu0)*D/2*log(2);
for c = 1:C
  F7 = F7 - 0.5*vardist.nu0(c)*log(det(vardist.W0(:,:,c))) ...
      - lmvgamma(D,vardist.nu0(c)/2) ...
      + (vardist.nu0(c)-D-1)/2*E_log_Rc(c) ...
      - 0.5*trace(vardist.nu(c)*vardist.invW0(:,:,c)*vardist.W(:,:,c));
end

%E2
E2 = - sum(sum(vardist.gamma.*log(vardist.gamma+(vardist.gamma==0))));

%E3
E3 = - sum(gammaln(vardist.beta1+vardist.beta2)) ...
     + sum(gammaln(vardist.beta1)) + sum(gammaln(vardist.beta2)) ...
     - (vardist.beta1-1)'*E_log_vc - (vardist.beta2-1)'*E_log_omvc;

%E4
E4 = - vardist.etahat1*log(vardist.etahat2) ...
     + gammaln(vardist.etahat1) ...
   - (vardist.etahat1-1)*E_log_alpha ...
    + vardist.etahat2*E_alpha;

%E6
E6 = C*D/2*log(2*pi*exp(1));
for c = 1:C
  E6 = E6 + 0.5*log(det(vardist.G(:,:,c)));
end

%E7
E7 = D/2*(log(2)+1)*sum(vardist.nu);
for c = 1:C
  E7 = E7 + 0.5*vardist.nu(c)*log_det_W(c) ...
          + lmvgamma(D,vardist.nu(c)/2) ...
        - (vardist.nu(c)-D-1)/2*E_log_Rc(c);
end

%F5 and E5
F5plusE5 = C*N/2;
for c = 1:C
  Kc = feval(model.GP{c}.covfunc, model.GP{c}.logtheta, model.X);
  Bc = vardist.B(:,c);
  [ X L logdet ] = solve_chol_zeros(Kc, ...
                                    Bc, ...
                                    diag(Bc)+(model.Y-model.GP{c}.mean)*(vardist.mu(:,c)-model.GP{c}.mean)'
                                    max(diag(Kc))+sig2+1e10);
  F5plusE5 = F5plusE5 - 0.5*logdet + 1/2*sum(log(Bc)) ...
                      - 1/2*trace(X);
end

LBparts = [ F1 F2 F3 F4 F6 F7 E2 E3 E4 E6 E7 F5plusE5 ];
lb = sum(LBparts);

assert( F4+E4 <= 1e-10 );
assert( F6+E6 <= 1e-10 );
assert( F7+E7 <= 1e-10 );
```

```
117   assert( F5plusE5 <= 1e-10 );
```

To make some computations robust against round-off errors, we observe that from the lower bound

$$\frac{1}{2}\sum_{c=1}^{C}\log|\mathbf{\Sigma}_c| - \frac{1}{2}\sum_{c=1}^{C}\log|\mathbf{K}_c| = \frac{1}{2}\sum_{c=1}^{C}\log|\mathbf{K}_c^{-1}\mathbf{\Sigma}_c| \tag{190}$$

$$= \frac{1}{2}\sum_{c=1}^{C}\log|(\mathbf{K}_c+\mathbf{B}_c)^{-1}\mathbf{B}_c| \tag{191}$$

$$= -\frac{1}{2}\sum_{c=1}^{C}\log|\mathbf{K}_c+\mathbf{B}_c| + \frac{1}{2}\sum_{c=1}^{C}\log|\mathbf{B}_c| \tag{192}$$

Moreover, using (37) and (39), we have

$$\mathbf{K}_c^{-1}(\boldsymbol{\mu}_c-\mathbf{a}_c) = \mathbf{K}_c^{-1}\left(\mathbf{K}_c(\mathbf{K}_c+\mathbf{B}_c)^{-1}\mathbf{y} + (\mathbf{B}_c(\mathbf{K}_c+\mathbf{B}_c)^{-1}-\mathbf{I})\mathbf{a}_c\right) \tag{193}$$

$$= (\mathbf{K}_c+\mathbf{B}_c)^{-1}\mathbf{y} + \mathbf{K}_c^{-1}\mathbf{B}_c((\mathbf{K}_c+\mathbf{B}_c)^{-1}-\mathbf{B}_c^{-1})\mathbf{a}_c \tag{194}$$

We make use of the Woodbury identity:

$$(\mathbf{A}+\mathbf{A}\mathbf{B}^{-1}\mathbf{A})^{-1} = \mathbf{A}^{-1} - (\mathbf{B}+\mathbf{A})^{-1} \tag{195}$$

Thus

$$\mathbf{K}_c^{-1}\mathbf{B}_c((\mathbf{K}_c+\mathbf{B}_c)^{-1}-\mathbf{B}_c^{-1}) = -\mathbf{K}_c^{-1}\mathbf{B}_c(\mathbf{B}_c+\mathbf{B}_c\mathbf{K}_c^{-1}\mathbf{B}_c)^{-1} \tag{196}$$

$$= -\left((\mathbf{B}_c+\mathbf{B}_c\mathbf{K}_c^{-1}\mathbf{B}_c)\mathbf{B}_c^{-1}\mathbf{K}_c\right)^{-1} \tag{197}$$

$$= -(\mathbf{K}_c+\mathbf{B}_c)^{-1} \tag{198}$$

and so

$$\mathbf{K}_c^{-1}(\boldsymbol{\mu}_c-\mathbf{a}_c) = (\mathbf{K}_c+\mathbf{B}_c)^{-1}(\mathbf{y}-\mathbf{a}_c) \tag{199}$$

Finally

$$\mathrm{tr}(\mathbf{K}_c^{-1}(\mathbf{\Sigma}_c+(\boldsymbol{\mu}_c-\mathbf{a}_c)(\boldsymbol{\mu}_c-\mathbf{a}_c)^T)) = \mathrm{tr}((\mathbf{K}_c+\mathbf{B}_c)^{-1}(\mathbf{B}_c+(\mathbf{y}-\mathbf{a}_c)(\boldsymbol{\mu}_c-\mathbf{a}_c)^T)) \tag{200}$$

This is the trace of a symmetric positive definite matrix and thus is positive. If we were not doing a joint update for $q(\boldsymbol{f}_c)$ and $\boldsymbol{\theta}_c$, to maximize the lower bound w.r.t. to the means $a_c$, we would minimize this trace. The minimum would be obtained for

$$a_c = \frac{1}{2}\frac{\mathbf{1}^T(\mathbf{K}_c+\mathbf{B}_c)^{-1}(\mathbf{y}+\boldsymbol{\mu}_c)}{\mathbf{1}^T(\mathbf{K}_c+\mathbf{B}_c)^{-1}\mathbf{1}} \tag{201}$$

At lines 114-117, we check that

$$\mathrm{E}[\log p(\alpha)] - \mathrm{E}[\log q(\alpha)] = \int q(\alpha)\log p(\alpha)\,d\alpha - \int q(\alpha)\log q(\alpha)\,d\alpha = -\mathrm{KL}(q(\alpha)||p(\alpha)) \leq 0 \tag{202}$$

and

$$\mathrm{E}[\log p(\{\boldsymbol{f}_c\}_{c=1}^C)] - \mathrm{E}[\log q(\{\boldsymbol{f}_c\}_{c=1}^C)] = -\mathrm{KL}(q(\{\boldsymbol{f}_c\}_{c=1}^C)||p(\{\boldsymbol{f}_c\}_{c=1}^C)) \leq 0 \tag{203}$$

$$\mathrm{E}[\log p(\{\boldsymbol{m}_c\}_{c=1}^C)] - \mathrm{E}[\log q(\{\boldsymbol{m}_c\}_{c=1}^C)] = -\mathrm{KL}(q(\{\boldsymbol{m}_c\}_{c=1}^C)||p(\{\boldsymbol{m}_c\}_{c=1}^C)) \leq 0 \tag{204}$$

$$\mathrm{E}[\log p(\{\boldsymbol{R}_c\}_{c=1}^C)] - \mathrm{E}[\log q(\{\boldsymbol{R}_c\}_{c=1}^C)] = -\mathrm{KL}(q(\{\boldsymbol{R}_c\}_{c=1}^C)||p(\{\boldsymbol{R}_c\}_{c=1}^C)) \leq 0 \tag{205}$$

respectively. It would be wise to also check that

$$\mathrm{H}_q = -\int q(\mathbf{W}) \log q(\mathbf{W}) \, d\mathbf{W} \leq 0 \tag{206}$$

with the following piece of code:

```
assert( E2 + E3 + E4 + E5 + E6 + E7 > 0 );
```

but it is difficult since for numerical reasons we compute the sum of E5 and F5 and not each term separately.

### 2.1.1 Making predictions

Prediction of the mean and variance at an unseen input is done in **imgpPredict**:

```
function [ yp sig2 omega ypc ] = imgpPredict(model, vardist, Xtest, omega)

C = vardist.C;
D = model.D;
N = model.N;
Nstar = size(Xtest, 1);

if nargin < 4
  omega = ones(C,1);
  for c = 2:C
    omega(c) = omega(c-1)*(1-vardist.beta1(c-1)/(vardist.beta1(c-1)+vardist.beta2(c-1)));
  end
  omega(1:C-1) = omega(1:C-1).*vardist.beta1(1:C-1)./(vardist.beta1(1:C-1)+vardist.beta2(1:C-1));
else
  disp('omega is _given_ for predictions');
  assert( length(omega) == C );
end

yp = zeros(Nstar,1);
out1 = zeros(Nstar,1);

mix = gmm(D, C, 'full');
for c = 1:C
  mix.priors(c) = omega(c);
  mix.centres(c,:) = vardist.g(:,c)';
  mix.covars(:,:,c) = inv(vardist.W(:,:,c))/vardist.nu(c);
end
post = gmmpost(mix, Xtest);

for c = 1:C
  Kc = feval(model.GP{c}.covfunc, model.GP{c}.logtheta, model.X);
  [Kss, Kstar] = feval(model.GP{c}.covfunc, model.GP{c}.logtheta, model.X, Xtest);
  Bc = vardist.B(:,c);
  V = solve_chol_zeros(Kc, Bc, Kstar, max(diag(Kc))+model.Likelihood.sigma2+1e10);
  V = V';

  mustarc = model.GP{c}.mean + V*(model.Y(:)-model.GP{c}.mean);
  % wo/ approximation
  % mustarc = model.GP{c}.mean + V*(model.Y(:)-(Kc+diag(Bc))*inv(Kc+model.Likelihood.minSigma2*eye(N))*mode
  sigma2starc = Kss - sum(V.*Kstar',2) + model.Likelihood.sigma2;

  yp = yp + post(:,c).*mustarc;
  out1 = out1 + post(:,c).*(mustarc.^2+sigma2starc);
end

if nargout > 2
  ypc = zeros(Nstar,C);
```

```
48
49    for c = 1:C
50      Kc = feval(model.GP{c}.covfunc, model.GP{c}.logtheta, model.X);
51      [Kss, Kstar] = feval(model.GP{c}.covfunc, model.GP{c}.logtheta, model.X, Xtest);
52      Bc = vardist.B(:,c);
53      Lc = chol(Kc+diag(Bc),'lower');
54      V = (Lc'\(Lc\(Kstar)))';
55      ypc(:,c) = model.GP{c}.mean + V*(model.Y(:)-model.GP{c}.mean);
56    end
57  end
```

### 2.2 Demonstration program

The program **demimgp_sine** shows the performance of the IM-GP model on the function $\sin(p\pi x^q)$ on the interval $[-1:2]$:

We run the IM-GP on some well-known regression tasks. Data sets are *Kin-40k* (8 dimensions, 10000 training samples, 30000 testing) and *Pumadyn-32nm* (32 dimensions, 7168 training samples, 1024 testing), both artificially created using a robot arm simulator. We follow the procedure described in Lazaro-Gredilla [30]: Each problem is run ten times and averaged.

The *Pumadyn-32nm* problem can be seen as a test of the ARD capabilities of a regression model since only 4 out of the 32 input dimensions are relevant. Notice that Lazaro-Gredilla [30] initialized the length-scales of the standard GP on a subset of 1024 training data points. The standard GP equipped with the ARD squared exponential covariance function achieved a normalized MSE of circa 0.045 and succesfully singled out all inputs except [4,5,15,16]. We do not adhere to this strategy and run the standard GP on the full training set directly. After 50 iterations, we obtain a normalized MSE of 0.03. As expected, the standard GP is able to discreminate between the inputs and achieves an impressive RSM = 0.040. On the other hand, the zero-mean IM-GP with $C = 10$ gets RSM = 0.079 ( $\mathcal{L} = -136275$ ) after 100 iterations. Using a non-zero mean, it slightly improves to RSM = 0.076 ( $\mathcal{L} = -136269$ ). Strinkingly, all components have almost the same size.

Figures 12 and 13 show results for the function

$$f(x) = (s(1-s))^{\frac{1}{2}} \sin(2\pi(1+a)/(s+a)) \quad s \in [0:1] \tag{207}$$

with $a = 0.05$ using 1024 regularly spaced samples (demo program **demimgp_doppler**). Following Pintore and Holmes [31], the function is rescaled such that the variance is 7 and normal noise is added. Using as few as 128 samples, results are poor, Figures 14 and 15. The method developped by Pintore and Holmes [31] based on a spatially adaptive non-stationary covariance function yields far better results.

### 2.3 Acknowledgement

We are indebted to Titsias and Lazaro-Gredilla for putting the full version of their code online, to Godfrey for the Matlab / Octave routine to compute the polygamma functions and to Nabney for the Matlab Toolbox NETLAB.

## A Product of two Gaussian distributions

The following can be found in Quinonero-Candela and Rasmussen [3].

Consider the random variable $\mathbf{x}$ of size $n$ x 1 and the following product:

$$\mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{x}, \mathbf{\Sigma}_A)\, \mathcal{N}(\mathbf{x}|\mathbf{b}, \mathbf{\Sigma}_B) = z_c\, \mathcal{N}(\mathbf{x}|\mathbf{c}, \mathbf{\Sigma}_C)$$

where $\mathbf{y} \in R^m$ and $\mathbf{A}$ is thus a matrix of size $m$ x $n$. The product of two Gaussian distributions is proportional to a new Gaussian distribution with covariance and mean given by:

$$\mathbf{\Sigma}_C = (\mathbf{A}^T\mathbf{\Sigma}_A^{-1}\mathbf{A} + \mathbf{\Sigma}_B^{-1})^{-1} \quad \mathbf{c} = \mathbf{\Sigma}_C\ (\mathbf{A}^T\mathbf{\Sigma}_A^{-1}\mathbf{y} + \mathbf{\Sigma}_B^{-1}\mathbf{b})$$

The normalizing constant $z_c$ is Gaussian in the means $\mathbf{Ab}$ and $\mathbf{y}$:

$$z_c = (2\pi)^{-\frac{m}{2}}|\mathbf{\Sigma}_A + \mathbf{A}\mathbf{\Sigma}_B\mathbf{A}^T|^{-\frac{1}{2}}\ \exp(-\frac{1}{2}(\mathbf{y}-\mathbf{Ab})^T(\mathbf{\Sigma}_A + \mathbf{A}\mathbf{\Sigma}_B\mathbf{A}^T)^{-1}(\mathbf{y}-\mathbf{Ab}))$$

# B Some inequalities for the Gamma function

We recall the recurrence formula:

$$\Gamma(x+1) = x\Gamma(x) \tag{208}$$

From Merkle [12], we have that the functions

$$\log\,\Gamma(x) \tag{209}$$
$$\log\,\Gamma(x) - x\log\,x \tag{210}$$
$$\log\,\Gamma(x) - (x-\frac{1}{2})\log\,x \tag{211}$$
$$\tag{212}$$

are convex.

From Batir [10], we have:

$$\sqrt{2\mathrm{e}}\left(\frac{x+1/2}{e}\right)^{x+1/2} \leq \Gamma(x+1) \leq \sqrt{2\pi}\left(\frac{x+1/2}{e}\right)^{x+1/2} \quad \text{for} \quad x>0$$

$$x^x\mathrm{e}^{-x}\sqrt{2\pi(x+1/6)} \leq \Gamma(x+1) \leq x^x\mathrm{e}^{-x}\sqrt{2\pi(x+\mathrm{e}^2/(2\pi)-1)} \quad \text{for} \quad x \geq 1$$

Qi [18] listed (almost ?) all formulae for the ratio of two Gamma functions, notably Wendell's double inequality:

$$\left(\frac{x}{x+s}\right)^{1-s} \leq \frac{\Gamma(x+s)}{x^s\Gamma(x)} \leq 1 \quad \text{for} \quad 0<s<1,\ x>0 \tag{213}$$

a double inequality resulting from a convexity established by Lazarevic-Lupas:

$$\frac{s}{2} < \left[\frac{\Gamma(x+1)}{\Gamma(x+s)}\right]^{\frac{1}{1-s}} - x \leq [\Gamma(s)]^{\frac{1}{1-s}} \quad \text{for} \quad 0<s<1,\ x>0 \tag{214}$$

and a double inequality based on Ismail-Lorch-Muldoon's monotonicity:

$$x^{a-b} < \frac{\Gamma(x+a)}{\Gamma(x+b)} < \frac{x^{a-b}}{x_0^{a-b}}\frac{\Gamma(x_0+a)}{\Gamma(x_0+b)} \quad \text{for} \quad a>b\geq 0,\ a+b\geq 1,\ x\geq x_0>0 \tag{215}$$

According to Literka [21]

$$S(x) = \frac{1}{x^a}\frac{\Gamma(x+a)}{\Gamma(x)} \quad \text{for} \quad 0<a<1,\ x>0 \tag{216}$$

is an increasing function of $x$ and

$$\lim_{x\to\infty} S(x) = 1 \tag{217}$$

# C Some inequalities for the Digamma function

According to Dragomir et al. [15], the Digamma function

$$\psi(x) = \frac{d}{dx}\log\,\Gamma(x) = \frac{\Gamma'(x)}{\Gamma(x)}$$

is monotonic nondecreasing and concave on $(0, \infty)$. We also have

$$\psi(x+1) = \psi(x) + \frac{1}{x} \quad \text{for} \quad x > 0 \tag{218}$$

From Batir [10], we have:

$$\log(x+1/2) \leq \psi(x+1) \leq \log(x+\mathrm{e}^{-\gamma}) \quad \text{for} \quad x > 0 \tag{219}$$

$$\log(x+1/2) \leq \psi(x+1) \leq \log(x+\mathrm{e}^{1-\gamma}-1) \quad \text{for} \quad x \geq 1 \tag{220}$$

From Qi and Guo [19], we found the following inequalities:

$$\frac{1}{2x} - \frac{1}{12x^2} < \psi(x+1) - \log(x) < \frac{1}{2x} \quad \text{for} \quad x > 0 \tag{221}$$

$$\log(x) - \frac{1}{x} < \psi(x) < \log(x) - \frac{1}{2x} \quad \text{for} \quad x > 0 \tag{222}$$

$$\log(x+\frac{1}{2}) - \frac{1}{x} < \psi(x) < \log(1+x) - \frac{1}{x} \quad \text{for} \quad x > 0 \tag{223}$$

Qi [18] presented some Keckic-Vasic type inequalities:

$$\psi(x+\beta) \leq \log(x) + \frac{\beta}{x} \quad \text{for} \quad \beta > 0,\, x > 0 \tag{224}$$

$$\psi(x+\beta) \leq \log(x) + \frac{\beta - 1/2}{x} \quad \text{for} \quad \beta \geq 1,\, x > 0 \tag{225}$$

$$\tag{226}$$

According to Alzer [16], the function $x^2\psi'(x)$ is strictly convex on $(0, \infty)$.
Batir [11] derived the following limit

$$\lim_{x\to\infty}(x - \mathrm{e}^{\psi(x)}) = \frac{1}{2} \tag{227}$$

and Alzer[17]

$$\lim_{x\to\infty} x(\log(x) - \psi(x)) = \frac{1}{2} \tag{228}$$

Using the recurrence formulae (208) and (218), the following

$$\lim_{x\to 0}\frac{\Gamma(x)}{\psi(x)} = -1 \tag{229}$$

can easily be derived.
Guo and Qi [20] proved that

$$\lim_{x\to 0}(\psi(x) + \log(\mathrm{e}^{1/x} - 1)) = -\gamma \tag{230}$$

and

$$\lim_{x\to\infty}(\psi(x) + \log(\mathrm{e}^{1/x} - 1)) = 0 \tag{231}$$

## References


[1] S. Sun, X. Xu: Variational inference for infinite mixtures of Gaussian processes with applications to traffic flow prediction. IEEE Transactions on intelligent transportation systems, Vol. 12, No. 2, June 2011.

[2] S. Chatzis, Y. Demiris: Nonparametric mixtures of Gaussian processes with power-law behavior. IEEE Transactions on Neural Networks and Learning Systems, Vol. 23, No. 12, December 2012.

[3] J. Quionero-Candela, C.E. Rasmussen: Analysis of some methods for reduced rank Gaussian process regression. In R. Murray-Smith and R. Shorten, editors, Switching and Learning in Feedback Systems, pages 98-127. Springer, Berlin, Heidelberg, 2005.

[4] C. Bishop: Pattern recognition and machine leaning. Springer, 2007.

[5] M. K. Titsias, M. Lazaro-Gredilla: Spike and slab inference for multi-task and multiple kernel learning. NIPS, 24, 2012.

[6] M. K. Titsias, M. Lazaro-Gredilla: Supplementary material for: Spike and slab inference for multi-task and multiple kernel learning. Available online at http://www.well.ox.ac.uk/~mtitsias/papers/supplementaryvmtmkl.pdf

[7] D.M. Blei, M.I. Jordan: Variational inference for Dirichlet process mixtures. Bayesian Analysis, Volume 1, Number 1 (2006), 121-143.

[8] Available online at http://en.wikipedia.org/wiki/Beta_distribution#Moments_of_logarithmically_transformed_random_variables

[9] Z. Ma, A. Leijon: Bayesian Estimation of Beta Mixture Models with Variational Inference. IEEE Trans. Pattern Anal. Mach. Intell. 33(11): 2160-2173, 2011.

[10] N. Batir: Inequalities for the Gamma function. Available online at http://ajmaa.org/RGMIA/papers/v12n1/AderM-1.pdf

[11] N. Batir: Some new inequalities for gamma and polygamma functions. JIPAM. J. Inequal. Pure Appl. , 6(2005), no.4, Article 103, 9 pp.

[12] M. Merkle: Logarithmic convexity and inequalities for the Gamma function. Journal of mathematical Analysis and Applications 208, 369-380, 1996. Available online at http://milanmerkle.com/documents/radovi/JMAA-203.pdf

[13] M. Merkle: Conditions for convexity of a derivative and applications to the Gamma and Digamma function. Facta Universitatis (NIS), Ser. Math. Inform. 16, 13-20, 2001. Available online at http://facta.junis.ni.ac.rs/mai/mai16/f16-02.pdf

[14] C.D. Cantrell: Modern mathematical methods for physicists and Engineers. Cambridge University Press, 2000. Available online at http://www.utdallas.edu/eecs/booksite/pdf/appG.pdf

[15] S.S. Dragomir, R.P. Agarwal, N.S. Barnett: Inequalities for Beta and Gamma functions via some classical and new integral inequalities.

[16] H. Alzer: Inequalities for the Gamma function. Proceedings of the American Mathematical Society, Vol. 128, no. 1, pp. 141-147, 1999.

[17] H. Alzer: On some inequalities for the gamma and psi functions. Math. Comp. 66(1997), no.217, 373-389.

[18] F. Qi: Bounds for the ratio of two Gamma functions. Available online at http://rgmia.org/papers/v11n3/bounds-two-gammas.pdf.

[19] F. Qi, B.-N. Guo: An inequality involving the Gamma and Digamma functions. Arxiv:1101.4698v1.

[20] B.-N. Guo, F. Qi: Some properties of the Psi and Polygamma functions. Hacettepe Journal of Mathematics and Statistics, Vol. 39 (2), 219-231, 2010.

[21] Literka: A Remarkable Monotonic Property of the Gamma Function. Available online at http://www.literka.addr.com/mathcountry/gamma.htm

[22] K.B. Petersen, M.S. Petersen: The matrix Cookbook http://matrixcookbook.com

[23] C.E. Rasmussen, C.K.I. Williams: Gaussian processes for machine learning. The MIT Press, 2006.

[24] Special functions math library by P. Godfrey. Available at http://www.mathworks.com/matlabcentral/fileexchange/978-special-functions-math-library.

[25] J. Duan, M. Guindani, A. Gelfand: Generalized spatial Dirichlet process models. Biometrika (2007) 94 (4): 809-825. doi: 10.1093/biomet/asm071.

[26] C. Yuan, C. Neubauer: Variational mixture of Gaussian process experts Adv. Neural Inf. Process. Syst., Vol. 21, pp.1897-1904, 2009.

[27] K. Kurihara, M. Welling, Y.W. Teh: Collapsed variational Dirichlet mixture models IJCAI, 2007.

[28] M. Lazaro-Gredilla, A.R. Figueiras-Vidal: Inter-domain Gaussian processes for sparse inference using inducing features Advances in Neural Information Processing Systems 22, p. 1087-1095, 2010.

[29] I. Nabney: Netlab. Springer, 2004.

[30] M. Lzaro-Gredilla: Sparse Gaussian Processes for Large-Scale Machine Learning. PhD Thesis, Universidad Carlos III de Madrid, 2010.

[31] A. Pintore, C. Holmes: Spatially adaptive non-stationary covariance functions via spatially adaptive spectra. Available online at http://www.stats.ox.ac.uk/~cholmes/Reports/spectral_tempering.pdf

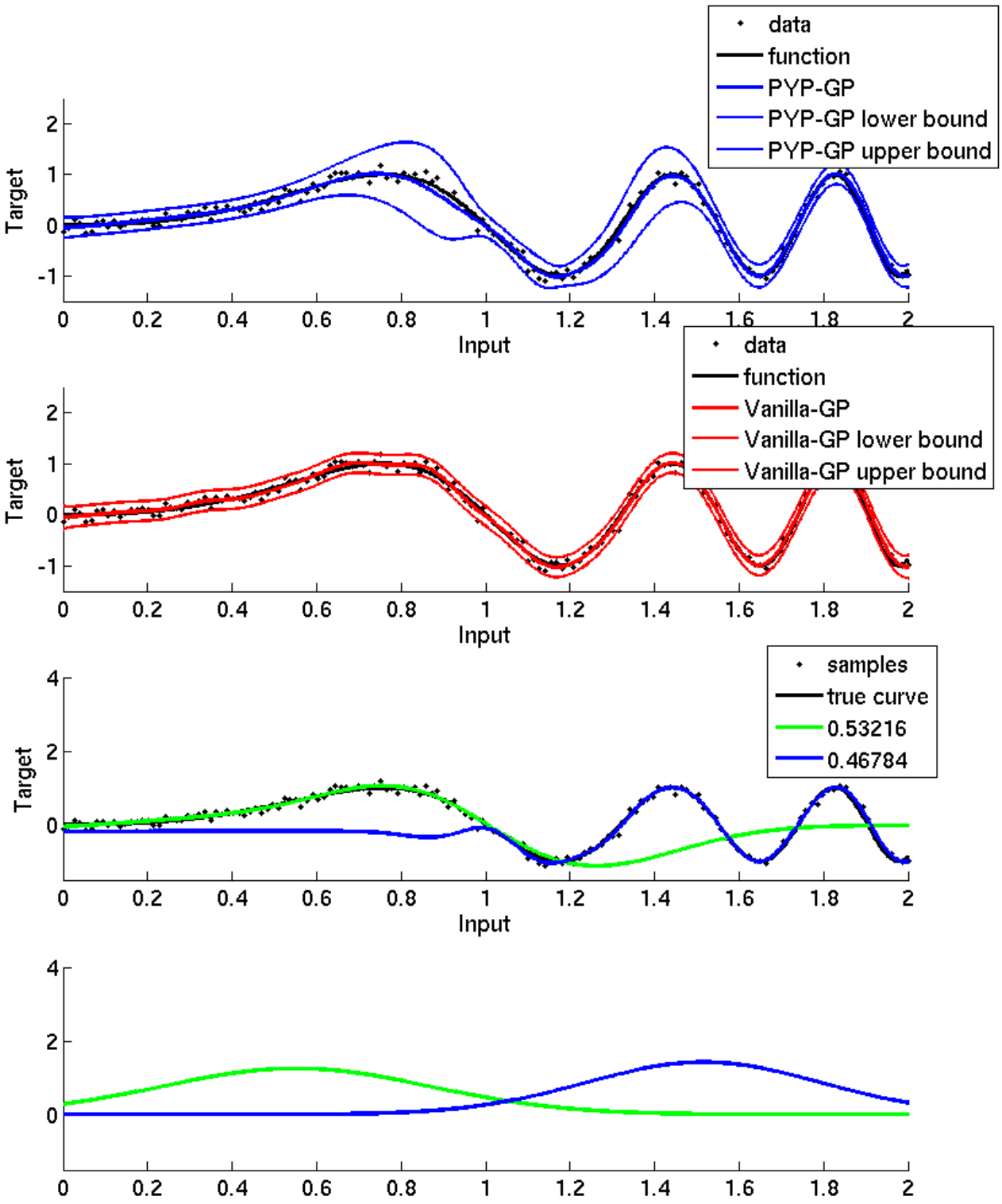


Figure 1: N=150,$\sigma_c^2 = 0.01$, $C = 2$,$f(x) = \sin(\pi x^{2.5})$, 50 iterations, with label re-ordering.

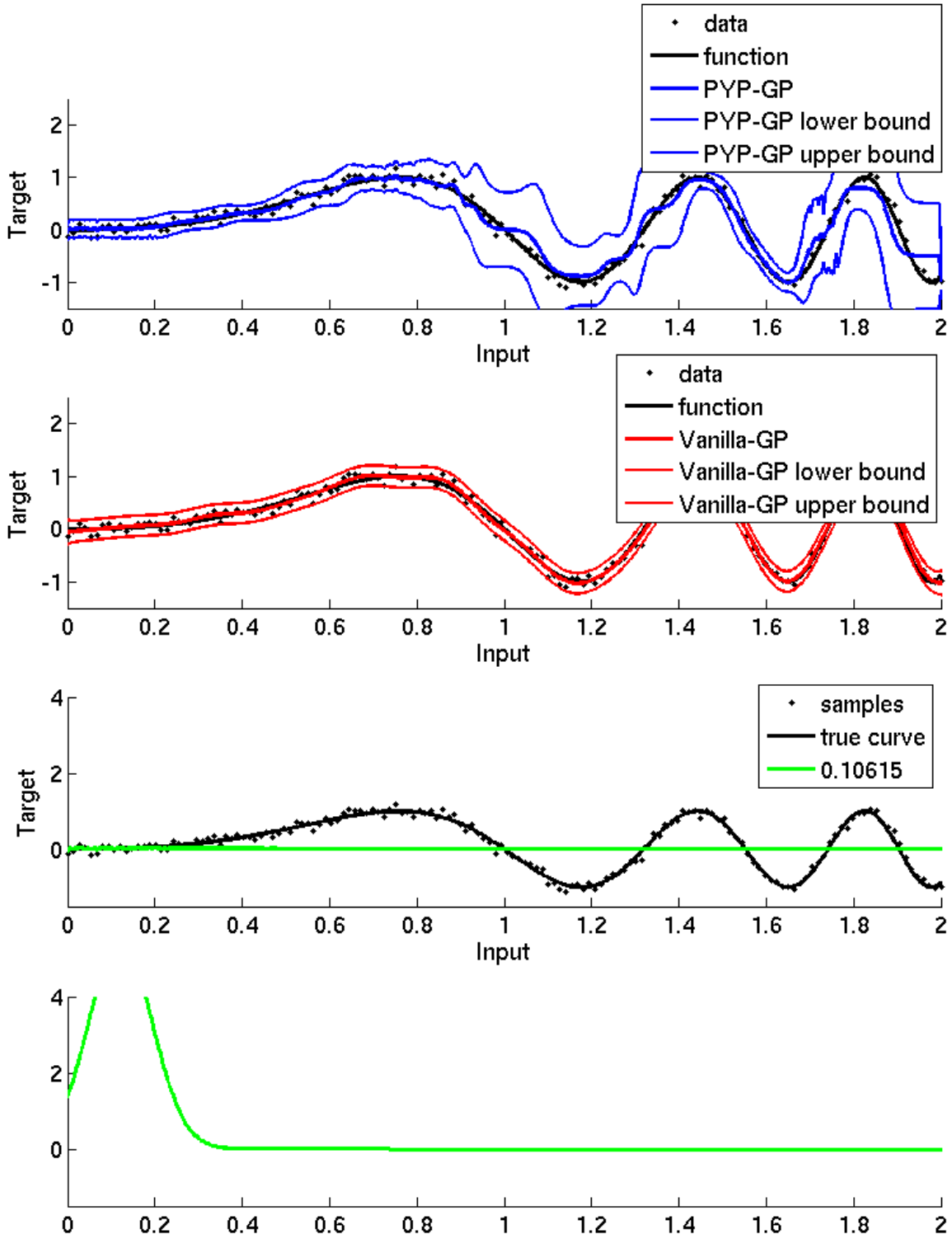


Figure 2: N=150,$\sigma_c^2 = 0.01$, $C = 20$,$f(x) = \sin(\pi x^{2.5})$, 100 iterations, without label re-ordering.

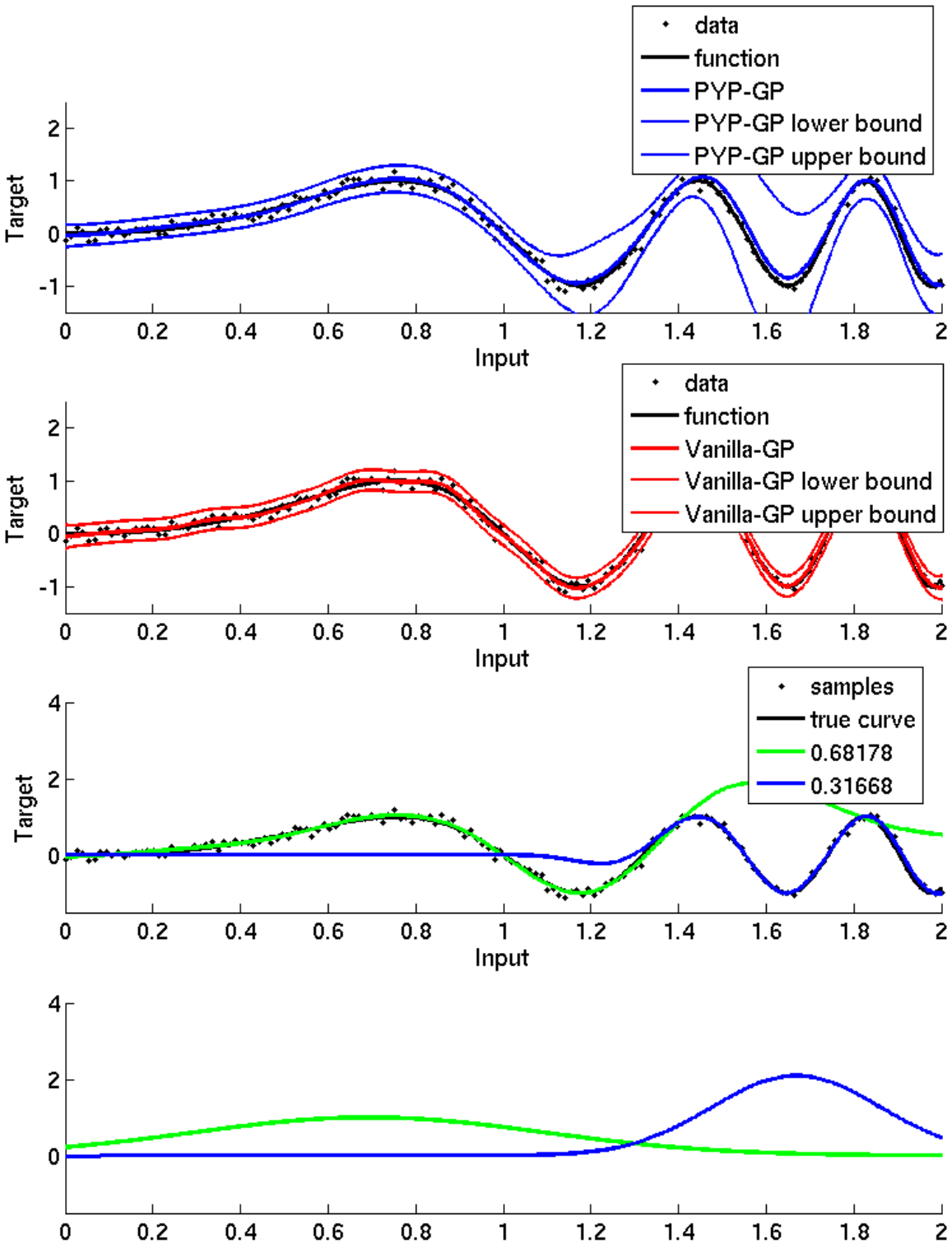


Figure 3: N=150,$\sigma_c^2 = 0.01$, $C = 20$,$f(x) = \sin(\pi x^{2.5})$, 100 iterations, with label re-ordering.

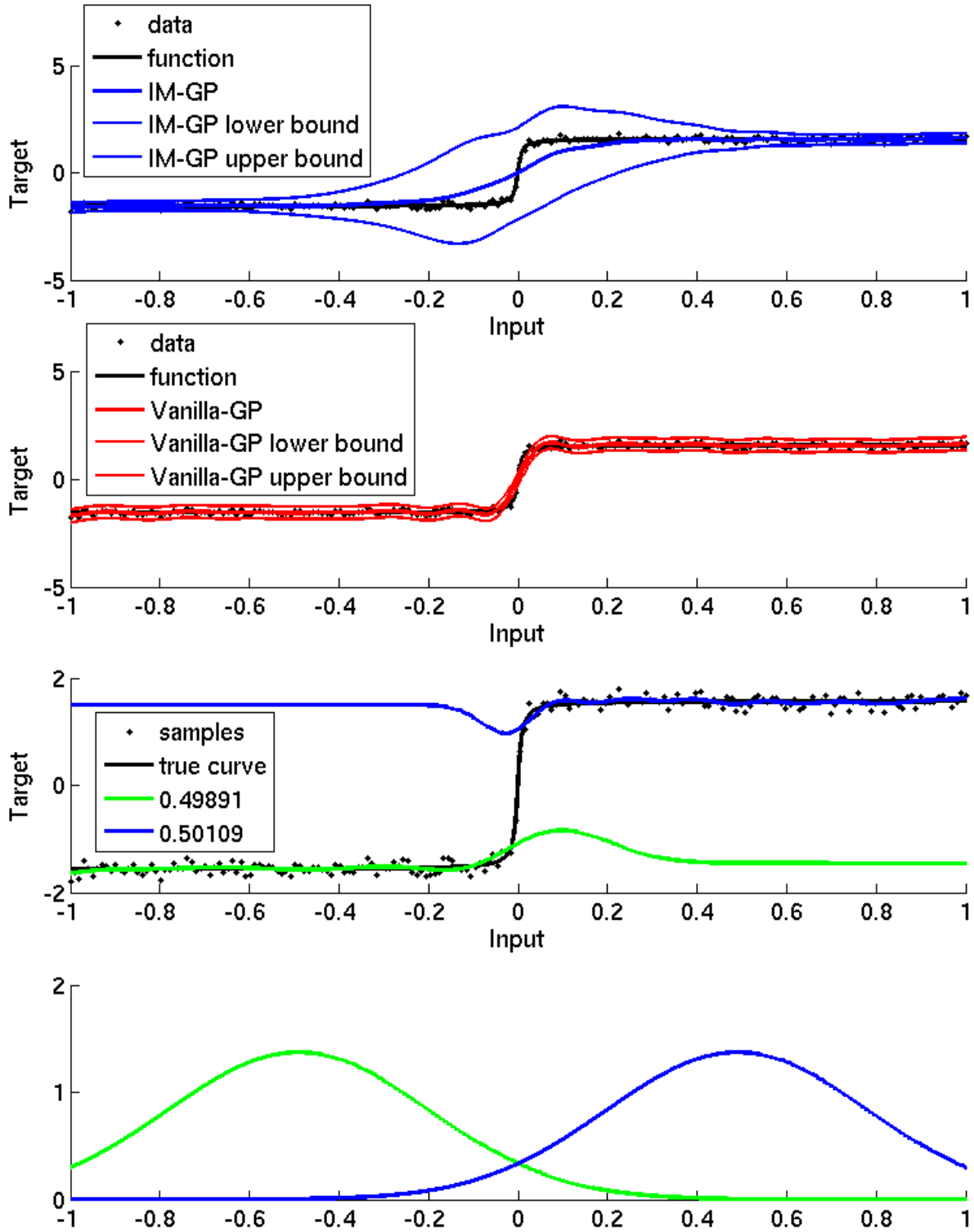


Figure 4: N=200,$\sigma_c^2 = 0.01$, $C = 2$,$f(x) = \text{atan}(150x)$, 100 iterations; $\text{RMS}_{\text{GP}} = 0.096$, $\text{RMS}_{\text{IMGP}} = 0.300$ ($\mathcal{L} = -73.3$).

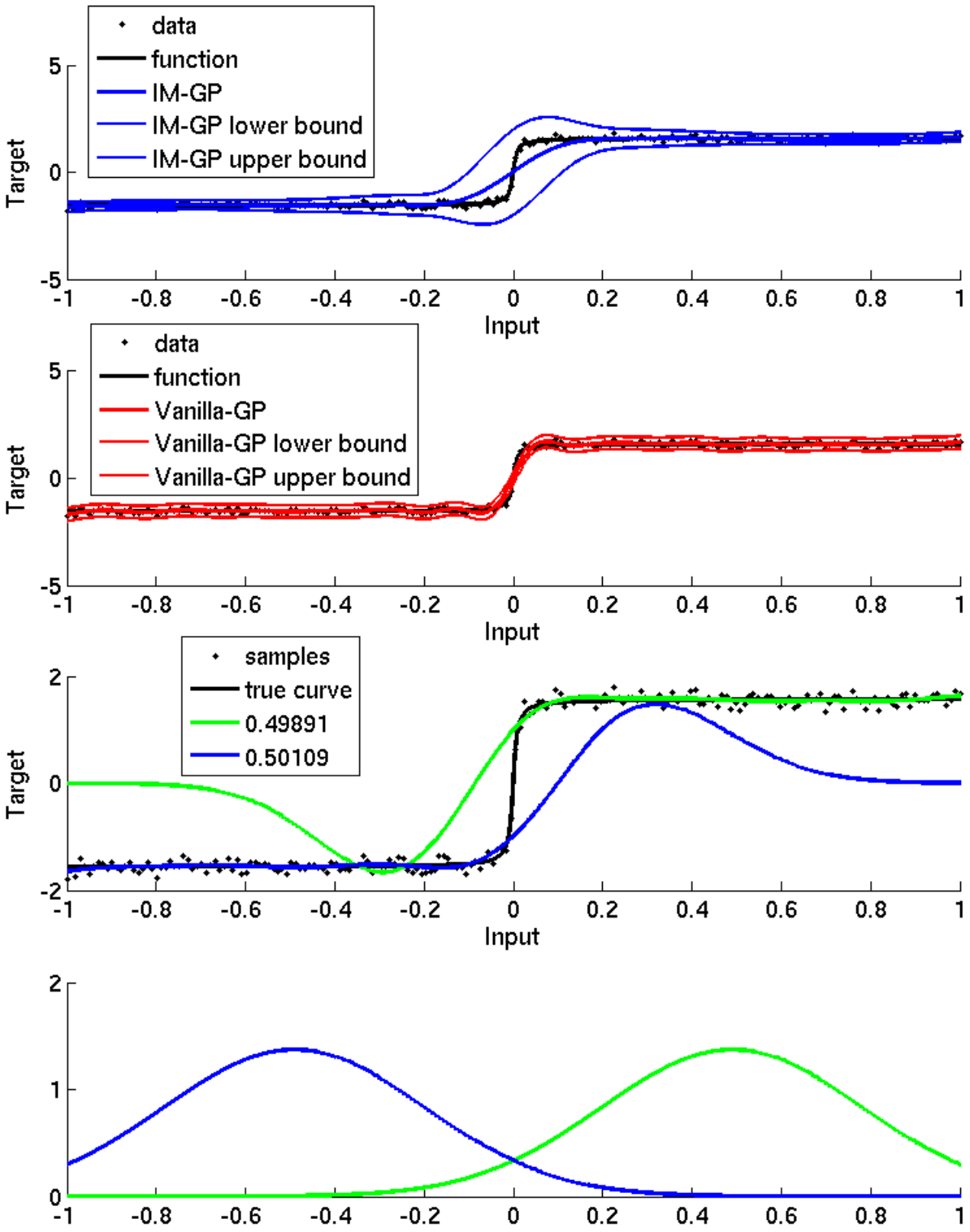


Figure 5: N=200,$\sigma_c^2 = 0.01$, $C = 2, f(x) = \text{atan}(150x)$, 100 iterations, zero-mean GPs; $\text{RMS}_{\text{GP}} = 0.096$, $\text{RMS}_{\text{IMGP}} = 0.229$ ($\mathcal{L} = -81.0$).

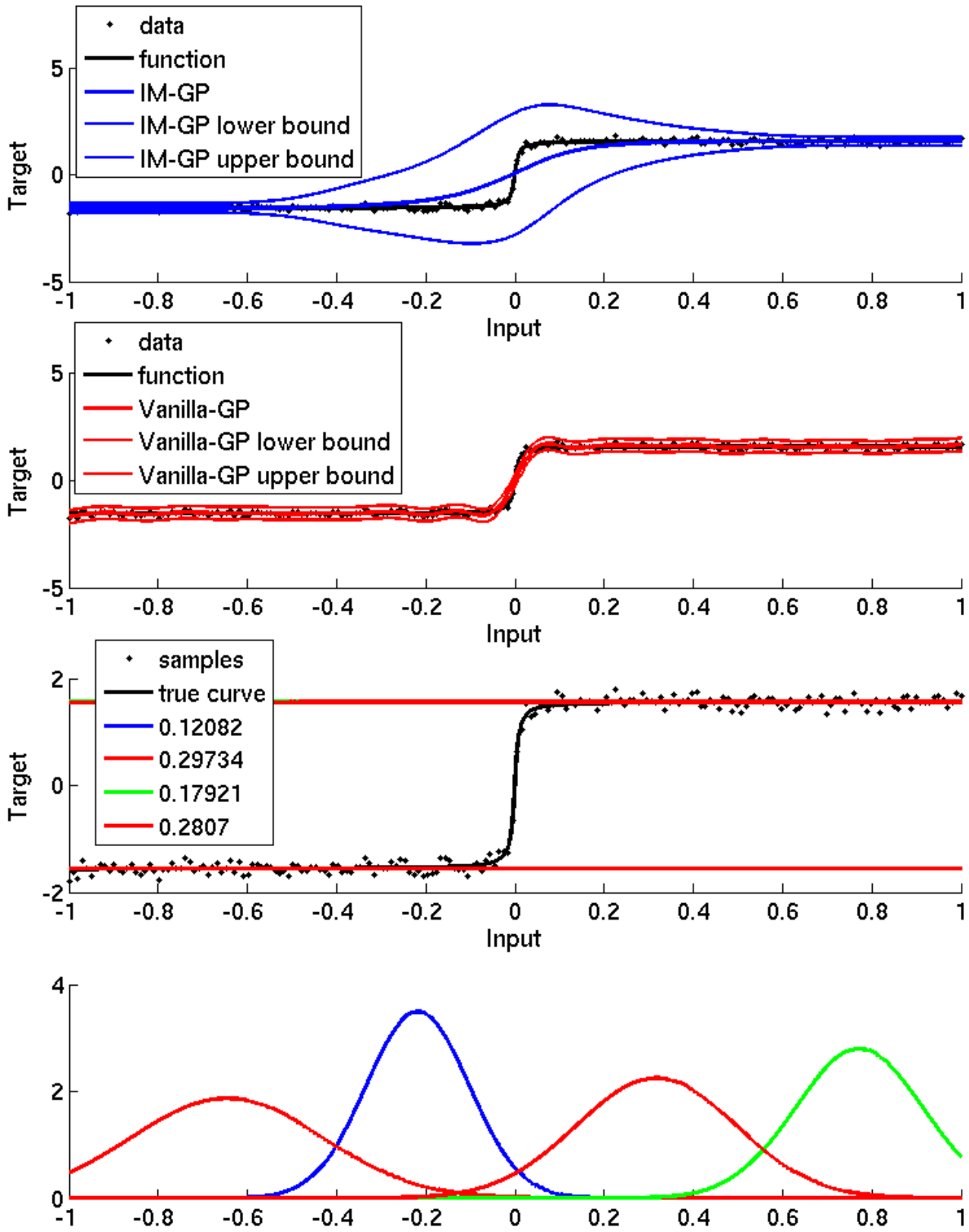


Figure 6: N=200,$\sigma_c^2 = 0.01$, $C = 20$,$f(x) = \text{atan}(150x)$, 100 iterations, without label re-ordering; $\text{RMS}_{\text{GP}} = 0.096$, $\text{RMS}_{\text{IMGP}} = 0.324$ ($\mathcal{L} = -82.1$).

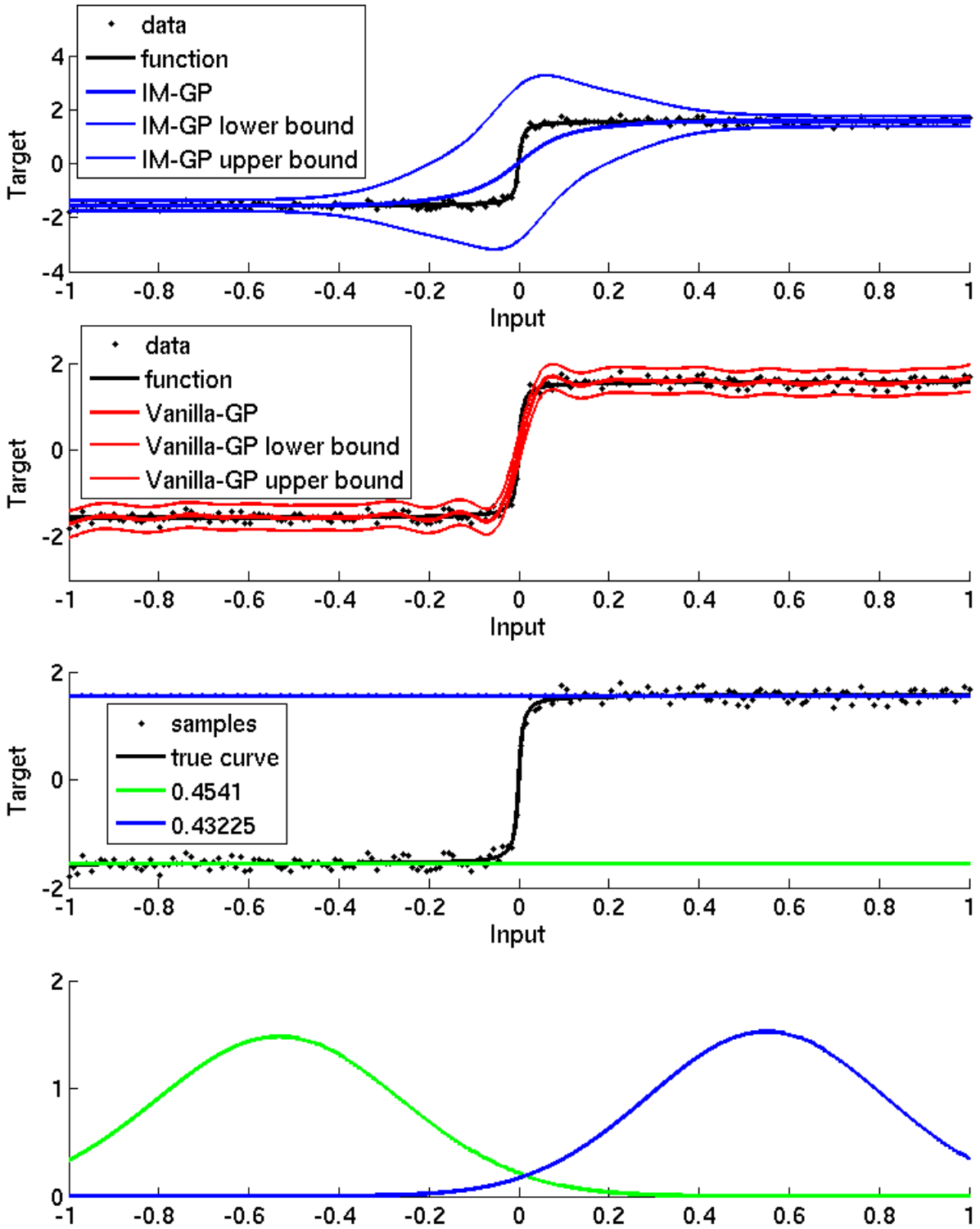


Figure 7: N=200,$\sigma_c^2 = 0.01$, $C = 20, f(x) = \text{atan}(150x)$, 100 iterations; $\text{RMS}_{\text{GP}} = 0.096$, $\text{RMS}_{\text{IMGP}} = 0.268$ ($\mathcal{L} = -39.3$).

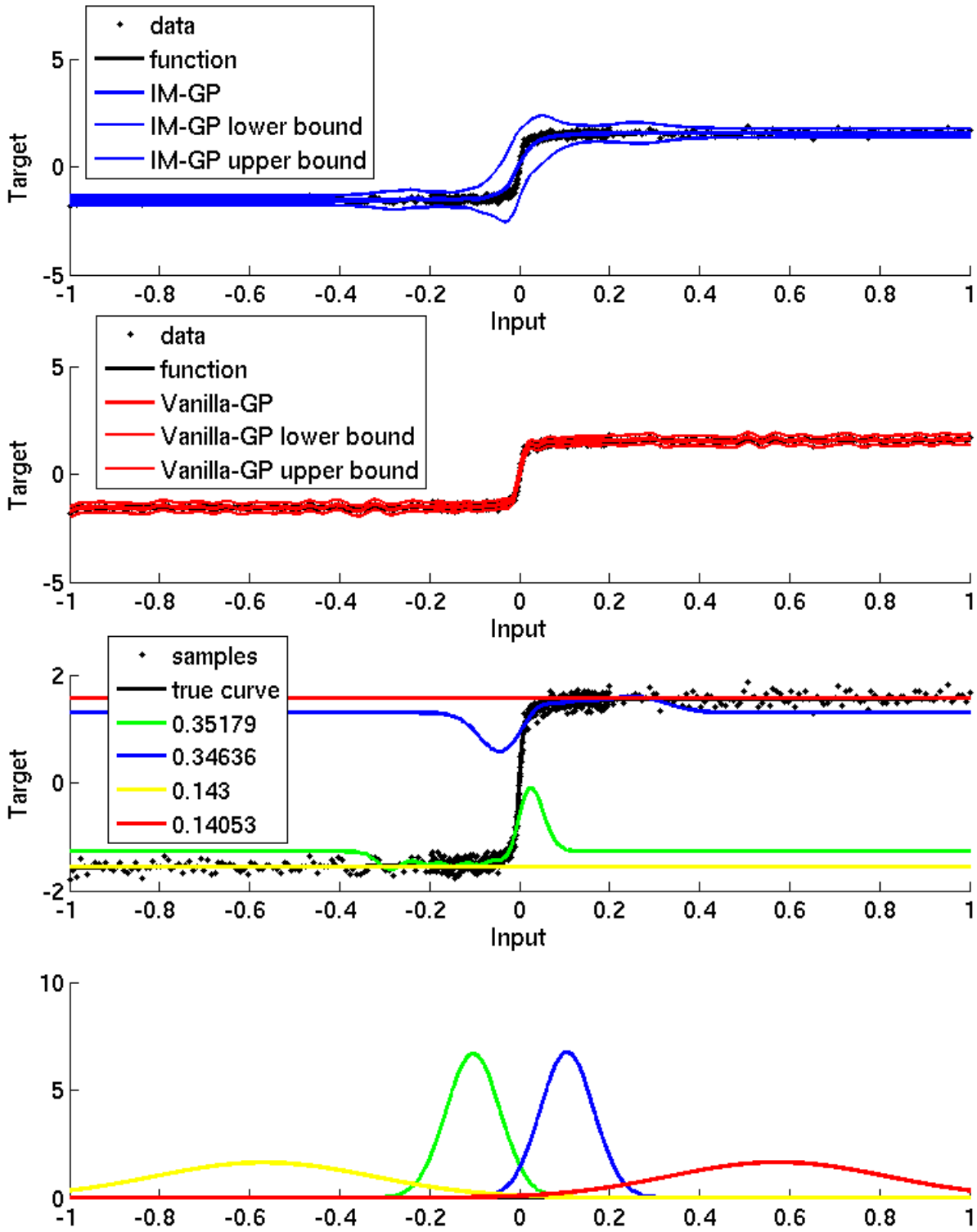


Figure 8: N=600 (500 in [-0.2:0.2]),$\sigma_c^2 = 0.01$, $C = 20$,$f(x) = \text{atan}(150x)$, 300 iterations; $\text{RMS}_{\text{GP}} = 0.057$, $\text{RMS}_{\text{IMGP}} = 0.096$ ($\mathcal{L} = 320.1$).

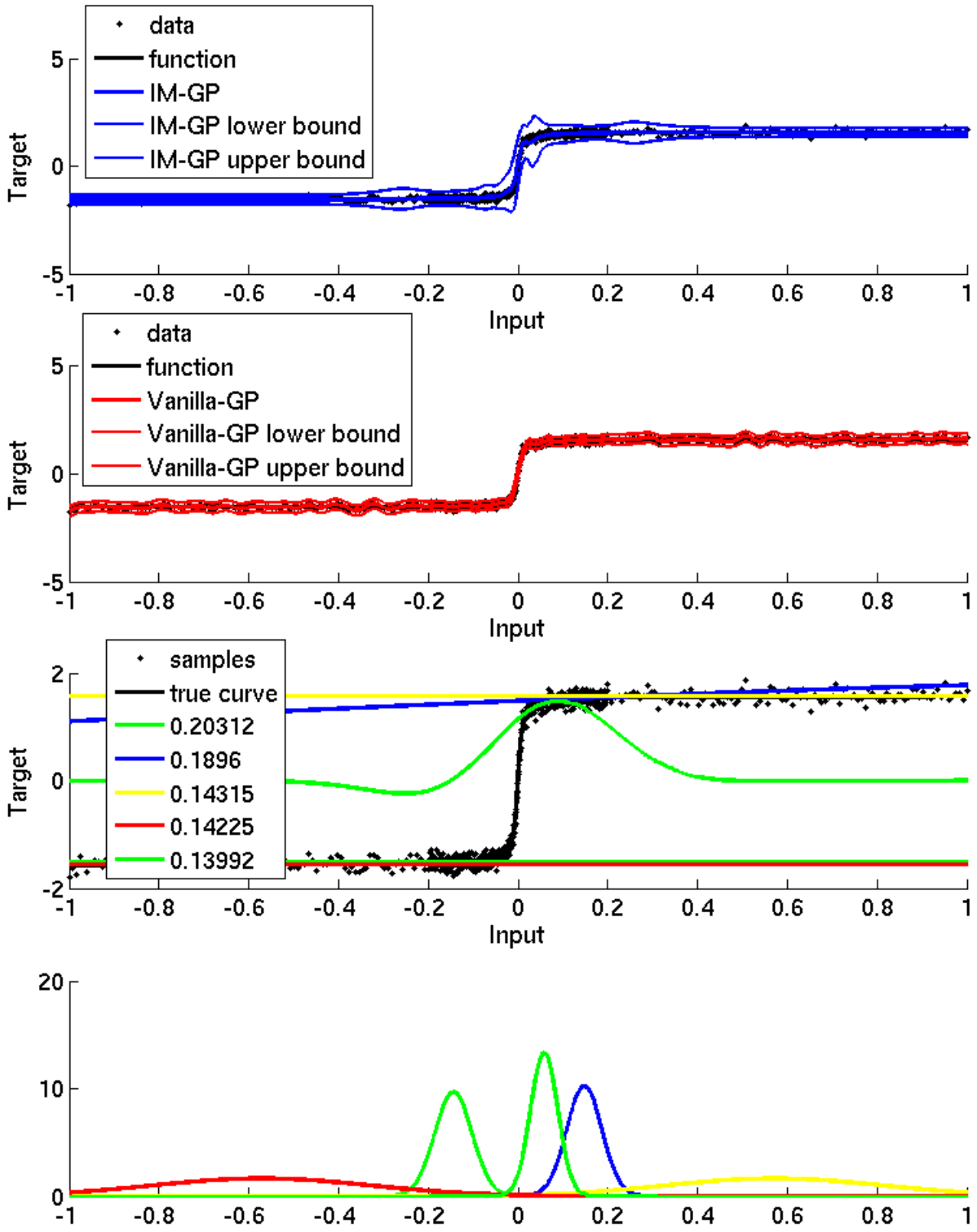


Figure 9: N=600 (500 in [-0.2:0.2]),$\sigma_c^2 = 0.01$, $C = 20$,$f(x) = \text{atan}(150x)$, 110 iterations, zero-mean IM-GP: $\text{RMS}_{\text{GP}} = 0.057$, $\text{RMS}_{\text{IMGP}} = 0.033$ ($\mathcal{L} = 343.4$).

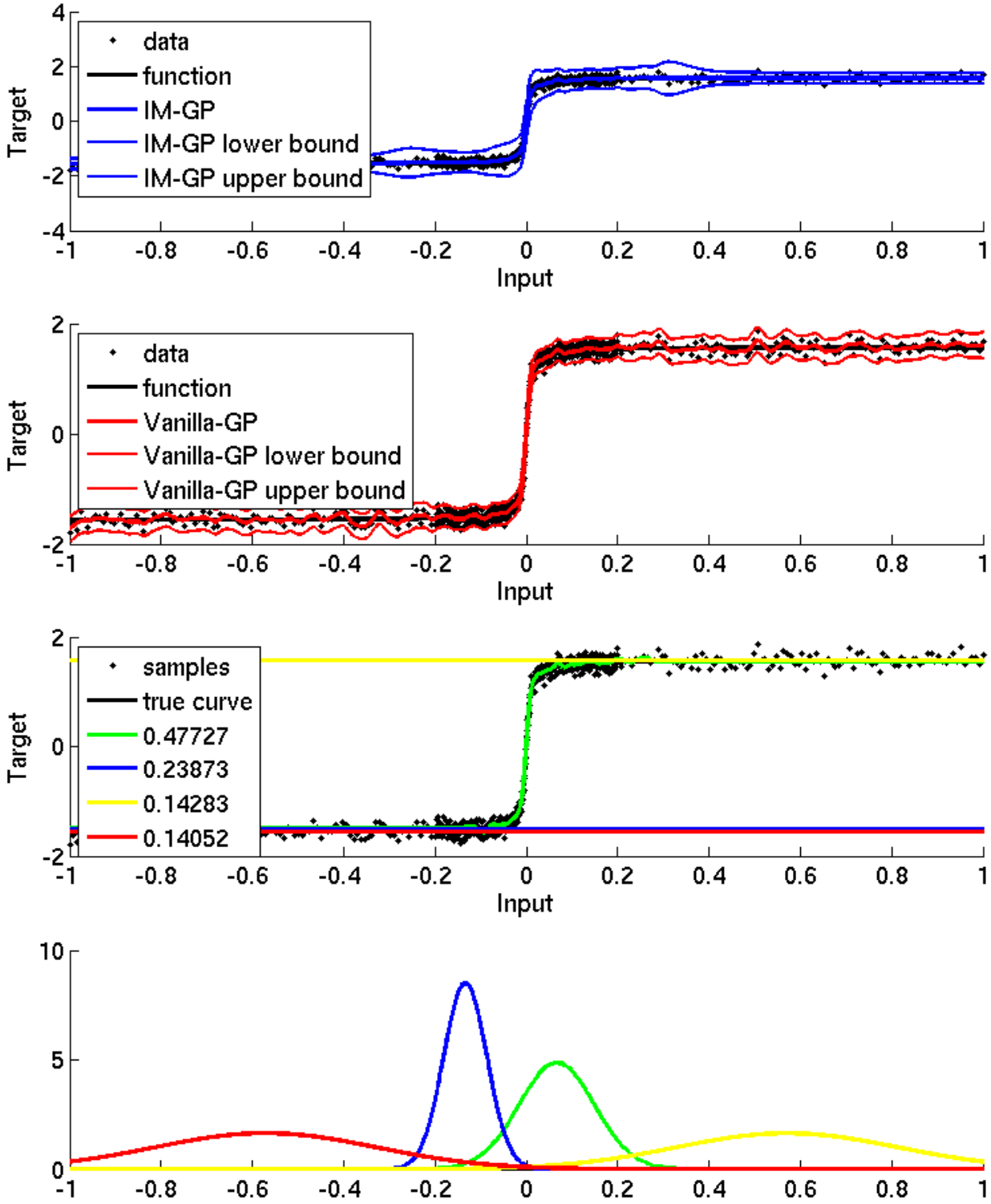


Figure 10: N=600 (500 in [-0.2:0.2]),$\sigma_c^2 = 0.01$, $C = 10$,$f(x) = \text{atan}(150x)$, 100 iterations, 5 kernels with ARD squared exp. covariance functions, the other 5 with neural network cov. functions: $\text{RMS}_{\text{GP}} = 0.048$, $\text{RMS}_{\text{IMGP}} = 0.015$ ($\mathcal{L} = 386.9$).

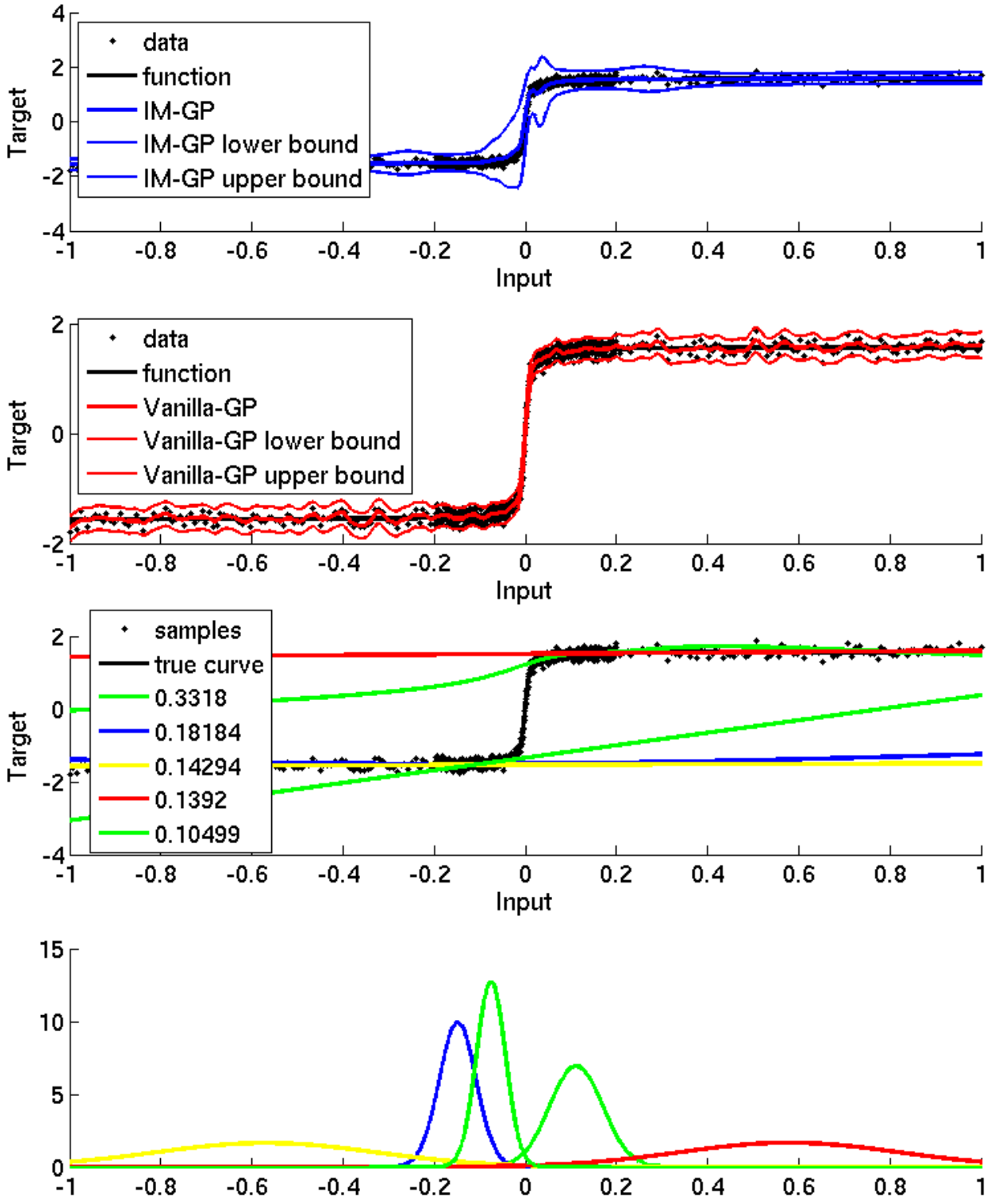


Figure 11: N=600 (500 in [-0.2:0.2]),$\sigma_c^2 = 0.01$, $C = 10$,$f(x) = \text{atan}(150x)$, 100 iterations, 5 kernels with ARD squared exp. covariance functions, the other 5 with neural network cov. functions, zero-mean IM-GP: $\text{RMS}_{\text{GP}} = 0.048$, $\text{RMS}_{\text{IMGP}} = 0.045$ ($\mathcal{L} = 348.8$).

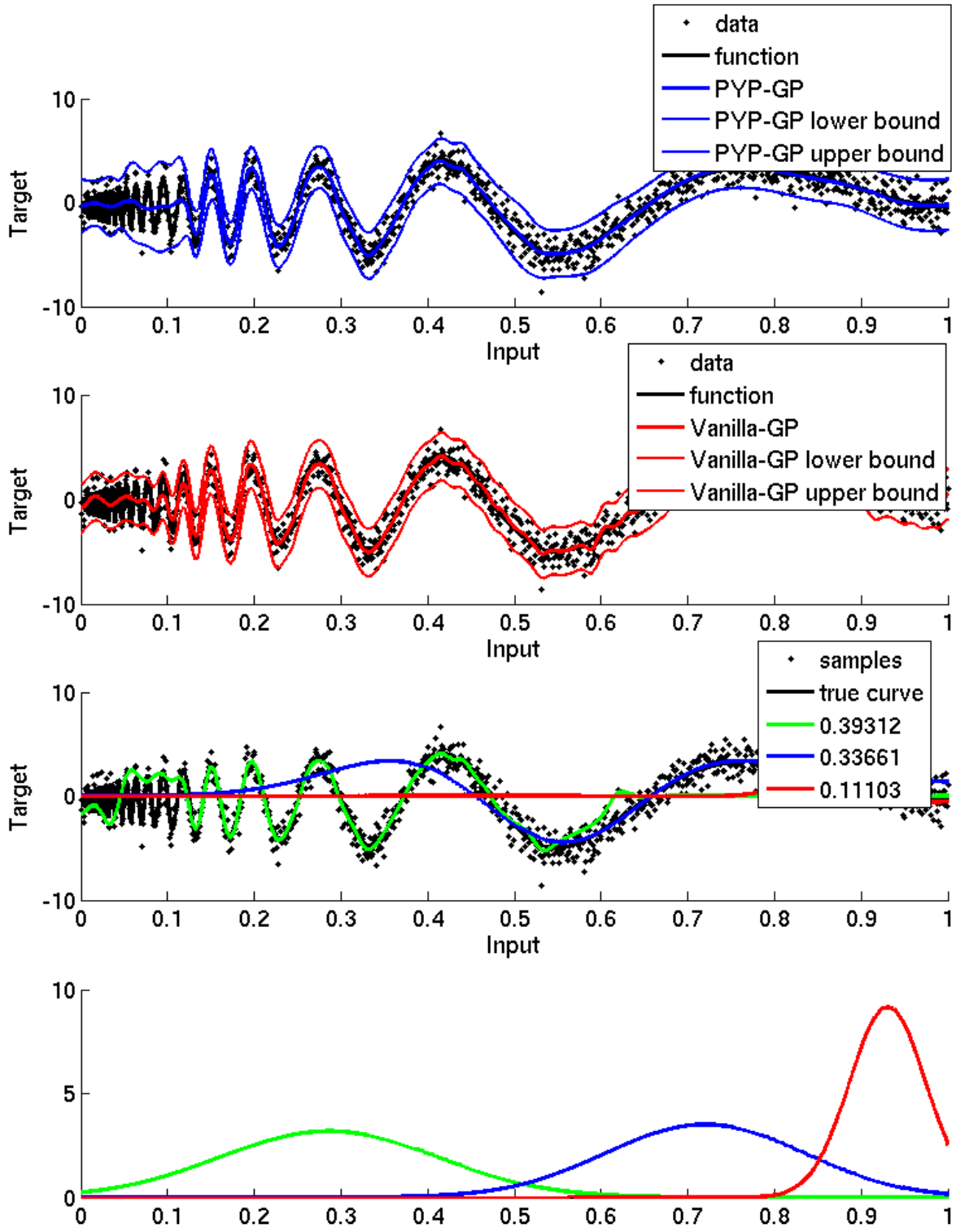


Figure 12: N=1024,$\sigma_c^2 = 1$, $C = 20$,$f(x) = (x(1-x))^{\frac{1}{2}}\sin(2\pi(1+a)/(x+a))$, $x \in [0:1]$, $a = 0.05$, 200 iterations; $\mathrm{RMS}_{\mathrm{GP}} = 0.513$, $\mathrm{RMS}_{\mathrm{IMGP}} = 0.568$ ($\mathcal{L} = -1767.6$).

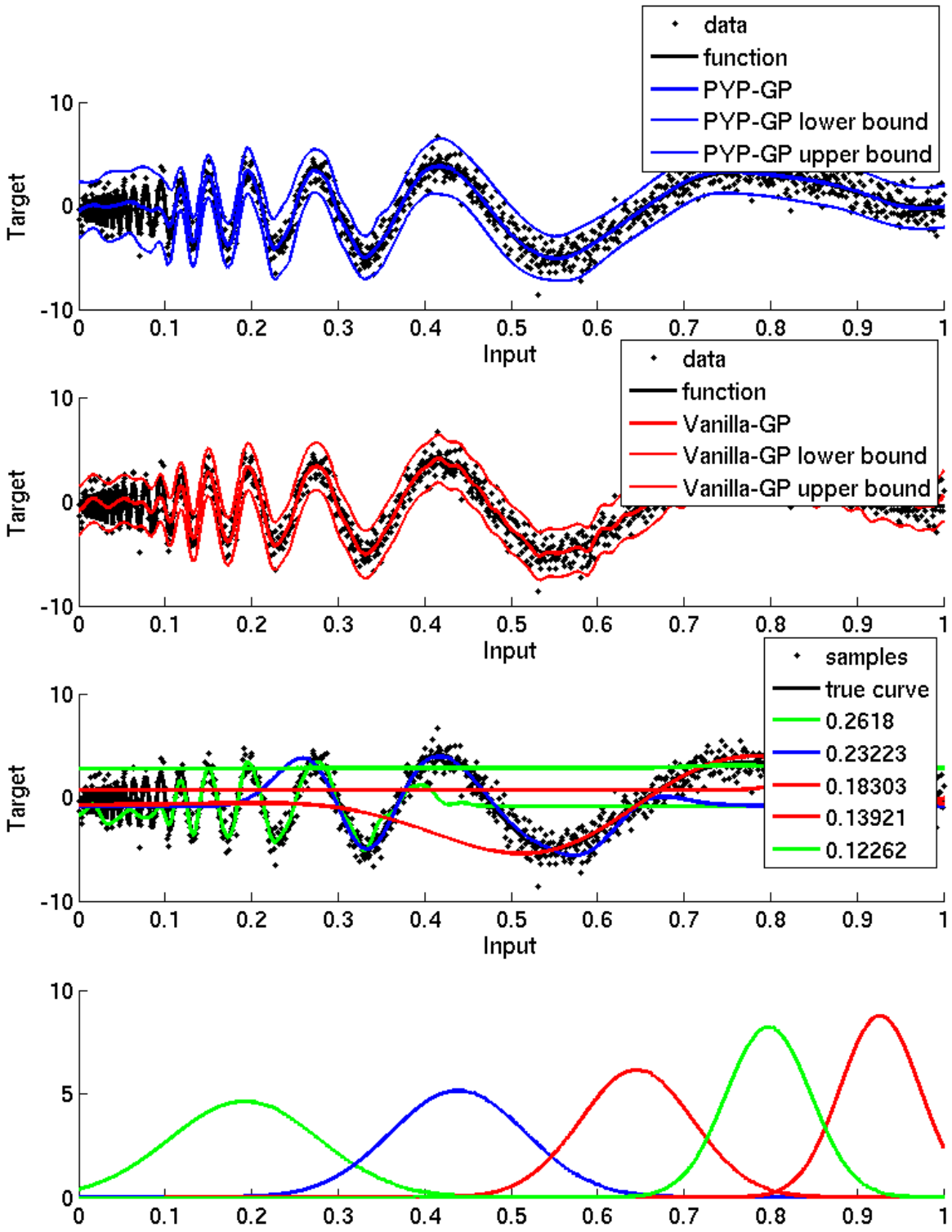


Figure 13: N=1024,$\sigma_c^2 = 1$, $C = 20$,$f(x) = (x(1-x))^{\frac{1}{2}}\sin(2\pi(1+a)/(x+a))$, $x \in [0:1]$, $a = 0.05$, 200 iterations; $\text{RMS}_{\text{GP}} = 0.513$, $\text{RMS}_{\text{IMGP}} = 0.506$ ($\mathcal{L} = -1731.8$).

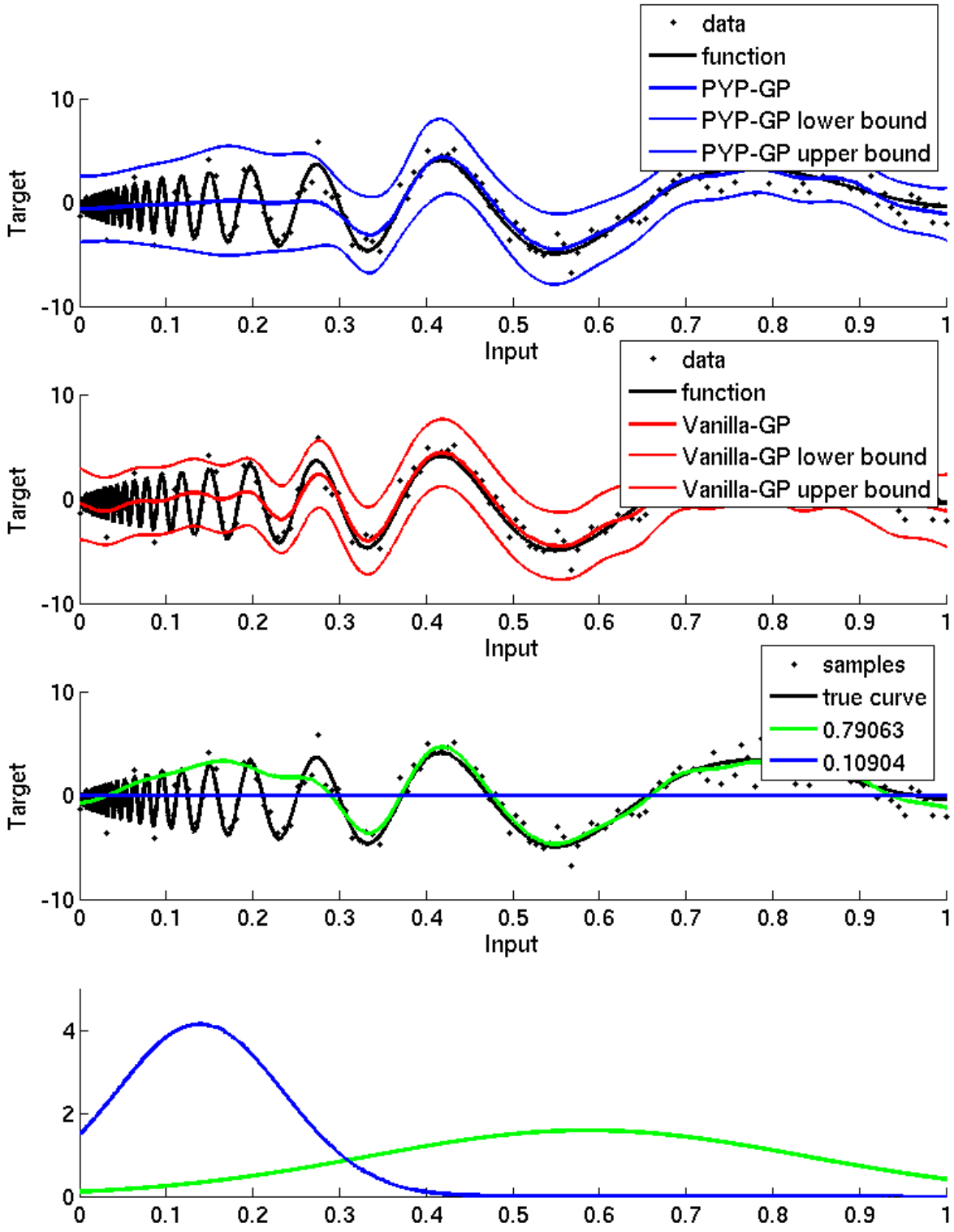


Figure 14: N=128,$\sigma_c^2 = 1$, $C = 20$,$f(x) = (x(1-x))^{\frac{1}{2}}\sin(2\pi(1+a)/(x+a))$, $x \in [0:1]$, $a = 0.05$, 100 iterations; $\text{RMS}_{\text{GP}} = 1.040$, $\text{RMS}_{\text{IMGP}} = 1.287$ ($\mathcal{L} = -291.4$).

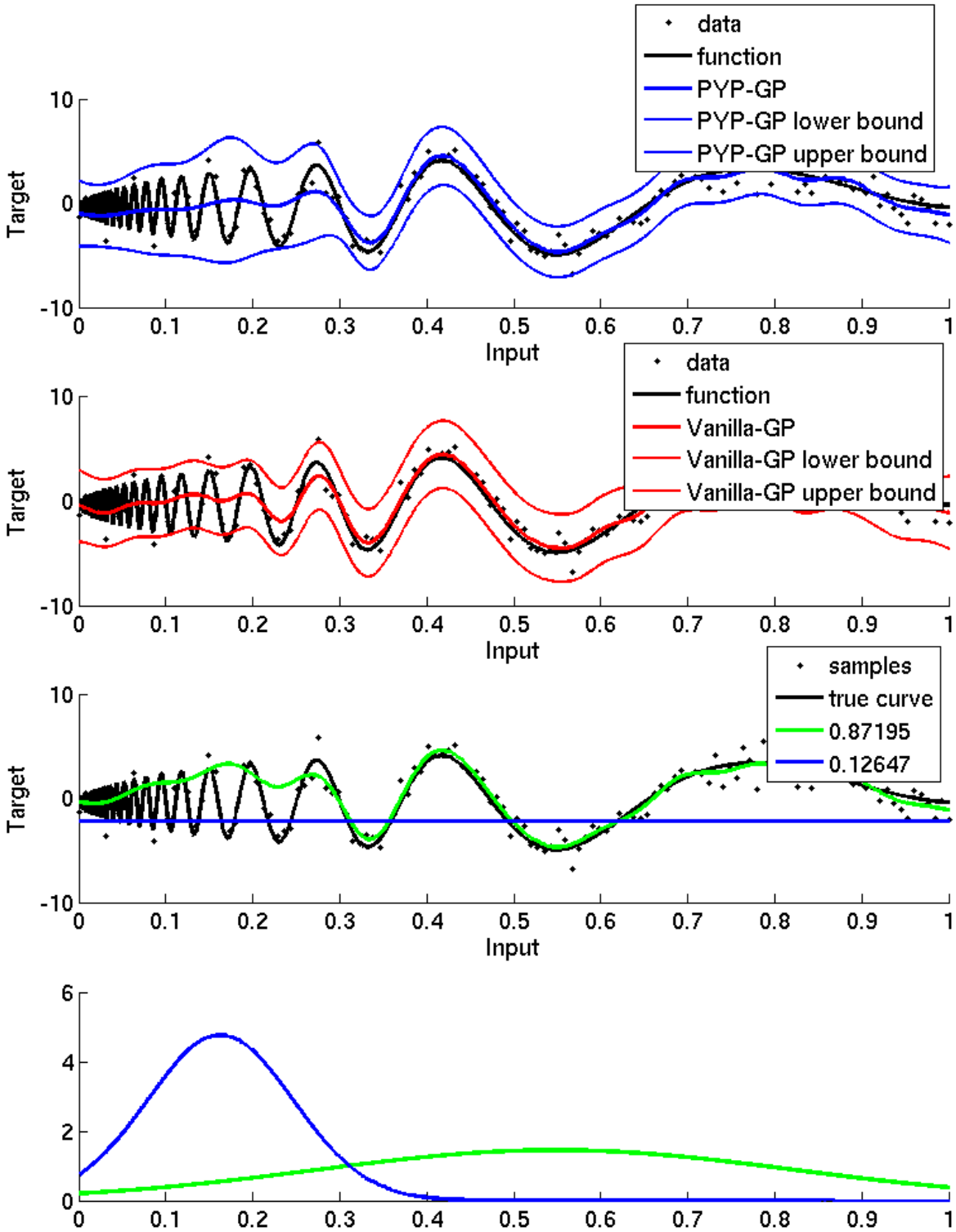


Figure 15: N=1024,$\sigma_c^2 = 1$, $C = 20$,$f(x) = (x(1-x))^{\frac{1}{2}}\sin(2\pi(1+a)/(x+a))$, $x \in [0:1]$, $a = 0.05$, 100 iterations; $\mathrm{RMS}_{\mathrm{GP}} = 1.040$, $\mathrm{RMS}_{\mathrm{IMGP}} = 1.200$ ($\mathcal{L} = -279.4$).