LISAL Code Manual

Sahil Garg, Amarjeet Singh {SAHILG, AMARJEET}@IIITD.AC.IN
Fabio Ramos FABIO.RAMOS@SYDNEY.EDU.AU

This document is part of code release on algorithm LISAL that is proposed in (Garg, Singh, & Ramos, 2014). While this document has been written for a broader level understanding of functionality in the code, it is expected that the reader has understood the algorithmic logic in details in the paper (Garg et al., 2014) pre-hand. In addition, the code itself has been made self explanatory with incorporation of inline comments in the code files. Here is the list of functions implemented as part of release:

- **runMain:** This is a script file that is added as an example for code users on how to run main() function in the code.
- **main():** This function performs the functionalities in sequence: 1) retrieves a dataset $\{\boldsymbol{X} \in \mathbb{R}^{n\times p}, \boldsymbol{y} \in \mathbb{R}^n\}$ as per a given data source name; 2) selects a subset of data samples $\{\boldsymbol{X_V} \in \mathbb{R}^{n_v\times p}, \boldsymbol{y_V} \in \mathbb{R}^n\}$; 3) initializes the hyper-parameters of GP model (stationary or NGP) using the statistical properties of training data; 4) learns the hyper-parameters of the GP model using the initialized the hyper-parameter values as seeding instance.; 5) tests the learned model on whole data set. Also see code documentation for more details.
- **getData():** This function retrieves dataset as per specification of data source name. Available data sources are: ozone, wind, image, rain, sp500, juraCu. See more code documentation for more details.
- **initGPCov():** This function initializes the hyper-parameters of GP model as per given covariance structure specification. For a stationary model, hyper-parameters of GPy are initialized. For an NGP model, hyper-parameters of GPs (a stationary GP), GPy (observation GP), GPz (latent GP) are initialized. See code documentation for more details.
- **lrnGPCov()**: This function learns the hyper-parameters of a GP model on the training data $\{\boldsymbol{X_V}, \boldsymbol{y_V}\}$ by maximizing log marginal likelihood of the observed real dynamics. For an NGP, the proposed algorithm LISAL is used for adaptively learning the hyper-parameters of GPy, GPz, and for exploring the latent parameter values $\boldsymbol{z_M} \in \mathbb{R}^m$ across latent locations $\boldsymbol{X_M} \in \mathbb{R}^{m\times p}$ that are selected with information gain (entropy or mutual information). See Algorithm 1 that briefs on logic of LISAL for the case of informative sensing of latent space with entropy gain. For detailed understanding on LISAL, it is advised to refer to the original paper (Garg et al., 2014). See code documentation for details on implementation and use of code.
- **evalLgMrgLkl()**: This function evaluates log marginal likelihood for a GP model. For an NGP, lml is evaluated as per the expression below:

$$-\frac{1}{2}\boldsymbol{y_V}^T\left[\mathcal{K}^y_{NS}(\boldsymbol{X_V},\boldsymbol{X_V}|\boldsymbol{\mu^z_V},\boldsymbol{\Theta_y})\right]^{-1}\boldsymbol{y_V} - \log\left|\mathcal{K}^y_{NS}(\boldsymbol{X_V},\boldsymbol{X_V}|\boldsymbol{\mu^z_V},\boldsymbol{\Theta_y})\right| - \log|\boldsymbol{\Sigma}^z_V| - n_v\log 2\pi, \tag{1}$$

Algorithm 1 LISAL

Input: $\{\boldsymbol{X_V} \in \mathbb{R}^{n_v \times p}, \boldsymbol{y_V} \in \mathbb{R}^{n_v}\}, m_1, m_2, c$
Output: $\boldsymbol{\Theta_y^*}, \boldsymbol{\Theta_z^*}, \boldsymbol{X_M^*} \in \mathbb{R}^{m \times p}, \boldsymbol{z_M^*} \in \mathbb{R}^m$
$\boldsymbol{\Theta_{y_s}^*} = argmax_{\boldsymbol{\Theta_{y_s}}} \mathcal{P}(\boldsymbol{y_V}|\boldsymbol{\Theta_{y_s}})$
$\boldsymbol{X_{M_1}^*} = argmax_{\boldsymbol{X_{M_1}}:\boldsymbol{X_{M_1}} \subset \boldsymbol{X_V}} \mathcal{H}(\boldsymbol{\mathcal{Y}_{M_1}}|\boldsymbol{\Theta_{y_s}^*})$
$\{\boldsymbol{\Theta_y^*}, \boldsymbol{\Theta_z^*}, \boldsymbol{z_{M_1^*}^*}\} = argmax_{\boldsymbol{\Theta_y}, \boldsymbol{\Theta_z}, \boldsymbol{z_{M_1^*}}} \log \mathcal{P}(\boldsymbol{y_V}|\boldsymbol{\Theta_y}, \boldsymbol{\Theta_z}, \boldsymbol{z_{M_1^*}}, \boldsymbol{X_{M_1}^*})$
for $i = 1 \rightarrow c$ **do**
    $\boldsymbol{X_{M_{i+1}}^*} = argmax_{\boldsymbol{X_{M_{i+1}}}:\boldsymbol{X_{M_{i+1}}} \subset \boldsymbol{X_{V \setminus M_{1 \cdots i}}}} \mathcal{H}(\boldsymbol{\mathcal{Z}_{M_{i+1}}}|\boldsymbol{\Theta_z^*}, \boldsymbol{X_{M_{1 \cdots i}}^*})$
    $\boldsymbol{\Theta_y} \leftarrow \boldsymbol{\Theta_y^*}, \boldsymbol{\Theta_z} \leftarrow \boldsymbol{\Theta_z^*}$
    $\{\boldsymbol{\Theta_y^*}, \boldsymbol{\Theta_z^*}, \boldsymbol{z_{M_{i+1}^*}^*}\} = argmax_{\boldsymbol{\Theta_y}, \boldsymbol{\Theta_z}, \boldsymbol{z_{M_{i+1}^*}}} \log \mathcal{P}(\boldsymbol{y_V}|\boldsymbol{\Theta_y}, \boldsymbol{\Theta_z}, \boldsymbol{z_{M_{i+1}^*}}, \boldsymbol{X_{M_{i+1}}^*}, \{\boldsymbol{X_{M_{1 \cdots i}}^*}, \boldsymbol{z_{M_{1 \cdots i}^*}^*}\})$
end for
$\boldsymbol{X_M^*} \leftarrow \boldsymbol{X_{M_{1 \cdots c+1}}^*}, \boldsymbol{z_{M^*}^*} \leftarrow \boldsymbol{z_{M_{1 \cdots c+1}^*}^*}$

wherein $\boldsymbol{\mu_V^z}$, $\boldsymbol{\Sigma_V^z}$ are the latent dynamics predictive mean and predictive covariance inferred across $\boldsymbol{X_V}$ by conditioning upon latent parameters $\boldsymbol{z_M}$ across $\boldsymbol{X_M}$. See code documentation for details on how to call the function with appropriate parametrization.

- **evalCov():** This function evaluates covariance for a stationary GP model or a non-stationary GP model as per hyper-parameters of GPy, GPz and the latent parameters $\boldsymbol{z_M}$ across $\boldsymbol{X_M}$. This function can evaluate non-stationary covariance for models: Heteroscedastic Gaussian Process (HGP), Gaussian Process Product Model (GPPM), Process Convolution with Local Smoothing Kernels (PCLSK), Bayesian Warped Gaussian Process (BWGP), Spatial Deformation of Input Space (SDIS), Latent Extension of Input Space (LEIS). See code documentation for implementation details and input output specification.

References

Garg, S., Singh, A., & Ramos, F. (2014). Most likely bayesian nonstationary gaussian process models: Efficient algorithms and empirical studies. *JAIR*.