Gaussian-Process-Model-based System-Identification Toolbox for Matlab

Version 1.2.2

Martin Stepančič and Juš Kocijan

October 30, 2015

1 Introduction

The idea of this toolbox is to facilitate dynamic systems identification with Gaussian-process (GP) models. The presented toolbox is continuously developing and is put together with hope to be useful as a springboard for the modelling of dynamic systems with GP models.

The GP model belongs to the class of black-box models. GP modelling differs from most other black-box identification approaches in that it does not try to approximate the modelled system by fitting the parameters of the selected basis functions, but rather it searches for the relationship among the measured data. The model is composed of input-output data that describes the behaviour of the modelled system and the covariance function that describes the relation with respect to the input-output data. The prediction of the GP model output is given as a normal distribution, expressed in terms of the mean and the variance. The mean value represents the most likely output, and the variance can be interpreted as a measure of its confidence.

System identification is composed of methods to build mathematical models of dynamic systems from measured data. It is one of the scientific pillars used for dynamic-systems analysis and control design. The identification of a dynamic system means that we are looking for a relationship between past observations and future outputs. Identification can be interpreted as the concatenation of a mapping from measured data to a regression vector, followed by a nonlinear mapping from the regression vector to the output space. Various machine-learning methods and statistical methods are employed to determine the nonlinear mapping from the regression vector to the output space. One of the possible methods for a description of the nonlinear mapping used in identification is GP models. It is straightforward to employ GP models for the discrete-time modelling of dynamic systems within the prediction-error framework.

Many dynamic systems are often considered as complex; however, simplified input-output behaviour representations are sufficient for certain purposes, e.g., feedback control design, prediction models for supervisory control, etc.

More on the topic of system identification with GP models and the use of this models for control design can be found in the book:
Juš Kocijan (2016) Modelling and Control of Dynamic Systems Using Gaussian Process Models, Springer.

2 GP-Model-based System-Identification Toolbox for Matlab

2.1 Prerequisites

As this toolbox is intended to use within Matlab environment the user should have Matlab installed. It works on Matlab 7 and later, but there should be no problems using the toolbox on previous versions of Matlab, e.g., 6 or 5.

It is also assumed that the GPML toolbox[1], general purpose GP modelling toolbox for Matlab, is installed. The GP-model-based system-identification toolbox serves as upgrade to GPML toolbox.

The user should posses some familiarity with the Matlab structure and programming.

2.2 Installing GPdyn toolbox

Unzip the file GPdyn into chosen directory and add path, with subdirectories, to Matlab path.

2.3 Overview of the GPdyn toolbox

GPdyn files are contained in several directories, depending on their purpose:

training functions, used for training GP models of dynamic systems;

GP-model evaluation functions, used for simulating the dynamic GP model;

LMGP-model evaluation functions, which are used when modelling and simulating the system with a GP model with incorporated local models (LMGP model);

utilities functions, that are various support functions;

demo functions, which demonstrate the use of the toolbox for identification of dynamic systems.

[1]It can be obtained from *http://www.gaussianprocess.org/gpml*.

The list of included functions, demos and one model is given in following tables.

GP-model training functions:

trainGParx	GP-model training of ARX model
`trainGPoe`	GP-model training of OE model
`gp_initial`	- finding initial values of hyperparameters with random search
`minimizeDE`	minimize a multivariate function using differential evolution

Covariance functions:

Included and explained in enclosed GPML toolbox

GP-model evaluation:

`simulGPnaive`	GP model simulation without the propagation of uncertainty
`simulGPmcmc`	GP model simulation with Monte Carlo approximation
`simulGPtaylorSE`	GP model simulation with analytical approximation of statistical moments with a Taylor expansion for the squared exponential covariance function
`simulGPexactSE`	GP model simulation with exact matching of statistical moments for the squared exponential covariance function
`simulGPexactLIN`	GP model simulation with exact matching of statistical moments for the linear covariance function
`predGPnaive`	multi-step-ahead prediction of GP model without the propagation of uncertainty
`gpx`	modified version of GP rutine from the GPML toolbox
`gmx_sample`	creates samples of mixture components
`gpTaylorSEard`	GP model prediction with stochastic inputs for the squared exponential covariance function with Taylor expansion
`gpExactLINard`	GP model prediction with stochastic inputs for the linear covariance function
`gpExactSEard`	GP model prediction with stochastic inputs for the squared exponential covariance function

LMGP-model evaluation:

simulLMGPnaive	LMGP model simulation without the propagation of uncertainty
simulLMGPmcmc	LMGP model simulation with Monte Carlo approximation
trainLMGP	LMGP model training
gpSD00	- LMGP model prediction - data likelihood and its derivatives

Supporting functions:

add_noise_to_vector	adding white noise to noise-free simulation results
construct	construction of the input regressors from system's input signals
eval_func	method to evaluate covariance, mean and likelihood functions
likelihood	calculates negative log marginal likelihood
lipschitz	the method for the lag-space selection, based on Lipschitz quotients
validate	checking of the parameters match
loss	performance measures
mcmc_test_pdfs	testing sampled probability distributions
plotgp	plot results (output and error) of the GP model prediction
plotgpe	plot error of the GP model prediction
plotgpy	plot output of the GP model prediction
preNorm	preprocessing of data
postNorm	postprocessing of data
postNormVar	postprocessing of predicted variance
sig_prbs	generating pseudo-random binary signal
sig_prs_minmax	generating pseudo-random signal

Demos:

`demo_example_present`	present the system used in demos
`demo_example_gp_data`	generate data for the identification and validation of the GP model
`demo_example_gp_norm`	normalization of input and output data
`demo_example_gp_training`	training of the GP model
`demo_example_gp_simulation`	validation with simulation of the GP model
`demo_example_lmgp_data`	generate data for the identification and validation of the LMGP model
`demo_example_lmgp_training`	training of the LMGP model
`demo_example_lmgp_simulation`	simulation of the LMGP model
`demo_example`	system simulation
`demo_example_derivative`	obtaining system's derivatives
`demo_example_LM_ident`	identification of system's local models

2.4 How to use this toolbox

2.4.1 Demos

A simple nonlinear dynamic system is used to demonstrate the identification and simulation of the GP models:

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k) \tag{1}$$

The system was used as an example of dynamic system identification with artificial neural networks in:
K.S. Narendra and K. Parthasarathy. Identification and Control of Dynamical Systems Using Neural Networks, IEEE Transactions on Neural Networks, Vol.1 No. 1, 4–27, 1990.

demo_example_present, presents this system.

Following three demos present the identification of dynamic systems with the GP model:

demo_example_gp_data, which presents how to obtain and assemble data for identification;

demo_example_gp_norm, which shows how to normalise input and output data for training;

demo_example_gp_training, which demonstrates the identification with a GP model;

demo_example_gp_simulation, which shows how to simulate the GP model.

The use of the GP model with incorporated local models is presented with demos:

demo_example_lmgp_data, which presents how to obtain and assemble data for identification;

demo_example_lmgp_training, which demonstrates the training (=identifying) the LMGP model;

demo_example_lmgp_simulation, which shows how to simulate the LMGP model.

2.4.2 Acknowledgements

We would like to thank all past, present and future contributors to this toolbox.