Deep and Linked Gaussian Process Emulations

Release 2.4.0

Deyu Ming

Mar 05, 2024

CONTENTS:

1 API Reference **1**

1.1 dgp module . . . 1
1.2 emulation module . . . 1
1.3 gp module . . . 1
1.4 imputation module . . . 1
1.5 kernel_class module . . . 2
1.6 likelihood_class module . . . 8
1.7 linkgp module . . . 8
1.8 utils module . . . 8
1.9 functions module . . . 8
1.10 synthetic module . . . 10

2 Indices and tables **11**

Python Module Index **13**

Index **15**

CHAPTER

ONE

API REFERENCE

1.1 dgp module

1.2 emulation module

1.3 gp module

1.4 imputation module

class dgpsi.imputation.**imputer**(*all_layer*, *block=True*)

Bases: `object`

Class to implement imputation of latent variables.

Parameters

- **all_layer** (*list*) – a list that contains the DGP model
- **block** (*bool, optional*) – whether to use the blocked (layer-wise) ESS for the imputations. Defaults to *True*.

key_stats()

Compute and store key statistics used in predictions

static one_sample(*target_kernel*, *linked_upper_kernels*, *k*)

Impute one latent variable produced by a particular GP.

Parameters

- **target_kernel** (*class*) – the GP whose output is a latent variable that needs to be imputed.
- **linked_upper_kernels** (*list*) – a list of GPs (in the next layer) that link the output produced by the GP defined by the argument **target_kernel**.
- **k** (*int*) – the index indicating the position of the GP defined by the argument **target_kernel** in its layer.

static one_sample_block(*target_layer*, *upper_layer*)

Impute a latent layer.

Parameters

- **target_layer** (*list*) – a list of GPs that produce a latent layer that needs to be imputed.

- **upper_layer** (*list*) – a list of GPs (in the next layer) that are fed by the output of GPs in **target_layer**.

sample(*burnin=0*)

Implement the imputation via the ESS-within-Gibbs.

Parameters

burnin (*int, optional*) – the number of burnin iterations for the ESS-within-Gibbs sampler to generate one realisation of latent variables. Defaults to *0*.

1.5 kernel_class module

dgpsi.kernel_class.**combine**(**layers*)

Combine layers into one list as a DGP or linked (D)GP structure.

Parameters

layers (*list*) – a sequence of lists, each of which contains the GP nodes (defined by the *kernel* class), likelihood nodes (e.g., defined by the `Poisson` class), or containers (defined by the `container` class) in that layer.

Returns

a list of layers defining the DGP or linked (D)GP structure.

Return type

list

class dgpsi.kernel_class.**kernel**(*length*, *scale=1.0*, *nugget=1e-06*, *name='sexp'*, *prior_name='ga'*, *prior_coef=None*, *bds=None*, *nugget_est=False*, *scale_est=False*, *input_dim=None*, *connect=None*)

Bases: `object`

Class that defines the GPs in the DGP hierarchy.

Parameters

- **length** (*ndarray*) – a numpy 1d-array, whose length equals to:
  1. either one if the lengthscales in the kernel function are assumed same across input dimensions; or
  2. the total number of input dimensions, which is the sum of the number of feeding GPs in the last layer (defined by the argument **input_dim**) and the number of connected global input dimensions (defined by the argument **connect**), if the lengthscales in the kernel function are assumed different across input dimensions.
- **scale** (*float, optional*) – the variance of a GP. Defaults to *1*.
- **nugget** (*float, optional*) – the nugget term of a GP. Defaults to *1e-6*.
- **name** (*str, optional*) – kernel function to be used. Either *sexp* for squared exponential kernel or *matern2.5* for Matern2.5 kernel. Defaults to *sexp*.
- **prior_name** (*str, optional*) – prior options for the lengthscales and nugget term. Either gamma (*ga*), inverse gamma (*inv_ga*) or the reference prior (*ref*) for the lengthscales and nugget term. Set *None* to disable the prior. Defaults to *ga*.
- **prior_coef** (*ndarray, optional*) – if **prior_name** is either *ga* or *inv_ga*, it is a numpy 1d-array that contains two values specifying the shape and rate parameters of gamma prior, or shape and scale parameters of inverse gamma prior. If **prior_name** is *ref*, it is a numpy

1d-array that gives the value of the coefficient **a** in the reference prior. When set to *None*, it defaults to `np.array([1.6,0.3])` for gamma or inverse gamma priors. When set to the reference prior, it defaults to `np.array([0.2])`. Defaults to *None*.

- **bds** (*ndarray, optional*) – a numpy 1d-array of length two that gives the lower and upper bounds of the lengthscales. Default to *None*.
- **nugget_est** (*bool, optional*) – set to *True* to estimate nugget term or to *False* to fix the nugget term as specified by the argument **nugget**. If set to *True*, the value set to the argument **nugget** is used as the initial value. Defaults to *False*.
- **scale_est** (*bool, optional*) – set to *True* to estimate the variance or to *False* to fix the variance as specified by the argument **scale**. Defaults to *False*.
- **input_dim** (*ndarray, optional*) – a numpy 1d-array that contains either

  1. the indices of GPs in the feeding layer whose outputs feed into the GP; or
  2. the indices of dimensions in the global input if the GP is in the first layer.

  When set to *None*,

  1. all outputs from GPs in the feeding layer; or
  2. all global input dimensions feed into the GP.

  Defaults to *None*.
- **connect** (*ndarray, optional*) – a numpy 1d-array that contains the indices of dimensions in the global input connecting to the GP as additional input dimensions to the input obtained from the output of GPs in the feeding layer (as determined by the argument **input_dim**). When set to *None*, no global input connection is implemented. Defaults to *None*. When the kernel class is used in GP/DGP emulators for linked emulation and some input dimensions to the computer models are not connected to some feeding computer models, set **connect** to a 1d-array of indices of these external global input dimensions, and accordingly, set **input_dim** to a 1d-array of indices of the remaining input dimensions that are connected to the feeding computer models.

type

identifies that the kernel is a GP.

Type
str

g

a function giving the log probability density function of gamma or inverse gamma distribution ignoring the constant part.

Type
function

gfod

a function giving the first order derivative of **g** with respect to the log-transformed lengthscales and nugget.

Type
function

para_path

a numpy 2d-array that contains the trace of model parameters. Each row is a parameter estimate produced by one SEM iteration. The model parameters in each row are ordered as follow: `np.array([scale estimate, lengthscale estimate (whose length>=1), nugget estimate])`.

Type
ndarray

`global_input`

a numpy 2d-array that contains the connect global input dimensions determined by the argument **connect**. The value of the attribute is assigned during the initialisation of `dgp` class. If **connect** is set to *None*, this attribute is also *None*.

Type
ndarray

`input`

a numpy 2d-array (each row as a data point and each column as a data dimension) that contains the input training data (according to the argument **input_dim**) to the GP. The value of this attribute is assigned during the initialisation of `dgp` class.

Type
ndarray

`output`

a numpy 2d-array with only one column that contains the output training data to the GP. The value of this attribute is assigned during the initialisation of `dgp` class.

Type
ndarray

`rep`

a numpy 1d-array used to re-construct repetitions in the data according to the repetitions in the global input, i.e., rep is assigned during the initialisation of `dgp` class if one input position has multiple outputs. Otherwise, it is *None*. Defaults to *None*.

Type
ndarray

`Rinv`

a numpy 2d-array that stores the inversion of correlation matrix. Defaults to *None*.

Type
ndarray

`Rinv_y`

a numpy 1d-array that stores the product of correlation matrix inverse and the output Y. Defaults to *None*.

Type
ndarray

`rff`

indicates weather random Fourier features are used. Defaults to *None*.

Type
bool

`D`

the dimension of input data to the GP node. Defaults to *None*.

Type
int

`M`

the number of features in random Fourier approximation. Defaults to *None*.

Type
int

W

a 2d-array (M, D) sampled to construct the Fourier approximation to the kernel matrix. Defaults to *None*.

Type
ndarray

b

a 1d-array (D,) sampled to construct the Fourier approximation to the kernel matrix. Defaults to *None*.

Type
ndarray

R2

a 2d-array that stores the R2 of the linear regression between **global_input** and **input**. Defaults to *None*.

Type
ndarray

add_to_path()

Add updated model parameter estimates to the class attribute **para_path**.

compute_cl()

compute_scale()

Compute the scale parameter.

compute_stats()

Compute and store key statistics for the GP predictions

cv_stats(*i*)

Compute the inversion for the LOO.

gfod(*x*)

gp_prediction(*x*, *z*)

Make GP predictions.

Parameters

- **x** (*ndarray*) – a numpy 2d-array that contains the input testing data (whose rows correspond to testing data points and columns correspond to testing data dimensions) with the number of columns same as the **input** attribute.
- **z** (*ndarray*) – a numpy 2d-array that contains additional input testing data (with the same number of columns of the **global_input** attribute) from the global testing input if the argument **connect** is not *None*. Set to None if the argument **connect** is *None*.

Returns
a tuple of two 1d-arrays giving the means and variances at the testing input data positions.

Return type
tuple

k_matrix(*fod_eval=False*)

Compute the correlation matrix and/or first order derivatives of the correlation matrix wrt log-transformed lengthscales and nugget.

Parameters

fod_eval (*bool*) – indicates if the gradient information is also computed along with the correlation matrix. Defaults to *False*.

Returns

1. If **fod_eval** = *False*, a numpy 2d-array *K* is returned as the correlation matrix.
2. If **fod_eval** = *True*, a tuple is returned. It includes *K* and fod, a numpy 3d-array that contains the first order derivatives of the correlation matrix wrt log-transformed lengthscales and nugget. The length of the array equals to the total number of model parameters (i.e., the total number of lengthscales and nugget).

Return type

ndarray_or_tuple

linkgp_prediction(*m*, *v*, *z*, *nb_parallel*)

Make linked GP predictions.

Parameters

- **m** (*ndarray*) – a numpy 2d-array that contains predictive means of testing outputs from the GPs in the last layer. The number of rows equals to the number of testing positions and the number of columns equals to the length of the argument **input_dim**. If the argument **input_dim** is *None*, then the number of columns equals to the number of GPs in the last layer.
- **v** (*ndarray*) – a numpy 2d-array that contains predictive variances of testing outputs from the GPs in the last layer. It has the same shape of **m**.
- **z** (*ndarray*) – a numpy 2d-array that contains additional input testing data (with the same number of columns of the **global_input** attribute) from the global testing input if the argument **connect** is not *None*. Set to *None* if the argument **connect** is *None*.
- **nb_parallel** (*bool*) – whether to use *Numba*'s multi-threading to accelerate the predictions.

Returns

a tuple of two 1d-arrays giving the means and variances at the testing input data positions (that are represented by predictive means and variances).

Return type

tuple

linkgp_prediction_full(*m*, *v*, *m_z*, *v_z*, *z*, *nb_parallel*)

Make linked GP predictions with additional input also generated by GPs/DGPs.

Parameters

- **m** (*ndarray*) – a numpy 2d-array that contains predictive means of testing outputs from the GPs in the last layer. The number of rows equals to the number of testing positions and the number of columns equals to the length of the argument **input_dim**. If the argument **input_dim** is *None*, then the number of columns equals to the number of GPs in the last layer.
- **v** (*ndarray*) – a numpy 2d-array that contains predictive variances of testing outputs from the GPs in the last layer. It has the same shape of **m**.
- **m_z** (*ndarray*) – a numpy 2d-array that contains predictive means of additional input testing data from GPs.
- **v_z** (*ndarray*) – a numpy 2d-array that contains predictive variances of additional input testing data from GPs.

- **z** (*ndarray*) – a numpy 2d-array that contains additional input testing data from the global testing input that are not from GPs. Set to *None* if the argument **connect** is None.
- **nb_parallel** (*bool*) – whether to use *Numba*'s multi-threading to accelerate the predictions.

Returns
a tuple of two 1d-arrays giving the means and variances at the testing input data positions (that are represented by predictive means and variances).

Return type
tuple

llik(*x*)

Compute the negative log-likelihood function of the GP and the first order derivatives of the negative log-likelihood function wrt log-transformed model parameters..

Parameters
x (*ndarray*) – a numpy 1d-array that contains the values of log-transformed model parameters: log-transformed lengthscales followed by the log-transformed nugget.

Returns
a tuple is returned. The tuple contains two numpy 1d-arrays. The first one gives the negative log-likelihood. The second one (whose length equal to the total number of lengthscales and nugget) contains first order derivatives of the negative log-likelihood function wrt log-transformed lengthscales and nugget.

Return type
tuple

llik_rff(*x*)

Compute the negative log-likelihood function of the GP under RFF.

Parameters
x (*ndarray*) – a numpy 1d-array that contains the values of log-transformed model parameters: log-transformed lengthscales followed by the log-transformed nugget.

Returns
a numpy 1d-array giving negative log-likelihood.

Return type
ndarray

log_likelihood_func()

log_likelihood_func_rff()

Compute Gaussian log-likelihood function using random Fourier features (RFF).

log_prior()

Compute the value of log priors specified to the lengthscales and nugget.

Returns
a numpy 1d-array giving the sum of log priors of the lengthscales and nugget.

Return type
ndarray

log_prior_fod()

Compute the first order derivatives of log priors wrt the log-transformed lengthscales and nugget.

Returns
a numpy 1d-array (whose length equal to the total number of lengthscales and nugget) giving the first order derivatives of log priors wrt the log-transformed lengthscales and nugget.

Return type
ndarray

`log_t()`
Log transform the model parameters (lengthscales and nugget).

Returns
a numpy 1d-array of log-transformed model parameters

Return type
ndarray

`maximise(method='L-BFGS-B')`
Optimise and update model parameters by minimising the negative log-likelihood function.

Parameters
method (*str, optional*) – optimisation algorithm. Defaults to *L-BFGS-B*.

`r2(overwritten=False)`
Compute R2 of the linear regression between **global_input** and **input**.

`sample_basis()`
Sample **W** and **b** to construct random Fourier approximations to correlation matrices.

`update(log_theta)`
Update the model parameters (lengthscales and nugget).

Parameters
log_theta (*ndarray*) – optimised numpy 1d-array of log-transformed lengthscales and nugget.

1.6 likelihood_class module

1.7 linkgp module

1.8 utils module

1.9 functions module

dgpsi.functions.**IJ**(*X*, *z_m*, *z_v*, *length*)
Compute I and J involved in linked GP predictions.

dgpsi.functions.**IJ_parallel**(*X*, *z_m*, *z_v*, *length*)
Compute I and J involved in linked GP predictions.

dgpsi.functions.**I_matern_parallel**(*z_v*, *length*, *muA*, *muB*, *zXi*)

dgpsi.functions.**I_sexp**(*X*, *z_m*, *z_v*, *length*)

dgpsi.functions.**J_matern_parallel**(*z_m*, *z_v*, *length*, *Xi*, *Xj*, *zXi*, *zXj*)

dgpsi.functions.**J_sexp**(*X*, *z_m*, *z_v*, *length*, *Psexp*, *R2sexp*)

dgpsi.functions.**Jd**(*X1*, *X2*, *z_m*, *z_v*, *length*)

Compute J components in different input dimensions for Matern2.5 kernel.

dgpsi.functions.**Pmatrix**(*X*)

dgpsi.functions.**Z_fct**(*X*, *W*, *b*, *length*, *M*)

dgpsi.functions.**cond_mean**(*x*, *z*, *w1*, *global_w1*, *Rinv_y*, *length*, *name*)

Make GP predictions.

dgpsi.functions.**fmvn**(*cov*)

Generate multivariate Gaussian random samples without means.

dgpsi.functions.**fmvn_mu**(*mu*, *cov*)

Generate multivariate Gaussian random samples with means.

dgpsi.functions.**fod_exp**(*X*, *K*)

dgpsi.functions.**g**(*coef1*, *coef2*, *x*, *name*)

dgpsi.functions.**ghdiag**(*fct*, *mu*, *var*, *y*)

dgpsi.functions.**gp**(*x*, *z*, *w1*, *global_w1*, *Rinv*, *Rinv_y*, *scale*, *length*, *nugget*, *name*)

Make GP predictions.

dgpsi.functions.**inv_swp**(*X*, *k*)

dgpsi.functions.**k_one_vec**(*X*, *z*, *length*, *name*)

Compute cross-correlation matrix between the testing and training input data.

dgpsi.functions.**link_gp**(*m*, *v*, *z*, *w1*, *global_w1*, *Rinv*, *Rinv_y*, *R2sexp*, *Psexp*, *scale*, *length*, *nugget*, *name*, *nb_parallel*)

Make linked GP predictions.

dgpsi.functions.**link_gp_sexp**(*m*, *v*, *z*, *w1*, *global_w1*, *Rinv*, *Rinv_y*, *R2sexp*, *Psexp*, *scale*, *length*, *nugget*)

Make linked GP predictions for sexp kernels.

dgpsi.functions.**logdet_nb**(*L*)

dgpsi.functions.**matern_coef**(*v*, *u*)

dgpsi.functions.**matern_multi**(*v*, *u*)

dgpsi.functions.**matern_one**(*v*, *u*)

dgpsi.functions.**mice_var**(*x*, *x_extra*, *kernel*, *nugget_s*)

Calculate smoothed predictive variances of the GP using the candidate design set.

dgpsi.functions.**pdist_matern_coef**(*X*)

dgpsi.functions.**pdist_matern_multi**(*X*)

dgpsi.functions.**pdist_matern_one**(*X*)

dgpsi.functions.**quad**(*A*, *B*)

dgpsi.functions.**trace_nb**(*K*)

dgpsi.functions.**trace_sum**(*A*, *B*)

dgpsi.functions.**update_f**(*f*, *nu*, *theta*)

Update ESS proposal samples.

1.10 synthetic module

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

d

dgpsi.functions, 8
dgpsi.imputation, 1
dgpsi.kernel_class, 2

INDEX

A

add_to_path() (*dgpsi.kernel_class.kernel method*), 5

B

b (*dgpsi.kernel_class.kernel attribute*), 5

C

combine() (*in module dgpsi.kernel_class*), 2
compute_cl() (*dgpsi.kernel_class.kernel method*), 5
compute_scale() (*dgpsi.kernel_class.kernel method*), 5
compute_stats() (*dgpsi.kernel_class.kernel method*), 5
cond_mean() (*in module dgpsi.functions*), 9
cv_stats() (*dgpsi.kernel_class.kernel method*), 5

D

D (*dgpsi.kernel_class.kernel attribute*), 4
dgpsi.functions
    module, 8
dgpsi.imputation
    module, 1
dgpsi.kernel_class
    module, 2

F

fmvn() (*in module dgpsi.functions*), 9
fmvn_mu() (*in module dgpsi.functions*), 9
fod_exp() (*in module dgpsi.functions*), 9

G

g (*dgpsi.kernel_class.kernel attribute*), 3
g() (*in module dgpsi.functions*), 9
gfod (*dgpsi.kernel_class.kernel attribute*), 3
gfod() (*dgpsi.kernel_class.kernel method*), 5
ghdiag() (*in module dgpsi.functions*), 9
global_input (*dgpsi.kernel_class.kernel attribute*), 4
gp() (*in module dgpsi.functions*), 9
gp_prediction() (*dgpsi.kernel_class.kernel method*), 5

I

I_matern_parallel() (*in module dgpsi.functions*), 8
I_sexp() (*in module dgpsi.functions*), 8
IJ() (*in module dgpsi.functions*), 8
IJ_parallel() (*in module dgpsi.functions*), 8
imputer (*class in dgpsi.imputation*), 1
input (*dgpsi.kernel_class.kernel attribute*), 4
inv_swp() (*in module dgpsi.functions*), 9

J

J_matern_parallel() (*in module dgpsi.functions*), 8
J_sexp() (*in module dgpsi.functions*), 8
Jd() (*in module dgpsi.functions*), 9

K

k_matrix() (*dgpsi.kernel_class.kernel method*), 5
k_one_vec() (*in module dgpsi.functions*), 9
kernel (*class in dgpsi.kernel_class*), 2
key_stats() (*dgpsi.imputation.imputer method*), 1

L

link_gp() (*in module dgpsi.functions*), 9
link_gp_sexp() (*in module dgpsi.functions*), 9
linkgp_prediction() (*dgpsi.kernel_class.kernel method*), 6
linkgp_prediction_full() (*dgpsi.kernel_class.kernel method*), 6
llik() (*dgpsi.kernel_class.kernel method*), 7
llik_rff() (*dgpsi.kernel_class.kernel method*), 7
log_likelihood_func() (*dgpsi.kernel_class.kernel method*), 7
log_likelihood_func_rff() (*dgpsi.kernel_class.kernel method*), 7
log_prior() (*dgpsi.kernel_class.kernel method*), 7
log_prior_fod() (*dgpsi.kernel_class.kernel method*), 7
log_t() (*dgpsi.kernel_class.kernel method*), 8
logdet_nb() (*in module dgpsi.functions*), 9

M

M (*dgpsi.kernel_class.kernel attribute*), 4
matern_coef() (*in module dgpsi.functions*), 9
matern_multi() (*in module dgpsi.functions*), 9
matern_one() (*in module dgpsi.functions*), 9
maximise() (*dgpsi.kernel_class.kernel method*), 8
mice_var() (*in module dgpsi.functions*), 9

module
    dgpsi.functions, 8
    dgpsi.imputation, 1
    dgpsi.kernel_class, 2

O

one_sample() (*dgpsi.imputation.imputer static method*), 1
one_sample_block() (*dgpsi.imputation.imputer static method*), 1
output (*dgpsi.kernel_class.kernel attribute*), 4

P

para_path (*dgpsi.kernel_class.kernel attribute*), 3
pdist_matern_coef() (*in module dgpsi.functions*), 9
pdist_matern_multi() (*in module dgpsi.functions*), 9
pdist_matern_one() (*in module dgpsi.functions*), 9
Pmatrix() (*in module dgpsi.functions*), 9

Q

quad() (*in module dgpsi.functions*), 9

R

R2 (*dgpsi.kernel_class.kernel attribute*), 5
r2() (*dgpsi.kernel_class.kernel method*), 8
rep (*dgpsi.kernel_class.kernel attribute*), 4
rff (*dgpsi.kernel_class.kernel attribute*), 4
Rinv (*dgpsi.kernel_class.kernel attribute*), 4
Rinv_y (*dgpsi.kernel_class.kernel attribute*), 4

S

sample() (*dgpsi.imputation.imputer method*), 2
sample_basis() (*dgpsi.kernel_class.kernel method*), 8

T

trace_nb() (*in module dgpsi.functions*), 9
trace_sum() (*in module dgpsi.functions*), 9
type (*dgpsi.kernel_class.kernel attribute*), 3

U

update() (*dgpsi.kernel_class.kernel method*), 8
update_f() (*in module dgpsi.functions*), 10

W

W (*dgpsi.kernel_class.kernel attribute*), 5

Z

Z_fct() (*in module dgpsi.functions*), 9