pyGPGO Documentation

Release 0.1.0.dev1

Jose Jimenez

May 16, 2017

CONTENTS

pyGPGO is a simple and modular Python (>3.5) package for Bayesian Optimization. It supports:

- Different surrogate models: Gaussian Processes, Student-t Processes, Random Forests, Gradient Boosting Machines.
- Type II Maximum-Likelihood of covariance function hyperparameters.
- MCMC sampling for full-Bayesian inference of hyperparameters (via pyMC3).
- Integrated acquisition functions

Check us out on Github.

Overall, pyGPGO is a very easy to use package. In practice, a user needs to specify:

- A function to optimize according to some parameters.
- A dictionary defining parameters, their type and bounds.
- **A surrogate model, such as a Gaussian Process, from the surrogates module.**
    - Some surrogate models require defining a covariance function, with hyperparameters. (from the covfunc module)
- An acquisition strategy, from the acquisition module.
- A GPGO instance, from the GPGO module

A simple example can be checked below:

code:: python import numpy as np from pyGPGO.covfunc import squaredExponential from pyGPGO.acquisition import Acquisition from pyGPGO.surrogates.GaussianProcess import GaussianProcess from pyGPGO.GPGO import GPGO

def f(x): return (np.sin(x))

sexp = squaredExponential() gp = GaussianProcess(sexp) acq = Acquisition(mode='ExpectedImprovement') param = {'x': ('cont', [0, 2 * np.pi])}

np.random.seed(23) gpgo = GPGO(gp, acq, f, param) gpgo.run(max_iter=20)

Contents:

CHAPTER

ONE

PYGPGO DOCUMENTATION

Contents:

1.1 pyGPGO.GPGO module

class pyGPGO.GPGO.**GPGO**(*GPRegressor*, *Acquisition*, *f*, *parameter_dict*, *n_jobs=1*)

Bases: `object`

Bayesian Optimization class.

Parameters

- **GPRegressor** (*GaussianProcess instance*) – Gaussian Process surrogate model instance.
- **Acquisition** (*Acquisition instance*) – Acquisition instance.
- **f** (*fun*) – Function to maximize over parameters specified by *parameter_dict*.
- **parameter_dict** (*dict*) – Dictionary specifying parameter, their type and bounds.
- **n_jobs** (*int. Default 1*) – Parallel threads to use during acquisition optimization.

parameter_key

list – Parameters to consider in optimization

parameter_type

list – Parameter types.

parameter_range

list – Parameter bounds during optimization

history

list – Target values evaluated along the procedure.

_acqWrapper(*xnew*)

Evaluates the acquisition function on a point.

Parameters **xnew** (*np.ndarray, shape=((len(self.parameter_key),))*) – Point to evaluate the acquisition function on.

Returns Acquisition function value for *xnew*.

Return type float

_firstRun(*n_eval=3*)

Performs initial evaluations before fitting GP.

Parameters **n_eval** (*int*) – Number of initial evaluations to perform. Default is 3.

_optimizeAcq(*method='L-BFGS-B'*, *n_start=100*)
Optimizes the acquisition function using a multistart approach.

Parameters

- **method** (*str. Default 'L-BFGS-B'.*) – Any *scipy.optimize* method that admits bounds and gradients. Default is 'L-BFGS-B'.
- **n_start** (*int.*) – Number of starting points for the optimization procedure. Default is 100.

_sampleParam()
Randomly samples parameters over bounds.

Returns A random sample of specified parameters.

Return type dict

getResult()
Prints best result in the Bayesian Optimization procedure.

Returns

- *OrderedDict* – Point yielding best evaluation in the procedure.
- *float* – Best function evaluation.

run(*max_iter=10*, *init_evals=3*, *resume=False*)
Runs the Bayesian Optimization procedure.

Parameters

- **max_iter** (*int*) – Number of iterations to run. Default is 10.
- **init_evals** (*int*) – Initial function evaluations before fitting a GP. Default is 3.
- **resume** (*bool*) – Whether to resume the optimization procedure from the last evaluation. Default is *False*.

updateGP()
Updates the internal model with the next acquired point and its evaluation.

1.2 pyGPGO.surrogates package

1.2.1 Submodules

pyGPGO.surrogates.BoostedTrees module

class pyGPGO.surrogates.BoostedTrees.**BoostedTrees**(*q1=0.16*, *q2=0.84*, ***params*)
Bases: object

Gradient boosted trees as surrogate model for Bayesian Optimization. Uses quantile regression for an estimate of the 'posterior' variance. In practice, the std is computed as (*q2* - *q1*) / 2. Relies on *sklearn.ensemble.GradientBoostingRegressor*

Parameters

- **q1** (*float*) – First quantile.
- **q2** (*float*) – Second quantile
- **params** (*tuple*) – Extra parameters to pass to *GradientBoostingRegressor*

fit (*X*, *y*)
Fit a GBM model to data *X* and targets *y*.

Parameters

- **X** (*array-like*) – Input values.
- **y** (*array-like*) – Target values.

predict (*Xstar*, *return_std=True*)
Predicts 'posterior' mean and variance for the GBM model.

Parameters

- **Xstar** (*array-like*) – Input values.
- **return_std** (*bool, optional*) – Whether to return posterior variance estimates. Default is *True*.
- **eps** (*float, optional*) – Floating precision value for negative variance estimates. Default is *1e-6*

Returns

- *array-like* – Posterior predicted mean.
- *array-like* – Posterior predicted std

update (*xnew*, *ynew*)
Updates the internal RF model with observations *xnew* and targets *ynew*.

Parameters

- **xnew** (*array-like*) – New observations.
- **ynew** (*array-like*) – New targets.

pyGPGO.surrogates.GaussianProcess module

class pyGPGO.surrogates.GaussianProcess.**GaussianProcess** (*covfunc*, *optimize=False*, *usegrads=False*, *mprior=0*)
Bases: object

Gaussian Process regressor class. Based on Rasmussen & Williams [1]_ algorithm 2.1.

Parameters

- **covfunc** (*instance from a class of covfunc module*) – Covariance function. An instance from a class in the *covfunc* module.
- **optimize** (*bool:*) – Whether to perform covariance function hyperparameter optimization.
- **usegrads** (*bool*) – Whether to use gradient information on hyperparameter optimization. Only used if *optimize=True*.

covfunc
object – Internal covariance function.

optimize
bool – User chosen optimization configuration.

usegrads
bool – Gradient behavior

mprior
float – Explicit value for the mean function of the prior Gaussian Process.

Notes

[1] Rasmussen, C. E., & Williams, C. K. I. (2004). Gaussian processes for machine learning. International journal of neural systems (Vol. 14). http://doi.org/10.1142/S0129065704001899

_grad (*param_vector*, *param_key*)
Returns gradient for each hyperparameter, evaluated at a given point.

Parameters

- **param_vector** (*list*) – List of values corresponding to hyperparameters to query.
- **param_key** (*list*) – List of hyperparameter strings corresponding to *param_vector*.

Returns Gradient for each evaluated hyperparameter.

Return type np.ndarray

_lmlik (*param_vector*, *param_key*)
Returns marginal negative log-likelihood for given covariance hyperparameters.

Parameters

- **param_vector** (*list*) – List of values corresponding to hyperparameters to query.
- **param_key** (*list*) – List of hyperparameter strings corresponding to *param_vector*.

Returns Negative log-marginal likelihood for chosen hyperparameters.

Return type float

fit (*X*, *y*)
Fits a Gaussian Process regressor

Parameters

- **X** (*np.ndarray, shape=(nsamples, nfeatures)*) – Training instances to fit the GP.
- **y** (*np.ndarray, shape=(nsamples,)*) – Corresponding continuous target values to X.

getcovparams ()
Returns current covariance function hyperparameters

Returns Dictionary containing covariance function hyperparameters

Return type dict

optHyp (*param_key*, *param_bounds*, *grads=None*, *n_trials=5*)
Optimizes the negative marginal log-likelihood for given hyperparameters and bounds. This is an empirical Bayes approach (or Type II maximum-likelihood).

Parameters

- **param_key** (*list*) – List of hyperparameters to optimize.
- **param_bounds** (*list*) – List containing tuples defining bounds for each hyperparameter to optimize over.

param_grad (*k_param*)
Returns gradient over hyperparameters. It is recommended to use *self._grad* instead.

Parameters **k_param** (*dict*) – Dictionary with keys being hyperparameters and values their queried values.

Returns Gradient corresponding to each hyperparameters. Order given by *k_param.keys()*

Return type np.ndarray

predict (*Xstar*, *return_std=False*)
Returns mean and covariances for the posterior Gaussian Process.

Parameters

- **Xstar** (*np.ndarray, shape=((nsamples, nfeatures))*) – Testing instances to predict.
- **return_std** (*bool*) – Whether to return the standard deviation of the posterior process. Otherwise, it returns the whole covariance matrix of the posterior process.

Returns

- *np.ndarray* – Mean of the posterior process for testing instances.
- *np.ndarray* – Covariance of the posterior process for testing instances.

update (*xnew*, *ynew*)
Updates the internal model with *xnew* and *ynew* instances.

Parameters

- **xnew** (*np.ndarray, shape=((m, nfeatures))*) – New training instances to update the model with.
- **ynew** (*np.ndarray, shape=((m,))*) – New training targets to update the model with.

pyGPGO.surrogates.GaussianProcessMCMC module

class pyGPGO.surrogates.GaussianProcessMCMC.**GaussianProcessMCMC** (*covfunc*, *niter=2000*, *burnin=1000*, *init='ADVI'*, *step=None*)

Bases: object

Gaussian Process class using MCMC sampling of covariance function hyperparameters.

Parameters

- **covfunc** – Covariance function to use. Currently this instance only supports *squaredExponential* and *matern* v=1.5, 2.5 kernel
- **niter** (*int*) – Number of iterations to run MCMC.
- **burnin** (*int*) – Burn-in iterations to discard at trace.
- **init** (*str*) – Initialization method for NUTS. Check pyMC3 docs.

fit (*X*, *y*)
Fits a Gaussian Process regressor using MCMC.

Parameters

- **X** (*np.ndarray, shape=(nsamples, nfeatures)*) – Training instances to fit the GP.

- **y** (*np.ndarray, shape=(nsamples,)*) – Corresponding continuous target values to *X*.

posteriorPlot()

Plots sampled posterior distributions for hyperparameters.

predict(*Xstar*, *return_std=False*, *nsamples=10*)

Returns mean and covariances for each posterior sampled Gaussian Process.

Parameters

- **Xstar** (*np.ndarray, shape=((nsamples, nfeatures))*) – Testing instances to predict.
- **return_std** (*bool*) – Whether to return the standard deviation of the posterior process. Otherwise, it returns the whole covariance matrix of the posterior process.
- **nsamples** – Number of posterior MCMC samples to consider.

Returns

- *np.ndarray* – Mean of the posterior process for each MCMC sample and *Xstar*.
- *np.ndarray* – Covariance posterior process for each MCMC sample and *Xstar*.

update(*xnew*, *ynew*)

Updates the internal model with *xnew* and *ynew* instances.

Parameters

- **xnew** (*np.ndarray, shape=((m, nfeatures))*) – New training instances to update the model with.
- **ynew** (*np.ndarray, shape=((m,))*) – New training targets to update the model with.

pyGPGO.surrogates.RandomForest module

class pyGPGO.surrogates.RandomForest.**ExtraForest**(***params*)

Bases: object

Wrapper around sklearn's ExtraTreesRegressor implementation for pyGPGO. Random Forests can also be used for surrogate models in Bayesian Optimization. An estimate of 'posterior' variance can be obtained by using the *impurity* criterion value in each subtree.

Parameters **params** (*tuple, optional*) – Any parameters to pass to *RandomForestRegressor*. Defaults to sklearn's.

fit(*X*, *y*)

Fit a Random Forest model to data *X* and targets *y*.

Parameters

- **X** (*array-like*) – Input values.
- **y** (*array-like*) – Target values.

predict(*Xstar*, *return_std=True*, *eps=1e-06*)

Predicts 'posterior' mean and variance for the RF model.

Parameters

- **Xstar** (*array-like*) – Input values.

- **return_std** (*bool, optional*) – Whether to return posterior variance estimates. Default is *True*.
- **eps** (*float, optional*) – Floating precision value for negative variance estimates. Default is *1e-6*

Returns

- *array-like* – Posterior predicted mean.
- *array-like* – Posterior predicted std

update (*xnew*, *ynew*)
Updates the internal RF model with observations *xnew* and targets *ynew*.

Parameters

- **xnew** (*array-like*) – New observations.
- **ynew** (*array-like*) – New targets.

class pyGPGO.surrogates.RandomForest.**RandomForest** (***params*)
Bases: object

Wrapper around sklearn's Random Forest implementation for pyGPGO. Random Forests can also be used for surrogate models in Bayesian Optimization. An estimate of 'posterior' variance can be obtained by using the *impurity* criterion value in each subtree.

Parameters params (*tuple, optional*) – Any parameters to pass to *RandomForestRegressor*. Defaults to sklearn's.

fit (*X*, *y*)
Fit a Random Forest model to data *X* and targets *y*.

Parameters

- **X** (*array-like*) – Input values.
- **y** (*array-like*) – Target values.

predict (*Xstar*, *return_std=True*, *eps=1e-06*)
Predicts 'posterior' mean and variance for the RF model.

Parameters

- **Xstar** (*array-like*) – Input values.
- **return_std** (*bool, optional*) – Whether to return posterior variance estimates. Default is *True*.
- **eps** (*float, optional*) – Floating precision value for negative variance estimates. Default is *1e-6*

Returns

- *array-like* – Posterior predicted mean.
- *array-like* – Posterior predicted std

update (*xnew*, *ynew*)
Updates the internal RF model with observations *xnew* and targets *ynew*.

Parameters

- **xnew** (*array-like*) – New observations.
- **ynew** (*array-like*) – New targets.

pyGPGO.surrogates.tStudentProcess module

pyGPGO.surrogates.tStudentProcess.**logpdf**(*x, df, mu, Sigma*)

Marginal log-likelihood of a Student-t Process

Parameters

- **x** (*array-like*) – Point to be evaluated
- **df** (*float*) – Degrees of freedom (>2.0)
- **mu** (*array-like*) – Mean of the process.
- **Sigma** (*array-like*) – Covariance matrix of the process.

Returns **logp** – log-likelihood

Return type float

class pyGPGO.surrogates.tStudentProcess.**tStudentProcess**(*covfunc, nu=3.0, optimize=False*)

Bases: object

t-Student Process regressor class. This class DOES NOT support gradients in ML estimation yet.

Parameters

- **covfunc** (*instance from a class of covfunc module*) – An instance from a class from the *covfunc* module.
- **nu** (*float*) – (>2.0) Degrees of freedom

covfunc

object – Internal covariance function.

nu

float – Degrees of freedom.

optimize

bool – Whether to optimize covariance function hyperparameters.

_lmlik(*param_vector, param_key*)

Returns marginal negative log-likelihood for given covariance hyperparameters.

Parameters

- **param_vector** (*list*) – List of values corresponding to hyperparameters to query.
- **param_key** (*list*) – List of hyperparameter strings corresponding to *param_vector*.

Returns Negative log-marginal likelihood for chosen hyperparameters.

Return type float

fit(*X, y*)

Fits a t-Student Process regressor

Parameters

- **X** (*np.ndarray, shape=(nsamples, nfeatures)*) – Training instances to fit the GP.
- **y** (*np.ndarray, shape=(nsamples,)*) – Corresponding continuous target values to *X*.

getcovparams()

Returns current covariance function hyperparameters

Returns Dictionary containing covariance function hyperparameters

Return type dict

optHyp (*param_key*, *param_bounds*, *n_trials=5*)
Optimizes the negative marginal log-likelihood for given hyperparameters and bounds. This is an empirical Bayes approach (or Type II maximum-likelihood).

Parameters

- **param_key** (*list*) – List of hyperparameters to optimize.
- **param_bounds** (*list*) – List containing tuples defining bounds for each hyperparameter to optimize over.

predict (*Xstar*, *return_std=False*)
Returns mean and covariances for the posterior t-Student process.

Parameters

- **Xstar** (*np.ndarray, shape=((nsamples, nfeatures))*) – Testing instances to predict.
- **return_std** (*bool*) – Whether to return the standard deviation of the posterior process. Otherwise, it returns the whole covariance matrix of the posterior process.

Returns

- *np.ndarray* – Mean of the posterior process for testing instances.
- *np.ndarray* – Covariance of the posterior process for testing instances.

update (*xnew*, *ynew*)
Updates the internal model with *xnew* and *ynew* instances.

Parameters

- **xnew** (*np.ndarray, shape=((m, nfeatures))*) – New training instances to update the model with.
- **ynew** (*np.ndarray, shape=((m,))*) – New training targets to update the model with.

pyGPGO.surrogates.tStudentProcessMCMC module

class pyGPGO.surrogates.tStudentProcessMCMC.**tStudentProcessMCMC** (*covfunc*, *nu=3.0*, *niter=2000*, *burnin=1000*, *init='ADVI'*, *step=None*)

Bases: object

Student-t class using MCMC sampling of covariance function hyperparameters.

Parameters

- **covfunc** – Covariance function to use. Currently this instance only supports *squaredExponential* and *Matern* from the *covfunc* module.
- **nu** (*float*) – Degrees of freedom (>2.0)
- **niter** (*int*) – Number of iterations to run MCMC.
- **burnin** (*int*) – Burn-in iterations to discard at trace.

- **init** (*str*) – Initialization method for NUTS. Check pyMC3 docs.

fit (*X*, *y*)

Fits a Student-t regressor using MCMC.

Parameters

- **X** (*np.ndarray, shape=(nsamples, nfeatures)*) – Training instances to fit the GP.
- **y** (*np.ndarray, shape=(nsamples,)*) – Corresponding continuous target values to *X*.

posteriorPlot ()

Plots sampled posterior distributions for hyperparameters.

predict (*Xstar*, *return_std=False*, *nsamples=10*)

Returns mean and covariances for each posterior sampled Student-t Process.

Parameters

- **Xstar** (*np.ndarray, shape=((nsamples, nfeatures))*) – Testing instances to predict.
- **return_std** (*bool*) – Whether to return the standard deviation of the posterior process. Otherwise, it returns the whole covariance matrix of the posterior process.
- **nsamples** – Number of posterior MCMC samples to consider.

Returns

- *np.ndarray* – Mean of the posterior process for each MCMC sample and Xstar.
- *np.ndarray* – Covariance posterior process for each MCMC sample and Xstar.

update (*xnew*, *ynew*)

Updates the internal model with *xnew* and *ynew* instances.

Parameters

- **xnew** (*np.ndarray, shape=((m, nfeatures))*) – New training instances to update the model with.
- **ynew** (*np.ndarray, shape=((m,))*) – New training targets to update the model with.

1.2.2 Module contents

1.3 pyGPGO.covfunc module

class pyGPGO.covfunc.**dotProd** (*sigmaf=1.0*, *sigman=1e-06*, *bounds=None*, *parameters=['sigmaf', 'sigman']*)

Bases: object

Dot-product kernel class.

Parameters

- **sigmaf** (*float*) – Signal variance. Controls the overall scale of the covariance function.
- **sigman** (*float*) – Noise variance. Additive noise in output space.

- **bounds** (*list*) – List of tuples specifying hyperparameter range in optimization procedure.
- **parameters** (*list*) – List of strings specifying which hyperparameters should be optimized.

K (*X*, *Xstar*)
Computes covariance function values over *X* and *Xstar*.

Parameters

- **X** (*np.ndarray, shape=((n, nfeatures))*) – Instances
- **Xstar** (*np.ndarray, shape=((n, nfeatures))*) – Instances

Returns Computed covariance matrix.

Return type np.ndarray

gradK (*X*, *Xstar*, *param*)
Computes gradient matrix for instances *X*, *Xstar* and hyperparameter *param*.

Parameters

- **X** (*np.ndarray, shape=((n, nfeatures))*) – Instances
- **Xstar** (*np.ndarray, shape=((n, nfeatures))*) – Instances
- **param** (*str*) – Parameter to compute gradient matrix for.

Returns Gradient matrix for parameter *param*.

Return type np.ndarray

class pyGPGO.covfunc.**expSine** (*l=1.0, period=1.0, bounds=None, parameters=['l', 'period']*)
Bases: object

Exponential sine kernel class.

Parameters

- **l** (*float*) – Characteristic length-scale. Units in input space in which posterior GP values do not change significantly. l: float
- **period** (*float*) – Period hyperparameter.
- **bounds** (*list*) – List of tuples specifying hyperparameter range in optimization procedure.
- **parameters** (*list*) – List of strings specifying which hyperparameters should be optimized.

K (*X*, *Xstar*)
Computes covariance function values over *X* and *Xstar*.

Parameters

- **X** (*np.ndarray, shape=((n, nfeatures))*) – Instances
- **Xstar** (*np.ndarray, shape=((n, nfeatures))*) – Instances

Returns Computed covariance matrix.

Return type np.ndarray

gradK (*X*, *Xstar*, *param*)

class `pyGPGO.covfunc.`**`gammaExponential`**(*gamma=1, l=1, sigmaf=1, sigman=1e-06, bounds=None, parameters=['gamma', 'l', 'sigmaf', 'sigman']*)

Bases: `object`

Gamma-exponential kernel class.

Parameters

- **`gamma`** (*`float`*) – Hyperparameter of the Gamma-exponential covariance function.
- **`l`** (*`float`*) – Characteristic length-scale. Units in input space in which posterior GP values do not change significantly.
- **`sigmaf`** (*`float`*) – Signal variance. Controls the overall scale of the covariance function.
- **`sigman`** (*`float`*) – Noise variance. Additive noise in output space.
- **`bounds`** (*`list`*) – List of tuples specifying hyperparameter range in optimization procedure.
- **`parameters`** (*`list`*) – List of strings specifying which hyperparameters should be optimized.

`K`(*X, Xstar*)

Computes covariance function values over *X* and *Xstar*.

Parameters

- **`X`** (*`np.ndarray, shape=((n, nfeatures))`*) – Instances
- **`Xstar`** (*`np.ndarray, shape=((n, nfeatures))`*) – Instances

Returns Computed covariance matrix.

Return type np.ndarray

`gradK`(*X, Xstar, param*)

Computes gradient matrix for instances *X*, *Xstar* and hyperparameter *param*.

Parameters

- **`X`** (*`np.ndarray, shape=((n, nfeatures))`*) – Instances
- **`Xstar`** (*`np.ndarray, shape=((n, nfeatures))`*) – Instances
- **`param`** (*`str`*) – Parameter to compute gradient matrix for.

Returns Gradient matrix for parameter *param*.

Return type np.ndarray

`pyGPGO.covfunc.`**`kronDelta`**(*X, Xstar*)

Computes Kronecker delta for rows in X and Xstar.

Parameters

- **`X`** (*`np.ndarray, shape=((n, nfeatures))`*) – Instances.
- **`Xstar`** (*`np.ndarray, shape((m, nfeatures))`*) – Instances.

Returns Kronecker delta between row pairs of *X* and *Xstar*.

Return type np.ndarray

`pyGPGO.covfunc.`**`l2norm_`**(*X, Xstar*)

Wrapper function to compute the L2 norm

Parameters

- **X** (*np.ndarray, shape=((n, nfeatures))*) – Instances.
- **Xstar** (*np.ndarray, shape=((m, nfeatures))*) – Instances

Returns Pairwise euclidian distance between row pairs of *X* and *Xstar*.

Return type np.ndarray

class pyGPGO.covfunc.**matern** (*v=1, l=1, sigmaf=1, sigman=1e-06, bounds=None, parameters=['v', 'l', 'sigmaf', 'sigman']*)

Bases: object

Matern kernel class.

Parameters

- **v** (*float*) – Scale-mixture hyperparameter of the Matern covariance function.
- **l** (*float*) – Characteristic length-scale. Units in input space in which posterior GP values do not change significantly.
- **sigmaf** (*float*) – Signal variance. Controls the overall scale of the covariance function.
- **sigman** (*float*) – Noise variance. Additive noise in output space.
- **bounds** (*list*) – List of tuples specifying hyperparameter range in optimization procedure.
- **parameters** (*list*) – List of strings specifying which hyperparameters should be optimized.

K (*X, Xstar*)

Computes covariance function values over *X* and *Xstar*.

Parameters

- **X** (*np.ndarray, shape=((n, nfeatures))*) – Instances
- **Xstar** (*np.ndarray, shape=((n, nfeatures))*) – Instances

Returns Computed covariance matrix.

Return type np.ndarray

class pyGPGO.covfunc.**matern32** (*l=1, sigmaf=1, sigman=1e-06, bounds=None, parameters=['l', 'sigmaf', 'sigman']*)

Bases: object

Matern v=3/2 kernel class.

Parameters

- **l** (*float*) – Characteristic length-scale. Units in input space in which posterior GP values do not change significantly.
- **sigmaf** (*float*) – Signal variance. Controls the overall scale of the covariance function.
- **sigman** (*float*) – Noise variance. Additive noise in output space.
- **bounds** (*list*) – List of tuples specifying hyperparameter range in optimization procedure.
- **parameters** (*list*) – List of strings specifying which hyperparameters should be optimized.

K (*X, Xstar*)

Computes covariance function values over *X* and *Xstar*.

Parameters

- **X** (*np.ndarray, shape=((n, nfeatures))*) – Instances
- **Xstar** (*np.ndarray, shape=((n, nfeatures))*) – Instances

Returns Computed covariance matrix.

Return type np.ndarray

gradK (*X*, *Xstar*, *param*)

Computes gradient matrix for instances *X*, *Xstar* and hyperparameter *param*.

Parameters

- **X** (*np.ndarray, shape=((n, nfeatures))*) – Instances
- **Xstar** (*np.ndarray, shape=((n, nfeatures))*) – Instances
- **param** (*str*) – Parameter to compute gradient matrix for.

Returns Gradient matrix for parameter *param*.

Return type np.ndarray

class pyGPGO.covfunc.**matern52** (*l=1, sigmaf=1, sigman=1e-06, bounds=None, parameters=['l', 'sigmaf', 'sigman']*)

Bases: object

Matern v=5/2 kernel class.

Parameters

- **l** (*float*) – Characteristic length-scale. Units in input space in which posterior GP values do not change significantly.
- **sigmaf** (*float*) – Signal variance. Controls the overall scale of the covariance function.
- **sigman** (*float*) – Noise variance. Additive noise in output space.
- **bounds** (*list*) – List of tuples specifying hyperparameter range in optimization procedure.
- **parameters** (*list*) – List of strings specifying which hyperparameters should be optimized.

K (*X*, *Xstar*)

Computes covariance function values over *X* and *Xstar*.

Parameters

- **X** (*np.ndarray, shape=((n, nfeatures))*) – Instances
- **Xstar** (*np.ndarray, shape=((n, nfeatures))*) – Instances

Returns Computed covariance matrix.

Return type np.ndarray

gradK (*X*, *Xstar*, *param*)

Computes gradient matrix for instances *X*, *Xstar* and hyperparameter *param*.

Parameters

- **X** (*np.ndarray, shape=((n, nfeatures))*) – Instances
- **Xstar** (*np.ndarray, shape=((n, nfeatures))*) – Instances
- **param** (*str*) – Parameter to compute gradient matrix for.

Returns Gradient matrix for parameter *param*.

Return type np.ndarray

class `pyGPGO.covfunc.`**`rationalQuadratic`**(*alpha=1, l=1, sigmaf=1, sigman=1e-06, bounds=None, parameters=['alpha', 'l', 'sigmaf', 'sigman']*)

Bases: `object`

Rational-quadratic kernel class.

Parameters

- **`alpha`** (`float`) – Hyperparameter of the rational-quadratic covariance function.
- **`l`** (`float`) – Characteristic length-scale. Units in input space in which posterior GP values do not change significantly.
- **`sigmaf`** (`float`) – Signal variance. Controls the overall scale of the covariance function.
- **`sigman`** (`float`) – Noise variance. Additive noise in output space.
- **`bounds`** (`list`) – List of tuples specifying hyperparameter range in optimization procedure.
- **`parameters`** (`list`) – List of strings specifying which hyperparameters should be optimized.

`K`(*X*, *Xstar*)

Computes covariance function values over *X* and *Xstar*.

Parameters

- **`X`** (*`np.ndarray, shape=((n, nfeatures))`*) – Instances
- **`Xstar`** (*`np.ndarray, shape=((n, nfeatures))`*) – Instances

Returns Computed covariance matrix.

Return type np.ndarray

`gradK`(*X*, *Xstar*, *param*)

Computes gradient matrix for instances *X*, *Xstar* and hyperparameter *param*.

Parameters

- **`X`** (*`np.ndarray, shape=((n, nfeatures))`*) – Instances
- **`Xstar`** (*`np.ndarray, shape=((n, nfeatures))`*) – Instances
- **`param`** (`str`) – Parameter to compute gradient matrix for.

Returns Gradient matrix for parameter *param*.

Return type np.ndarray

class `pyGPGO.covfunc.`**`squaredExponential`**(*l=1, sigmaf=1.0, sigman=1e-06, bounds=None, parameters=['l', 'sigmaf', 'sigman']*)

Bases: `object`

Squared exponential kernel class.

Parameters

- **`l`** (`float`) – Characteristic length-scale. Units in input space in which posterior GP values do not change significantly.
- **`sigmaf`** (`float`) – Signal variance. Controls the overall scale of the covariance function.
- **`sigman`** (`float`) – Noise variance. Additive noise in output space.

- **bounds** (*list*) – List of tuples specifying hyperparameter range in optimization procedure.
- **parameters** (*list*) – List of strings specifying which hyperparameters should be optimized.

K(*X*, *Xstar*)
Computes covariance function values over *X* and *Xstar*.

Parameters

- **X** (*np.ndarray, shape=((n, nfeatures))*) – Instances
- **Xstar** (*np.ndarray, shape=((n, nfeatures))*) – Instances

Returns Computed covariance matrix.

Return type np.ndarray

gradK(*X*, *Xstar*, *param='l'*)
Computes gradient matrix for instances *X*, *Xstar* and hyperparameter *param*.

Parameters

- **X** (*np.ndarray, shape=((n, nfeatures))*) – Instances
- **Xstar** (*np.ndarray, shape=((n, nfeatures))*) – Instances
- **param** (*str*) – Parameter to compute gradient matrix for.

Returns Gradient matrix for parameter *param*.

Return type np.ndarray

1.4 pyGPGO.acquisition module

class pyGPGO.acquisition.**Acquisition**(*mode*, *eps=1e-06*, ***params*)
Bases: object

Acquisition function class.

Parameters

- **mode** (*str*) – Defines the behaviour of the acquisition strategy. Currently supported values are *ExpectedImprovement*, *IntegratedExpectedÌmprovement*, *ProbabilityImprovement*, *IntegratedProbabilityImprovement*, *UCB*, *IntegratedUCB*, *Entropy*, *tExpectedImprovement*, and *tIntegratedExpectedImprovement*. Integrated improvement functions are only to be used with MCMC surrogates.
- **eps** (*float*) – Small floating value to avoid *np.sqrt* or zero-division warnings.
- **params** (*float*) – Extra parameters needed for certain acquisition functions, e.g. UCB needs to be supplied with *beta*.

Entropy(*tau*, *mean*, *std*, *sigman*)
Predictive entropy acquisition function

Parameters

- **tau** (*float*) – Best observed function evaluation.
- **mean** (*float*) – Point mean of the posterior process.
- **std** (*float*) – Point std of the posterior process.

- **sigman** (*float*) – Noise variance

Returns Predictive entropy.

Return type float

ExpectedImprovement (*tau*, *mean*, *std*)
Expected Improvement acquisition function.

Parameters

- **tau** (*float*) – Best observed function evaluation.
- **mean** (*float*) – Point mean of the posterior process.
- **std** (*float*) – Point std of the posterior process.

Returns Expected improvement.

Return type float

IntegratedExpectedImprovement (*tau*, *meanmcmc*, *stdmcmc*)
Integrated expected improvement. Can only be used with *GaussianProcessMCMC* instance.

Parameters

- **tau** (*float*) – Best observed function evaluation
- **meanmcmc** (*array-like*) – Means of posterior predictive distributions after sampling.
- **stdmcmc** – Standard deviations of posterior predictive distributions after sampling.

Returns Integrated Expected Improvement

Return type float

IntegratedProbabilityImprovement (*tau*, *meanmcmc*, *stdmcmc*)
Integrated probability of improvement. Can only be used with *GaussianProcessMCMC* instance.

Parameters

- **tau** (*float*) – Best observed function evaluation
- **meanmcmc** (*array-like*) – Means of posterior predictive distributions after sampling.
- **stdmcmc** – Standard deviations of posterior predictive distributions after sampling.

Returns Integrated Probability of Improvement

Return type float

IntegratedUCB (*tau*, *meanmcmc*, *stdmcmc*, *beta*)
Integrated probability of improvement. Can only be used with *GaussianProcessMCMC* instance.

Parameters

- **tau** (*float*) – Best observed function evaluation
- **meanmcmc** (*array-like*) – Means of posterior predictive distributions after sampling.
- **stdmcmc** – Standard deviations of posterior predictive distributions after sampling.
- **beta** (*float*) – Hyperparameter controlling exploitation/exploration ratio.

Returns Integrated UCB.

Return type float

ProbabilityImprovement (*tau*, *mean*, *std*)
Probability of Improvement acquisition function.

Parameters

- **tau** (*float*) – Best observed function evaluation.
- **mean** (*float*) – Point mean of the posterior process.
- **std** (*float*) – Point std of the posterior process.

Returns Probability of improvement.

Return type float

UCB (*tau, mean, std, beta*)

Upper-confidence bound acquisition function.

Parameters

- **tau** (*float*) – Best observed function evaluation.
- **mean** (*float*) – Point mean of the posterior process.
- **std** (*float*) – Point std of the posterior process.
- **beta** (*float*) – Hyperparameter controlling exploitation/exploration ratio.

Returns Upper confidence bound.

Return type float

eval (*tau, mean, std*)

Evaluates selected acquisition function.

Parameters

- **tau** (*float*) – Best observed function evaluation.
- **mean** (*float*) – Point mean of the posterior process.
- **std** (*float*) – Point std of the posterior process.

Returns Acquisition function value.

Return type float

tExpectedImprovement (*tau, mean, std, nu=3.0*)

Expected Improvement acquisition function. Only to be used with *tStudentProcess* surrogate.

Parameters

- **tau** (*float*) – Best observed function evaluation.
- **mean** (*float*) – Point mean of the posterior process.
- **std** (*float*) – Point std of the posterior process.

Returns Expected improvement.

Return type float

tIntegratedExpectedImprovement (*tau, meanmcmc, stdmcmc, nu=3.0*)

Integrated expected improvement. Can only be used with *tStudentProcessMCMC* instance.

Parameters

- **tau** (*float*) – Best observed function evaluation
- **meanmcmc** (*array-like*) – Means of posterior predictive distributions after sampling.
- **stdmcmc** – Standard deviations of posterior predictive distributions after sampling.

- **nu** – Degrees of freedom.

Returns Integrated Expected Improvement

Return type float

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

Symbols

A

B

C

D

E

F

G

H

I