Safe Learning Documentation

Release 0.1

Felix Berkenkamp, Matteo Turchetta, Angela P. Schoellig, Andreas

Feb 10, 2020

Contents

CHAPTER 1

Introduction

TODO

1.1 API Documentation

The *safeopt* package implements tools for Safe Bayesian optimization.

1.1.1 Stability verification

The `Lyapunov` class provides the main point of entry for the stability analysis. It can be used to compute the region of attraction and together with `get_safe_sample()` sets up the safe sampling scheme.

`Lyapunov`(discretization, lyapunov_function, ...)	A class for general Lyapunov functions.
`get_safe_sample`(lyapunov[, perturbations, ...])	Compute a safe state-action pair for sampling.
`smallest_boundary_value`(fun, discretization)	Determine the smallest value of a function on its boundary.
`get_lyapunov_region`(lyapunov, ...)	Get the region within which a function is a Lyapunov function.

1.1.2 Approximate Dynamics Programming

We use approximate dynamics programming to compute value functions.

`PolicyIteration`(policy, dynamics, ... [, gamma])	A class for policy iteration.

1.1.3 Functions

These are generic function classes for convenience. They are all compatible with `Lyapunov` and `PolicyIteration` and can be added, multiplied, and stacked as needed.

`GridWorld`(limits, num_points)	Base class for function approximators on a regular grid.
`FunctionStack`(functions[, name])	A combination of multiple 1d (uncertain) functions for each dim.
`Triangulation`(discretization, vertex_values)	Efficient Delaunay triangulation on regular grid.
`PiecewiseConstant`(discretization[, ...])	A piecewise constant function approximator.
`LinearSystem`(matrices[, name])	A linear system.
`QuadraticFunction`(matrix[, name])	A quadratic function.
`Saturation`(fun, lower, upper[, name])	Saturate the output of a *DeterministicFunction*.
`NeuralNetwork`(layers, nonlinearities[, ...])	A simple neural network.
`GaussianProcess`(gaussian_process[, beta, name])	A GaussianProcess model based on gpflow.
`GPRCached`(*args, **kwargs)	Create a new *Mock* object.
`sample_gp_function`(discretization, gpfun[, ...])	Sample a function from a gp with corresponding kernel within its bounds.

1.1.4 Utilities

These are utilities to make working with tensorflow more pleasant.

`utilities.combinations`(arrays)	Return a single array with combinations of parameters.
`utilities.linearly_spaced_combinations`(...)	Return 2-D array with all linearly spaced combinations with the bounds.
`utilities.lqr`(a, b, q, r)	Compute the continuous time LQR-controller.
`utilities.dlqr`(a, b, q, r)	Compute the discrete-time LQR controller.
`utilities.ellipse_bounds`(P, level[, n])	Compute the bounds of a 2D ellipse.
`utilities.concatenate_inputs`([start])	Concatenate the numpy array inputs to the functions.
`utilities.make_tf_fun`(return_type[, ...])	Convert a python function to a tensorflow function.
`utilities.with_scope`(name)	Set the tensorflow scope for the function.
`utilities.use_parent_scope`(function)	Use the parent scope for tensorflow.
`utilities.add_weight_constraint`(...)	Add weight constraints to an optimization step.
`utilities.batchify`(arrays, batch_size)	Yield the arrays in batches and in order.
`utilities.get_storage`(dictionary[, index])	Get a unique storage point within a class method.
`utilities.set_storage`(dictionary, name_value)	Set the storage point within a class method.
`utilities.unique_rows`(array)	Return the unique rows of the array.
`utilities.gradient_clipping`(optimizer, loss, ...)	Clip the gradients for the optimization problem.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index

S