Descent Documentation

Release 0.2.5

Niru Maheswaranathan

Jun 30, 2017

Contents

descent is a library for performing optimization.

Contents:

CHAPTER 1

Installation

1.1 Basic

The easiest way to install is with `pip`:

```
$ pip install descent
```

You can also install from source by grabbing the code from GitHub:

```
$ git clone https://github.com/nirum/descent.git
$ cd descent
$ pip install -r requirements.txt
$ python setup.py install
```

1.2 Dependencies

Descent works on Python 3.4-3.5. It has only been tested on CPython (not tested on PyPy yet!).

In addition, descent requires the following packages:

- `numpy`
- `toolz`
- `multipledispatch`
- `future`

And the following are optional:

- `scipy`

1.3 Development

Please submit any issues to the GitHub issue tracker.

To contribute to descent, you'll need to also install `sphinx` and `numpydoc` for documentation and `nose` for testing. We adhere to the NumPy/SciPy documentation standards.

CHAPTER 2

Quickstart

2.1 Overview

This document is a work in progress, but for now, check out example code:

2.2 Gradient-based algorithms

Each of the gradient based algorithms has the following interface. Given a function `f_df` that computes the objective and gradient of the function you want to minimize:

```
>>> opt = descent.GradientDescent(theta_init, f_df, 'sgd', {'lr': learning_rate})
>>> opt.run(maxiter=1000)
>>> plt.plot(opt.theta)
```

2.3 Proximal algorithms

Example code for ADMM, for solving a linear system with a sparsity penalty:

```
>>> opt = descent.Consensus(theta_init)
>>> opt.add('linsys', A, b)
>>> opt.add('sparse', 0.1)
>>> opt.run()
>>> plt.plot(opt.theta)
```

2.4 Storage

After calling the `run` command, the history of objective values is stored on the optimizer object:

```
>>> opt.run(maxiter=1000)
>>> plt.plot(opt.storage['objective'])
```

2.5 Utilities

Some other features that might be of interest:

- memoization (see: `descent.utils.wrap`)
- function wrapping (see: `descent.utils.destruct` and `descent.utils.restruct`)
- gradient checking (see: `descent.check_grad`)

2.6 Tutorial

There is a tutorial consisting of jupyter notebooks demoing the features of descent at: github.com/nirum/descent-tutorial.

CHAPTER 3

Proximal Operators

The `proximal_operators` module contains functions to compute the following proximal operators:

- Proximal operator of the nuclear norm (`nucnorm`)
- Proximal operator of the l1 norm (`sparse`)
- Proximal operator corresponding to solving a linear system of equations (`linsys`)
- Proximal operator of the l2 norm, a squared error penalty (`squared_error`)
- Proximal operator for minimizing an arbitrary smooth function given an oracle that computes the function value and gradient (`lbfgs`)
- Pxorimal operator for the l2 penalty of the discrete different operator, to encourage smoothness (`smooth`)
- Projection onto the non-negative orthant (`nonneg`)
- Projection onto the semidefinite cone (`semidefinite_cone`)

CHAPTER 4

API Reference

4.1 Main optimization loops

class descent.main.**Consensus**(*tau=(10.0, 2.0, 2.0)*, *tol=(1e-06, 0.001)*)

Bases: descent.main.Optimizer

add(*operator*, **args*)

Adds a proximal operator to the list of operators

minimize(*x0*, *display=None*, *maxiter=inf*)

descent.main.**gradient_optimizer**(*coro*)

Turns a coroutine into a gradient based optimizer.

4.2 Gradient algorithms

First order gradient descent algorithms

descent.algorithms.**sgd**

alias of GradientOptimizer

descent.algorithms.**nag**

alias of GradientOptimizer

descent.algorithms.**rmsprop**

alias of GradientOptimizer

descent.algorithms.**sag**

alias of GradientOptimizer

descent.algorithms.**smorms**

alias of GradientOptimizer

descent.algorithms.**adam**

alias of GradientOptimizer

4.3 Proximal operators

Proximal operators / mappings

descent.proxops.**nucnorm**
alias of ProxOp

descent.proxops.**sparse**
alias of ProxOp

class descent.proxops.**linsys**(*A*, *b*)
Bases: descent.proxops.ProximalOperatorBaseClass

descent.proxops.**squared_error**
alias of ProxOp

descent.proxops.**identity**
alias of ProxOp

descent.proxops.**lbfgs**
alias of ProxOp

descent.proxops.**tvd**
alias of ProxOp

descent.proxops.**smooth**
alias of ProxOp

descent.proxops.**linear**
alias of ProxOp

descent.proxops.**fantope**
alias of ProxOp

4.4 Objectives

Example objectives

descent.objectives.**rosenbrock**(*theta*)
Objective and gradient for the rosenbrock function

descent.objectives.**sphere**(*theta*)
l2-norm of the parameters

descent.objectives.**matyas**(*theta*)
Matyas function

descent.objectives.**beale**(*theta*)
Beale's function

descent.objectives.**booth**(*theta*)
Booth's function

descent.objectives.**mccormick**(*theta*)
McCormick function

descent.objectives.**camel**(*theta*)
Three-hump camel function

descent.objectives.**michalewicz**(*theta*)

descent.objectives.**bohachevsky1**(*theta*)
One of the Bohachevsky functions

descent.objectives.**zakharov**(*theta*)
Zakharov function

descent.objectives.**dixon_price**(*theta*)
Dixon-Price function

4.5 Utilities

descent.utils.**check_grad**(*f_df*, *xref*, *stepsize=1e-06*, *tol=1e-06*, *width=15*, *style='round'*, *out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)
Compares the numerical gradient to the analytic gradient

Parameters

- **f_df** (*function*) – The analytic objective and gradient function to check
- **x0** (*array_like*) – Parameter values to check the gradient at
- **stepsize** (*float, optional*) – Stepsize for the numerical gradient. Too big and this will poorly estimate the gradient. Too small and you will run into precision issues (default: 1e-6)
- **tol** (*float, optional*) – Tolerance to use when coloring correct/incorrect gradients (default: 1e-5)
- **width** (*int, optional*) – Width of the table columns (default: 15)
- **style** (*string, optional*) – Style of the printed table, see tableprint for a list of styles (default: 'round')

descent.utils.**destruct**(**args*, ***kwargs*)
Deconstructs the input into a 1-D numpy array

descent.utils.**restruct**(**args*, ***kwargs*)
Reshapes the input into the type of the second argument

descent.utils.**wrap**(*f_df*, *xref*, *size=1*)
Memoizes an objective + gradient function, and splits it into two functions that return just the objective and gradient, respectively.

Parameters

- **f_df** (*function*) – Must be unary (takes a single argument)
- **xref** (*list, dict, or array_like*) – The form of the parameters
- **size** (*int, optional*) – Size of the cache (Default=1)

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index