sciscipydirect Documentation

Release 1.0.0

Andreas Mayer

Jul 09, 2017

Contents

DIRECT is a method to solve global bound constraint optimization problems and was originally developed by D. R. Jones, C. D. Perttunen and B. E. Stuckmann. It is designed to find **global** solutions of mathematical optimization problems of the from

$$\min_{x \in R^n} f(x)$$

subject to

$$x_L \leq x \leq x_U$$

Where x are the optimization variables (with upper and lower bounds), $f(x)$ is the objective function.

The DIRECT package uses the Fortran implementation of DIRECT written by Joerg.M.Gablonsky, DIRECT Version 2.0.4. More information on the DIRECT algorithm can be found in Gablonsky's thesis.

CHAPTER 1

Further reading

Installation

The quickest way to install is to type:

```
$ pip install scipydirect
```

More detailed instructions follow. To install scipydirect you will need the following prerequisites:

- python 2.6+
- numpy
- C++ compiler
- FORTRAN compiler

Python(x,y) is a great way to get all of these if you are using windows and satisfied with 32bit.

Download the source files of scipydirect, unzip, and then execute:

```
$ python setup.py install
```

You can test the installation by running the examples under the folder `test/`. Some of the examples require matplotlib.

Tutorial - Solving the six-hump camelback function

Filename: `test/C6.py`

The following tutorial shows how to find the global minimum of a *Six-hump camelback function* using the *DIRECT* algorithm.

$$f(x_1, x_2) = (4 - 2.1x_1^2 + x_1^4 + x_1^4/3)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2,$$
$$\Omega = [-3, 3] \times [-2, 2].$$

First we need to import the `solve` function from the *DIRECT* package:

```
>>> from scipydirect import minimize
```

Then we need to define the objective of the function:

```
>>> def obj(x):
  ...     """Six-hump camelback function"""
  ...     x1 = x[0]
  ...     x2 = x[1]
  ...     f = (4 - 2.1*(x1*x1) + (x1*x1*x1*x1)/3.0)*(x1*x1) + x1*x2 + (-4 +␣
↪4*(x2*x2))*(x2*x2)
  ...     return f
```

We need to define the domain of the problem using block constraints:

```
>>> bounds = [(-3, 3), (-2, 2)]
```

We use the *DIRECT* algorithm to solve the optimization problem. The algoritm is called using the `minimize` function. The *solve* functions accepts the problem objective `obj` and block constraints:

```
>>> res = minimize(obj, bounds)
```

In the above we use the default settings of the *DIRECT* algorithm. It us possible to costumize the algorithm using the parameters of the `minimize` function (see *scipydirect.minimize()*).

The `minimize` function returns a result object `res` making accessible the optimal point, `res.x`, the value of the objective at the optimum, `res.fun`, and a status message `res.ierror`.

We can visualize the problem using *matplotlib*:

```
>>> fig = plt.figure()
>>> ax = fig.add_subplot(111, projection='3d')

>>> x = res.x
>>> X, Y = np.mgrid[x[0]-1:x[0]+1:50j, x[1]-1:x[1]+1:50j]
>>> Z = np.zeros_like(X)

>>> for i in range(X.size):
  ...     Z.ravel()[i] = obj([X.flatten()[i], Y.flatten()[i]])

>>> ax.plot_wireframe(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet)
>>> ax.scatter(x[0], x[1], res.fun, c='r', marker='o')
>>> ax.set_title('Six-hump Camelback Function')
>>> ax.view_init(30, 45)
>>> plt.show()
```

This results in

More examples can be found in the source distribution under the `test/` folder.

Reference

This is the class and function reference of pydirect. Please refer to the *tutorial* for further details, as the class and function raw specifications may not be enough to give full guidelines on their uses.

scipydirect.**minimize**(*func*, *bounds=None*, *nvar=None*, *args=()*, *disp=False*, *eps=0.0001*, *maxf=20000*, *maxT=6000*, *algmethod=0*, *fglobal=-1e+100*, *fglper=0.01*, *volper=-1.0*, *sigmaper=-1.0*, ***kwargs*)

Solve an optimization problem using the DIRECT (Dividing Rectangles) algorithm. It can be used to solve general nonlinear programming problems of the form:

$$\min_{x \in R^n} f(x)$$

subject to

$$x_L \leq x \leq x_U$$

Where x are the optimization variables (with upper and lower bounds), $f(x)$ is the objective function.

Parameters **func** : objective function

called as func(x, *args); does not need to be defined everywhere, raise an Exception where function is not defined

bounds : array-like

`(min, max)` pairs for each element in `x`, defining the bounds on that parameter.

nvar: integer :

Dimensionality of x (only needed if *bounds* is not defined)

eps : float

Ensures sufficient decrease in function value when a new potentially optimal interval is chosen.

maxf : integer

Approximate upper bound on objective function evaluations.

Note: Maximal allowed value is 90000 see documentation of Fortran library.

---

maxT : integer

Maximum number of iterations.

Note: Maximal allowed value is 6000 see documentation of Fortran library.

---

algmethod : integer

Whether to use the original or modified DIRECT algorithm. Possible values:

- `algmethod=0` - use the original DIRECT algorithm
- `algmethod=1` - use the modified DIRECT-l algorithm

fglobal : float

Function value of the global optimum. If this value is not known set this to a very large negative value.

fglper : float

Terminate the optimization when the percent error satisfies:

$$100 * (f_{min} - f_{global}) / \max(1, |f_{global}|) \leq f_{glper}$$

volper : float

Terminate the optimization once the volume of a hyperrectangle is less than volper percent of the original hyperrectangel.

sigmaper : float

Terminate the optimization once the measure of the hyperrectangle is less than sigmaper.

Returns **res** : OptimizeResult

The optimization result represented as a `OptimizeResult` object. Important attributes are: `x` the solution array, `success` a Boolean flag indicating if the optimizer exited successfully and `message` which describes the cause of the termination. See *OptimizeResult* for a description of other attributes.

class `scipydirect.`**`OptimizeResult`**

Bases: `dict`

Represents the optimization result.

Attributes **x** : ndarray

The solution of the optimization.

success : bool

Whether or not the optimizer exited successfully.

status : int

Termination status of the optimizer. Its value depends on the underlying solver. Refer to *message* for details.

message : str

Description of the cause of the termination.

fun, jac, hess, hess_inv : ndarray

Values of objective function, Jacobian, Hessian or its inverse (if available). The Hessians may be approximations, see the documentation of the function in question.

nfev, njev, nhev : int

Number of evaluations of the objective functions and of its Jacobian and Hessian.

nit : int

Number of iterations performed by the optimizer.

maxcv : float

The maximum constraint violation.

Notes

There may be additional attributes not listed above depending of the specific solver. Since this class is essentially a subclass of dict with attribute accessors, one can see which attributes are available using the *keys()* method.

Changelog

Version 1.2

- add numpy to install_requires and fix problem with newer f2py versions

Version 1.1

- added Python 3 support
- allow objective function to raise exception where undefined

Version 1.0

Initial version

Python Module Index

S

Index