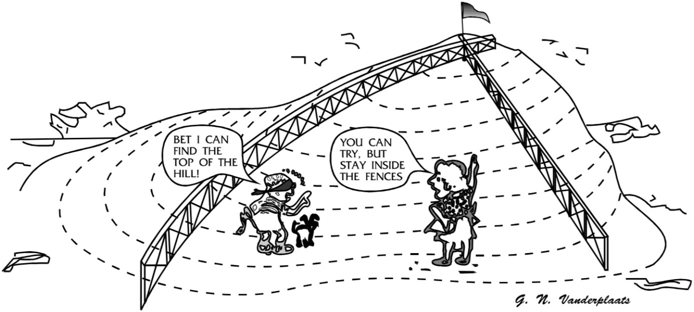
EML4930/EML6934: Lecture 08

Optimization with scipy.optimize

Charles Jekel

October 19, 2017

What is optimization?



Cartoon by Dr. Gary Vanderplaats of `http://www.vrand.com/` some of these example problems and HW problems are from the DOT reference manual.

Mathematical optimization formulation

Objective function:

$$\min F(\boldsymbol{x}) \tag{1}$$

Inequality constraints:

$$G(\boldsymbol{x}) \leq 0 \tag{2}$$

Variable constraints

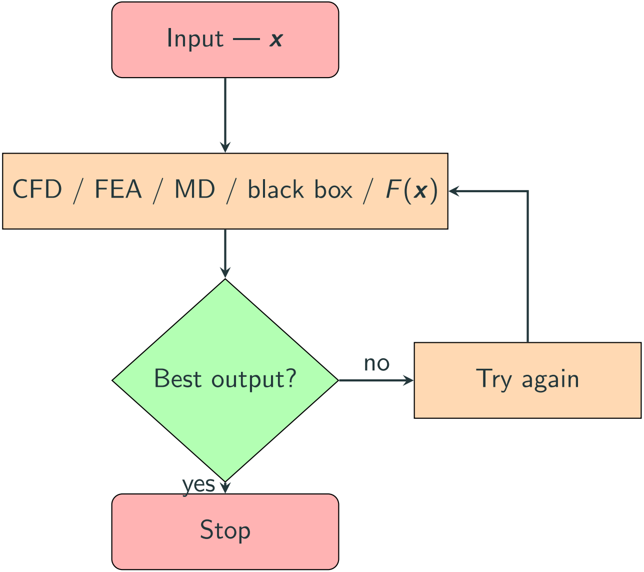
$$x_1^L \leq x_1 \leq x_1^U \tag{3}$$

$$x_2^L \leq x_2 \leq x_2^U \tag{4}$$

$$\cdots \tag{5}$$

$$x_n^L \leq x_n \leq x_n^U \tag{6}$$

Optimization in engineering: finding the best input



scipy.optimize at a glance

- local and global optimization algorithms
- gradient and stochastic
- constrainted and unconstrained algorithms

https://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html
https://docs.scipy.org/doc/scipy/reference/optimize.html

Unconstrained multivariate methods

Method	Description
fmin	Minimize a function using the downhill simplex algorithm.
fmin_powell	Minimize a function using modified Powells method.
fmin_cg	Nonlinear conjugate gradient algorithm.
fmin_bfgs	Minimize a function using the BFGS algorithm.
fmin_ncg	Minimization of a function using the Newton-CG method.

Constrained multivariate methods

Method	Description
fmin_l_bfgs_b	Minimize using the L-BFGS-B algorithm.
fmin_tnc	Minimize a function with truncated Newton algorithm.
fmin_cobyla	Constrained Optimization BY Linear Approximation.
fmin_slsqp	Minimize using Sequential Least SQuares Programming
differential_evolution	Finds the global minimum of a multivariate function.

Global optimization methods

Method	Description
basinhopping	Global minimum using the basin-hopping algorithm
brute	Minimize a function over a given range by brute force.
differential_evolution	Finds the global minimum of a multivariate function.

Methods I like

Method	My use	Pitfall
fmin_bfgs	local	local minima
fmin_l_bfgs_b	bounded local	local minima
fmin_slsqp $*$	Constrained local	Quadratic and local minima
differential_evolution	Global optimization	Number of function evaluations

$*$ the only true constrained optimization algorithm...

Optimizing engineering problems

- optimization is a great design tool
- FEA/ CFD/ MD take a long time to evaluate
- can only afford a limited number of function evaluations
- there is no method that will work well on all problems (see No Free Lunch by Wolpert and Macready 1997) `https://ti.arc.nasa.gov/m/profile/dhw/papers/78.pdf`
- Gradient based methods, non-gradient (stochastic) based methods, surrogate base methods, and various combinations

Gradient based methods

- work well when you have an initial design
- guaranteed to find an optima (thought it might be a local one)
- work with a large number of design variables ($n > 1000$)
- make the most of your function evaluations
- deal with multiminima by running multiple optimization from different starting point
- function must be smooth and near continuous!

Global optimization methods

- large number of function evaluations (which is fine when you can afford it)
- stochastic/evolutionary methods not guaranteed to converge to a minima
- sometimes we just want to find the best solution
- functions can be discontinuous

fmin_bfgs basic gradient based method

https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fmin_bfgs.html#scipy.optimize.fmin_bfgs

```
res = fmin_bfgs(f, x0, fprime=None, args=(), gtol=1e-05,
norm=inf, epsilon=1.4901161193847656e-08, maxiter=None,
full_output=0,  disp=1, retall=0, callback=None)
```

fmin_l_bfgs_b constrained version of BFGS

https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fmin_l_bfgs_b.html#scipy.optimize.fmin_l_bfgs_b

```
res = fmin_l_bfgs_b(func, x0, fprime=None, args=(),
approx_grad=0, bounds=None, m=10, factr=10000000.0,
pgtol=1e-05, epsilon=1e-08,  iprint=-1, maxfun=15000,
maxiter=15000, disp=None,   callback=None, maxls=20)
```

differential evolution

https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential_evolution.html#scipy.optimize.differential_evolution

```
res = differential_evolution(func, bounds, args=(),
strategy='best1bin', maxiter=1000, popsize=15, tol=0.01,
 mutation=(0.5, 1), recombination=0.7,  seed=None,
 callback=None,  disp=False, polish=True,
 init='latinhypercube', atol=0)
```

differential evolution parameters

I really love all the feature with this algorithm...

- strategy: the differential evolution strategy
- polish: if true a L-BFGS-B optimization is run with the optima found by differential evolution (This is a true meta-heuristic algorithm! - great global optimization)
- init: by default the first generation is made with a latin hypercube sampling!

warning this could use a considerable number of function evaluations

Example 1: non-linear regression with BFGS 1 of 3

Consider fitting a function

$$f(\boldsymbol{\beta}, x) = \frac{\beta_0 x}{\beta_1 + x} \tag{7}$$

in this case $\boldsymbol{\beta}$ are the design variables and x are the data points.

For this example I'm going to fit this function to some data points.

In most cases you won't know the exact beta parameters that the data comes from, but it makes for an easy example to know the solution.

Example 1: non-linear regression with BFGS 2 of 3

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import optimize

# generate some data from known beta values
x = np.linspace(0,10,30)
beta0 = 2.7
beta1 = 1.3
y = (beta0*x)/(beta1+x)

# determine beta by minimizing the mean residual error
def my_func(X):
    yHat = (X[0]*x)/(X[1]+x)
    resid = yHat - y
    return np.mean(np.abs(resid))
```

Example 1: non-linear regression with BFGS 3 of 3

```python
# peform the optimization with bfgs
x0 = [3.0, 3.0] # initial guess
res = optimize.fmin_bfgs(my_func, x0, full_output=True)

plt.figure()
plt.plot(x,y,'o')
beta = res[0] # these are the resulting beta parameters
# plot the resulting curve
plt.plot(x,(beta[0]*x)/(beta[1]+x))
plt.show()
```

Example 2: constrained optimization 1 of 4

Minimize

$$F(\boldsymbol{x}) = (x_0 + x_1)^2 + (x_1 + x_2)^2 \tag{8}$$

subject to

$$h_0 = x_0 + 2x_1 + 3x_2 - 1 = 0 \tag{9}$$

from the initial design point of

$$\boldsymbol{x_0} = [-4.0, 1.0, 2.0] \tag{10}$$

Note: You always set up your equality constraints equal to 0!

Example 2: constrained optimization 2 of 4

```python
# objective function
def func(X):
    F = (X[0]+X[1])**2 + (X[1]+X[2])**2
    return F
# equality constraint
def f_con(X):
    G = X[0] + 2.0*X[1] + 3.0*X[2] - 1.0
    return G
# initial design point
x0 = np.array([-4.0, 1.0, 2.0])
res = optimize.fmin_slsqp(func, x0, f_eqcons=f_con,
    iter=1000, acc=1e-06, disp=True, full_output=True)
```

Minimize

$$F(\boldsymbol{x}) = (x_0 + x_1)^2 + (x_1 + x_2)^2 \tag{11}$$

subject to an alternative inequality constraints

$$g_0 = x_0 + 2x_1 + 3x_2 - 1 \leq 0 \tag{12}$$

$$g_1 = -g_0 \leq 0 \tag{13}$$

from the initial design point of

$$\boldsymbol{x_0} = [-4.0, 1.0, 2.0] \tag{14}$$

Note: You always set up your inequality constraints less than or equal to zero!

Example 2: constrained optimization 4 of 4

```
# objective function
def func(X):
    F = (X[0]+X[1])**2 + (X[1]+X[2])**2
    return F
# inequality constraint
# note slsqp handles Gradient contraints as >= 0 and not <=0
# so for this case i have G >= 0 and -G > = 0
# don't ask me why... I have no clue why
def f_con1(X):
    G = X[0] + 2.0*X[1] + 3.0*X[2] - 1.0
    return G, -G
# initial design point
x0 = np.array([-4.0, 1.0, 2.0])
res = optimize.fmin_slsqp(func, x0, f_ieqcons=f_con1,
    iter=1000, acc=1e-06, disp=True, full_output=True)
```

Example 3: Global optimization of the Adjiman function

Minimize

$$f(\boldsymbol{x}) = \cos(x_0)\sin(x_1) - \frac{x_0}{x_1^2 + 1} \tag{15}$$

on the domain

$$-10 \leq x_0 \leq 10 \tag{16}$$

$$-10 \leq x_1 \leq 10 \tag{17}$$

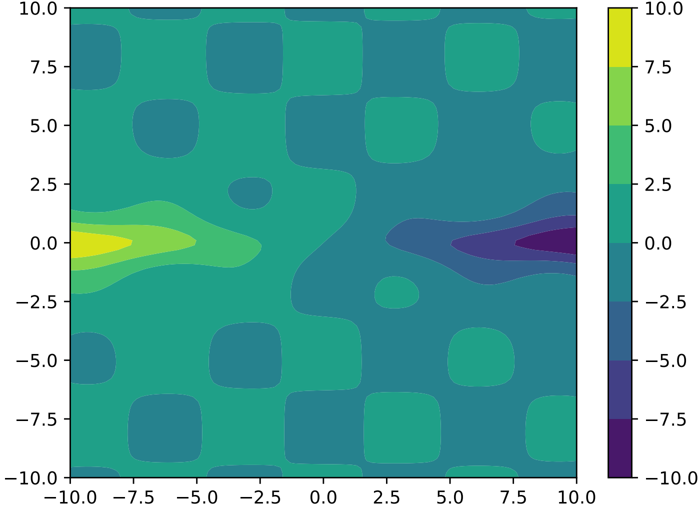
Example 3: Global optimization of the Adjiman function

Differential evolution works well on this multimodal problem.

```
# objective function
def adjiman(x):
    F = np.cos(x[0])*np.sin(x[1]) - ((x[0])/(x[1]**2 +1.0))
    return F

# optimization bounds
bounds = ((-10.0,10.0),
          (-10.0,10.0))
# run differential evolution
res = optimize.differential_evolution(adjiman, bounds,
    maxiter=1000, popsize=50, disp=True)
```

Example 3: Global optimization of the Adjiman function



Example 3: Global optimization of the Adjiman function

Issues with L-BFGS-B

```
# objective function
def adjiman(x):
    F = np.cos(x[0])*np.sin(x[1]) - ((x[0])/(x[1]**2 +1.0))
    return F

# optimization bounds
bounds = ((-10.0,10.0),
          (-10.0,10.0))

# this l bfgs b won't find the optimum
res2 = optimize.fmin_l_bfgs_b(adjiman, (-2,-2),
    approx_grad=True, bounds=bounds)

# however this l bfgs b will
res3 = optimize.fmin_l_bfgs_b(adjiman, (2,2),
    approx_grad=True, bounds=bounds)
```