quickstart

Unknown Author

August 21, 2013

title: SIGOPT: a python package for sigmoidal programming author: Madeleine Udell

1 SIGOPT: a python package for sigmoidal programming

SIGOPT is a python package for solving sigmoidal programming problems of the following form:

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{n} f_i(x_i) \\ \text{subject to} & Ax \leq b, \\ & Cx = d, \\ & l \leq x \leq u, \end{array}$$

where A, b, C, d, l, u are matrices or lists of the appropriate dimensions, and f_i, $i = 1, \ldots, k$ are *sigmoidal* functions. f_i is *sigmoidal* if it is *convex* on the interval $[l_i, z_i]$, and *concave* on the interval $[z_i, u_i]$.

(It's ok if $l_i = z_i$ or $z_i = u_i$, so f_i can be just convex or just concave.)

To understand how SIGOPT works, and to learn more about the theory of sigmoidal programming, you can take a look at this paper.

To learn more about how to use SIGOPT, read on.

2 Download

SIGOPT is available via the python package index. You can install it using `easy_install`

```
$ easy_install sigopt
```

or `pip`

```
$ pip install sigopt
```

Or the old-fashioned way: download it from PyPI, unzip the package, and install:

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

2.1 Dependencies

Before SIGOPT will work, you'll need to install the GNU Linear Programming Kit (GLPK). Make sure to install GLPK in a standard location so that PyGLPK can find it.

With the exception of GLPK, all of the dependencies are available via the python package index, and are automatically downloaded and installed if they are not already found on the system.

- PyGLPK
- cvxopt
- numpy
- scipy
- matplotlib

3 Quickstart examples

3.1 A simple sigmoidal program

As an example to get you started, we'll show how to solve the problem

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{9} logistic(x_i) \\ \text{subject to} & \sum_i x_i \leq 0, \\ & -5 \leq x \leq 5. \end{array}$$

Each function f_i is specified as a tuple consisting of the function, its derivative, and its inflection point z_i.

The logistic function $logistic(x) = 1/\exp(-x)$ and its derivative are defined for convenience in `sigopt.functions`, and its inflection point is located at $z = 0$.

In [1]:
```
import sigopt
from sigopt.functions import logistic, logistic_prime

# define f_i
fi = [(logistic, logistic_prime, 0)]
# each f is identical, so we just repeat f_i 9 times
n = 9
fs = fi*n
```

We write the constraint $\sum_i x_i \leq 0$ as $Ax \leq b$

In [2]:
```
A = numpy.ones((1,n))
b = numpy.ones((1,1))
```

and the box constraints $l \leq x \leq u$ are

In [3]:
```
l = [-5]*n
u = [5]*n
```

(these can be expressed either using numpy matrices, numpy arrays, or python lists — actually, anything iterable will do).

Now we're ready to formulate the problem

In [4]:
```
from sigopt import Problem
p = Problem(l,u,fs,A=A,b=b,name='example1',tol=.1,sub_tol=.01)
```

The parameter `tol` sets the accuracy to which we need to solve the problem, and `sub_tol` controls the quality of the upper approximation to the functions f_i. Finally we solve!

In [5]:
```
p.solve(maxiters = 100)
```

(It's a good idea to specify a maximum number of iterations, since sigmoidal programs can me NP-hard.)

We can examine the bounds on the optimal value, and the point x achieving those bounds.

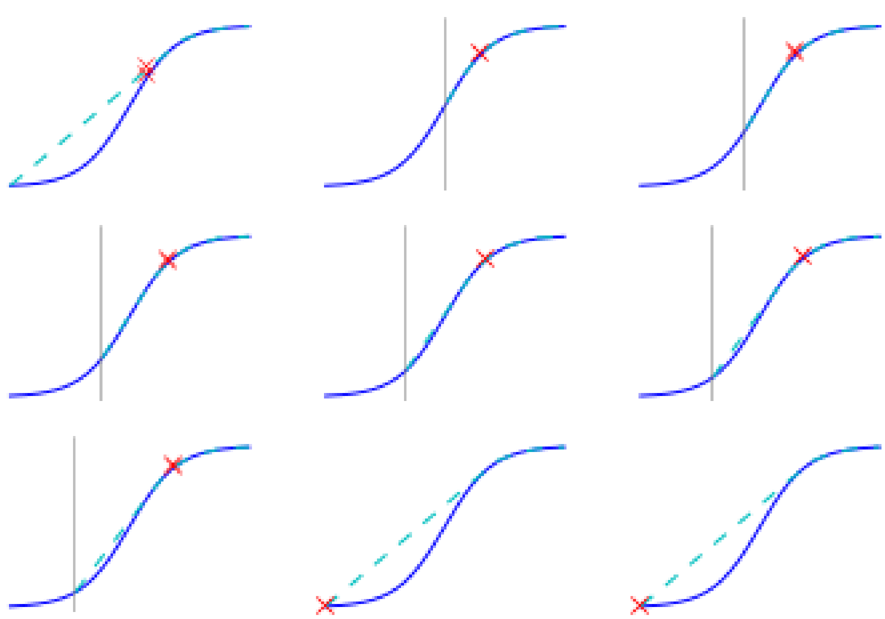
In [6]:
```
print 'bounds',p.bounds
print 'solution',p.x
```

```
bounds [(5.514779744166213, 5.820557154787409), (5.514779744166213,
5.823791589918229), (5.520849002950583, 5.828114180032125),
(5.546101925223407, 5.83318656497994), (5.601172194820388,
5.838282831707601), (5.6870586416451, 5.842525542312434),
(5.770608770092042, 5.846244704731432)]
solution [0.8001359691991636, 1.5064185015401645, 1.5438259874988653,
1.645357260519695, 1.7503615765704392, 1.8404877155833232,
1.9134129890883498, -5.0, -5.0]
```

Of course, it's nicer to see what's going on in a picture. We can plot the best solution we've found so far using the plotting library.

In [7]:
```
from sigopt.plot import *
f = plot_best_node(p)
f.show()
```

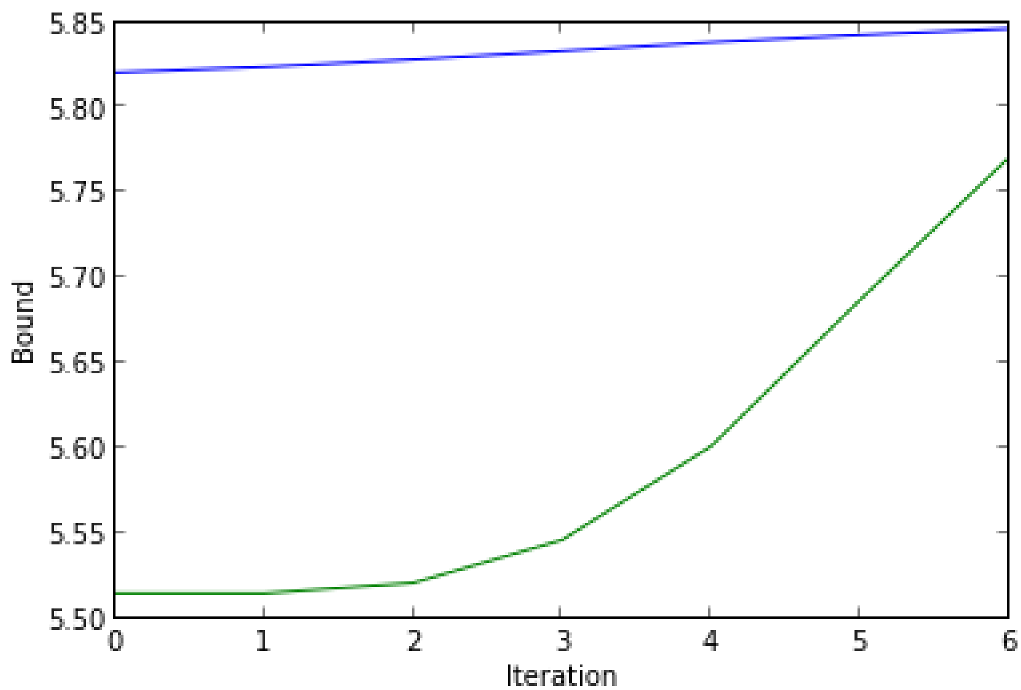
```
Saving figure in example1_best_node.eps

/Users/Madeleine/Applications/anaconda/lib/python2.7/site-
packages/matplotlib/figure.py:361: UserWarning: matplotlib is
currently using a non-GUI backend, so cannot show the figure
  "matplotlib is currently using a non-GUI backend, "
```



And if we'd like to see how quickly the algorithm converged, there's a convenient convergence plot.

In [8]:
```
f = plot_convergence(p)
f.show()
```

```
Saving plot of history in example1_convergence.eps
```



3.2 Function definitions, equality and inequality constraints

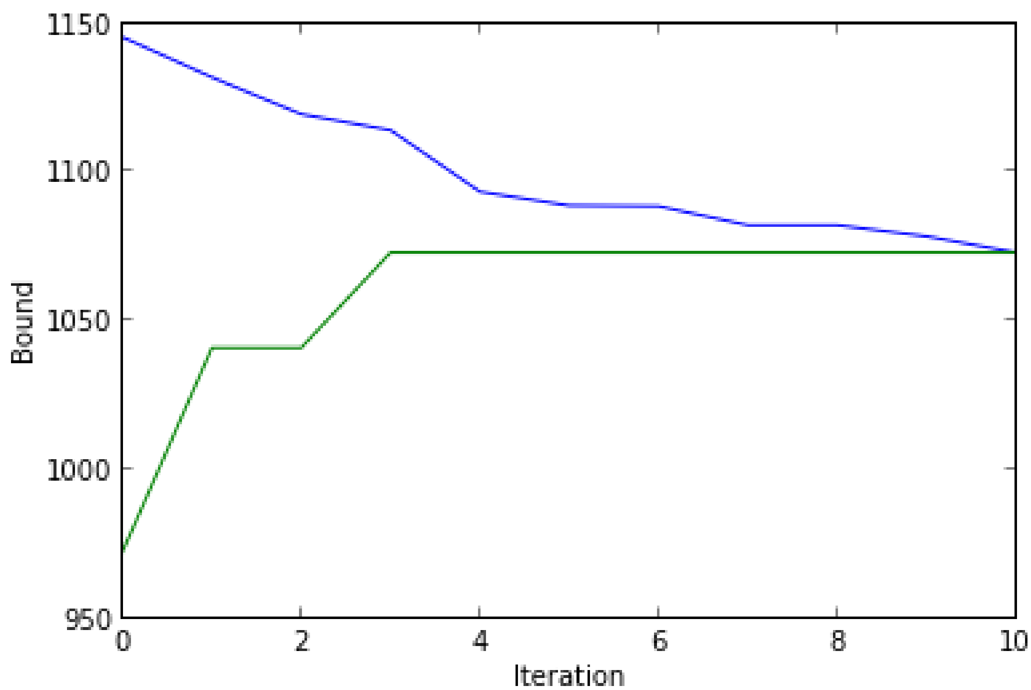
We can also define our own functions using the full power of python. For example,

In [9]:
```
import math

# my crazy sigmoidal function
def crazy_sigmoid(x,param):
    if x > param:
        return -math.pow(x - param,2)
    else:
        return math.pow(x - param,2)

def crazy_sigmoid_prime(x,param):
    if x > param:
        return -2*(x - param)
    else:
        return 2*(x - param)

n = 9
crazy_fs = [(lambda x, p=p: crazy_sigmoid(x,p),
             lambda x, p=p: crazy_sigmoid_prime(x,p),
             p) for p in range(n)]
```

Let's add a couple of equality and inequality constraints, and solve!

In [10]:
```
k,m = 1,2
# inequality constraints Ax <= b
A = numpy.random.normal(0,1,(m,n))
b = numpy.random.uniform(0,2,m)-1
# equality constraints Cx == d
C = numpy.random.normal(0,1,(k,n))
d = numpy.random.uniform(0,2,k)-1
# box constraints
l = [-10]*n
u = [10]*n

p = Problem(l,u,crazy_fs,A=A,b=b,C=C,d=d,name='example2',tol=.1,sub_tol=.(
p.solve(maxiters=20)
f = plot_convergence(p)
f.show()
```

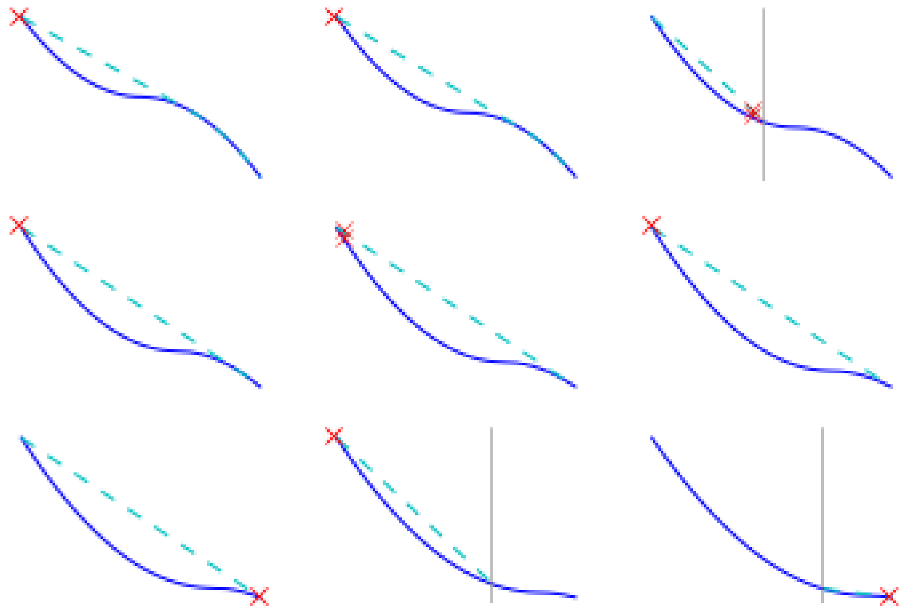
```
Saving plot of history in example2_convergence.eps
```



It looks like we've converged, so we'll take a look at the solution.

In [11]:
```
f = plot_best_node(p)
f.show()
```

```
Saving figure in example2_best_node.eps
```



4 More information

For more examples of how to use SIGOPT, consult the examples in `sigopt.examples`. All functions are documented in their docstrings as well.

To understand how SIGOPT works, and to learn more about the theory of sigmoidal programming, you can take a look at this paper.