Simplex Algorithm in Python

Updated on July 24, 2012

Introduction

The Simplex algorithm is an optimization procedure for linear programs. As implied by "linear", the objective function for such a problem is a linear combination of the decision variables. Additionally, the region of possible solutions (aka "feasible region") is a convex polyhedron.

Considered one of the most important algorithms of all time, I've always found the existing open source implementations rather frustrating to use. When I was in graduate school, we had to use AMPL, a language specifically for modeling optimization problems (which was horrible and taught me little to nothing).

Later on, through my work, I found myself again dealing with linear optimization problems. By this time, however, I was armed with a relatively strong skillset in Python and could avoid using such a domain-specific language like AMPL. Below is a relatively straightforward implementation of the famous algorithm. As an added bonus to those you who might use this for homework problems, you can print the tableau after each pivot.

Simplex in Python

```
from __future__ import division
from numpy import *

class Tableau:

    def __init__(self, obj):
        self.obj = [1] + obj
        self.rows = []
        self.cons = []

    def add_constraint(self, expression, value):
        self.rows.append([0] + expression)
        self.cons.append(value)

    def _pivot_column(self):
        low = 0
        idx = 0
        for i in range(1, len(self.obj)-1):
            if self.obj[i] < low:
                low = self.obj[i]
                idx = i
        if idx == 0: return -1
        return idx

    def _pivot_row(self, col):
        rhs = [self.rows[i][-1] for i in range(len(self.rows))]
        lhs = [self.rows[i][col] for i in range(len(self.rows))]
        ratio = []
        for i in range(len(rhs)):
            if lhs[i] == 0:
                ratio.append(99999999 * abs(max(rhs)))
                continue
            ratio.append(rhs[i]/lhs[i])
        return argmin(ratio)

    def display(self):
        print '\n', matrix([self.obj] + self.rows)

    def _pivot(self, row, col):
        e = self.rows[row][col]
        self.rows[row] /= e
        for r in range(len(self.rows)):
            if r == row: continue
            self.rows[r] = self.rows[r] - self.rows[r][col]*self.rows[row]
        self.obj = self.obj - self.obj[col]*self.rows[row]

    def _check(self):
        if min(self.obj[1:-1]) >= 0: return 1
        return 0

    def solve(self):

        # build full tableau
        for i in range(len(self.rows)):
            self.obj += [0]
            ident = [0 for r in range(len(self.rows))]
            ident[i] = 1
            self.rows[i] += ident + [self.cons[i]]
            self.rows[i] = array(self.rows[i], dtype=float)
        self.obj = array(self.obj + [0], dtype=float)

        # solve
        self.display()
        while not self._check():
            c = self._pivot_column()
            r = self._pivot_row(c)
            self._pivot(r,c)
            print '\npivot column: %s\npivot row: %s'%(c+1,r+2)
            self.display()

```

```
if __name__ == '__main__':

    """
    max z = 2x + 3y + 2z
    st
    2x + y + z <= 4
    x + 2y + z <= 7
    z          <= 5
    x,y,z >= 0
    """

    t = Tableau([-2,-3,-2])
    t.add_constraint([2, 1, 1], 4)
    t.add_constraint([1, 2, 1], 7)
    t.add_constraint([0, 0, 1], 5)
    t.solve()

```

7