Finding an initial Point

Mike Flynn

June 27, 2013

Minimizing distance from the origin

The thought process behind this idea was so: We need to get a point from the inside of a polytope, given the constraint equations. These equations give us a k-plane that will always go through the plane: $x_1 + x_2 + \cdots + x_n = 1$. Since the "triangle" defined by the intersection of this plane and the constraints $x_i > 0$ is already very close to zero, I assumed that any intersection of this plane with another constraint would automatically have it's closest point within the "triangle", however, this is only true if the intersection "goes through the triangle" in the first place, or so it seems.

The derivation is as follows:

The contraint equation $Ax = b$ can be though of geometrically as sequentially restricting x to plane by plane, with each row of A being a plane. It is a basic result of linear algebra that any plane can be represented by the equation $n * x = c$ where n is a vector that is normal to the plane, and c is some constant (i.e. $x + y + z = [1, 1, 1] * [x, y, z] = 1$). This is exactly what the rows of A are, normal vectors.

$$A = \begin{bmatrix} \mathbf{n_1} \\ \mathbf{n_2} \\ \vdots \\ \mathbf{n_n} \end{bmatrix}$$

This will help because the shortest distance between a point and a plane is the perpendicular distance between them, and therefore this distance vector must be in line with a normal vector. For this distance from the origin, this vector must be the position. It follows that:

$$x = \sum_{j=0}^{n} \mathbf{n_j} c_j = A^T c$$

We can now use the two equations for x to solve for it:

$$Ax = b$$
$$x = A^T c$$

Therefore:

$$c = (AA^T)^{-1} b$$

Finally:

$$x = A^T (AA^T)^{-1} b$$

I implemented this code as follows, which works fine in low dimensions but stop's being so good in higher dimensions:

```
find_x0 <- function(A, b) {
    return(t(A) %*% solve(A %*% t(A)) %*% b)
}
A = matrix(1, ncol = 3, nrow = 1)
A = rbind(A, rnorm(3))  # A random constraint, mimicking value
b = A %*% c(0.2, 0.3, 0.5)
x0 = find_x0(A, b)
b

##         [,1]
## [1,]  1.0000
## [2,] -0.4762

A %*% x0

##         [,1]
## [1,]  1.0000
## [2,] -0.4762

x0

##        [,1]
## [1,] 0.2243
## [2,] 0.4048
## [3,] 0.3709


## This stops working nicely for jan
data(jan)
A = matrix(1, ncol = nrow(jan), nrow = 1)
A = rbind(A, jan$value)
A = rbind(A, jan$growth)
b = A %*% jan$portfolio
b
```

```
##         [,1]
## [1,]  1.0000
## [2,]  1.9916
## [3,] -0.2689
```

```
x0 = find_x0(A, b)
length(which(x0 > 0))
```

```
## [1] 2113
```

```
length(which(x0 < 0))
```

```
## [1] 887
```

```
A %*% x0
```

```
##         [,1]
## [1,]  1.0000
## [2,]  1.9916
## [3,] -0.2689
```

As you can see, some of the weights are negative, which we do not want.