CVX

March 17, 2019

0.1 A new algorithm for solving Mixed Integer Non Linear Problems for inventory control

Abstract A new iterative algorithm is developed to solve a MINLP program...

Introduction The optimization problem is an abstraction of the process of choosing the best possible vector $\in R^n$ from a set. In this way it encompasses many ways of decision making, and so the reasons for its ubiquitous relevance becomes clear. The general formulation is:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq b_i, \; i = 1, \ldots, m. \end{aligned}$$

If all $f_i(x)$ fullfil linearity conditions, $f_i(x+y) = f_i(x) + f_i(y)$, then the problem corresponds to a *linear program*. A more general class of problems consist of all that comply $f_i(x+y) f_i(x) + f_i(y)$ given $,\ \in [0,1]$, $+\ = 1$, corresponding to the domain of *convex problems*. Finally, with $x \in \mathbb{Z}$, a *Mixed Integer Convex Non-Linear Program* can be defined.

Problem formulation A stock of a $p \in \mathbb{N}$ products has to be allocated to $s \in \mathbb{N}$ outlets while satisfying arbitrary equality and inequality restrictions. These intend to modulate sales in different areas, allow compliance to comercial agreements, and respond to diverse business needs. Both p and s are classified in multiple hierarchical levels, e.g. : *channel, area, group, delivery route...* for s; *family, product, flavour...* for p. One possiblility for the formulation is to organize this levels into a tensor $Q \in \mathbb{N}^{channel \times area \times ..., family \times product...}$, or tree spanning all levels and categories, but that conveys an extra memory consumption as it assigns space that is redundant, e.g. allocating a quantity of each product for each outlet when not all products are sold at all outlets. Therefore a vector is constructed as $\mathbf{q} = (s_0 p_0, ..., s_i p_j)$, $\mathbf{q} \in \mathbb{R}^{i \times j}$ where $p_i s_j$ corresponds to the quantity of product i in outlet j, and excluding the elements i, j that are not applicable for the period distribution. Continuous relaxation of the q vector is allowed for faster computation. The optimization problem is formulated as:

$$\begin{aligned} \underset{q}{\text{minimize}} \; & |(\mathbf{q} - \mathbf{q_0}) \oslash \mathbf{q_0}|^2, |(\mathbf{q} - \mathbf{q_0}) \oslash \mathbf{q_0}|^\infty \\ \text{subject to: } & \mathbf{Rq} = \mathbf{b} \\ & \mathbf{Mq} \leq \mathbf{d} \end{aligned}$$

Where each row r in $R \in \{0,1\}^{i \times j}$ represents a restriction and is defined as $r_i = 1$ if the corresponding element in $\mathbf{q}$ is in the subset to which the restriction applies. The symbols $\odot$ and $\oslash$ corresponds to element-wise (or Hadamard) product and division, respectively.

Solution 1

```
In [ ]:  while True:
             p = (b / (A @ q))
             idx_ineqs_ok = np.where(p[idx_ineqs] > 1 + 1e-10)
             #0?
             p[idx_ineqs_ok] = 1
             A_= A.multiply(p[:, np.newaxis])
             P = np.squeeze(np.array(np.true_divide(A_.sum(0), (A_!=0).sum(0))))
             q = q * P
             d = np.linalg.norm((b - (A @ q))[[idx for idx in range(A.shape[0]) if idx not i
             if verbose: print(d, end='\t')
             if d >= d_:
                 c += 1
             d_ = d
             if (c == 10) | (d_ < precision):
                 break
```

$$\text{Init } \mathbf{q}^* := q_0$$

$$\text{Repeat until convergence:}$$

$$\mathbf{e} = \mathbf{b} \oslash \mathbf{R} \cdot \mathbf{q}$$

$$\mathbf{R}^* = diag(\mathbf{e}) \cdot \mathbf{R}$$

$$\Delta_j = \frac{1}{i} \sum_{\{i:R*_{ij} \neq 0\}} \mathbf{R}^*_{ij}$$

$$\mathbf{q}^* = \mathbf{q} \odot \Delta$$

Solution 2

$$\mathbf{W} = diag(\mathbf{q}_0)$$

$$\mathbf{s} = Least\ squares[\mathbf{Rq} = \mathbf{b}]$$

$$\mathbf{q}^* = \mathbf{q}_0 - \mathbf{s}$$

$$\mathbf{o} = Orth[(R \cdot W)^T]$$

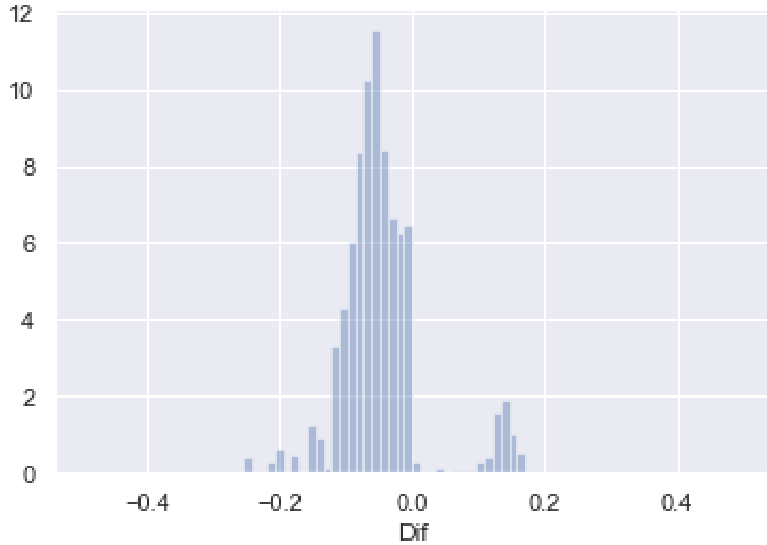
$$\mathbf{v} = \mathbf{W} \cdot o$$

$$\mathbf{x} = \mathbf{o} - \mathbf{v} \cdot (v^T \cdot \mathbf{W}^{-2} \cdot \mathbf{o})$$

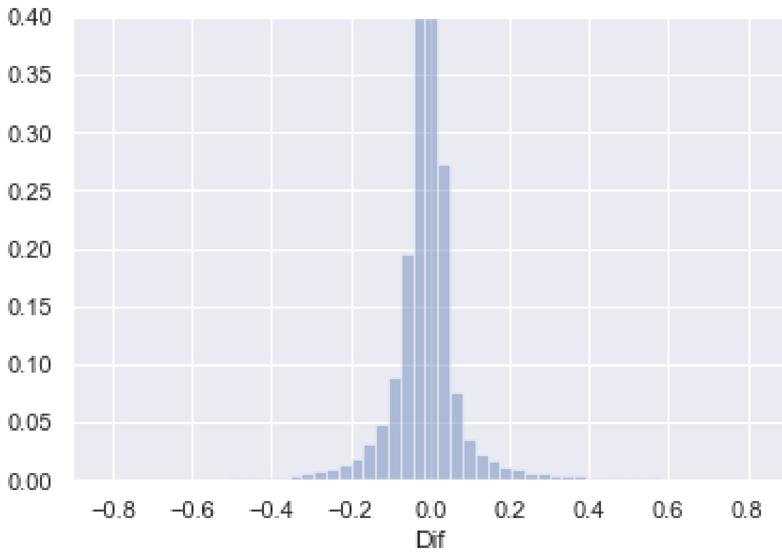
$$\mathbf{q} = x + s$$

For least squares, LSQR algorithm is used (c), and the orthogonal basis o is computed using singular value descomposition.

sparse matrix, lsqr, orth



Results




```
In [ ]:
```


References Belotti, P., Kirches, C., Leyffer, S., Linderoth, J., Luedtke, J., & Mahajan, A. (2013). Mixed-integer nonlinear optimization. Acta Numerica, 22, 1-131.

Boyd, S., & Vandenberghe, L. (2004). Convex optimization. Cambridge university press.

Paige, C. C., & Saunders, M. A. (1982). LSQR: An algorithm for sparse linear equations and sparse least squares. ACM Transactions on Mathematical Software (TOMS), 8(1), 43-71.