Sparse Linear Programming

Martin de La Gorce

April 25, 2016

Abstract

This document describes the linear program solvers algorithms implemented in the associated Python code. Most existing solvers use the simplex or interior point method and do not scale to very large as their complexity is a least quadratic with number of variables [11]. We present some methods to approximatively solve large problems. This are very naive algorithms written by a non-specialist of the linear programming domain and that come with no convergence guaranties.

Contents

1 Introduction

The complexity of state-of-the-art LP solvers (i.e. Interior-Point method and Primal, Dual Simplex methods) is still at least quadratic inthe number of variables or constraints [11]. The quadratic complexity comes from the need to solve each linear system exactly in both simplex and interior point method.

$$\mathbf{x}^* = argmin_x \mathbf{c}^t\mathbf{x} \ s.t. \ A_e\mathbf{x} = \mathbf{b_e}, A_i\mathbf{x} \leq \mathbf{b_i}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \tag{1}$$

2 Generic LP solvers

2.1 Dual Gradient Ascent and Dual Coordinate Ascend

We perform coordinate ascend in the dual.

We first express the Linear program in the slack form

$$\mathbf{x}^* = argmin_x \mathbf{c}^t\mathbf{x} \ s.t. \ A\mathbf{x} = \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \tag{2}$$

this can be rewritten as

$$\mathbf{x}^* = argmin_{\mathbf{x}} max_{\mathbf{y}}, L(\mathbf{x}, \mathbf{y}) \ s.t. \ \ \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \tag{3}$$

with

$$L(\mathbf{x}, \mathbf{y}) = \mathbf{c}^t\mathbf{x} + \mathbf{y}^t(A\mathbf{x} - \mathbf{b}) \tag{4}$$

We can try to solve the dual problem

$$\mathbf{y}^* = argmax_{\mathbf{y}} min_{\mathbf{x}} L(\mathbf{x}, \mathbf{y}) \tag{5}$$

We denote $f(\mathbf{y}) = min_{\mathbf{x}} L(\mathbf{x}, \mathbf{y})$ the dual function, we have $\mathbf{y}^* = argmax_{\mathbf{y}} f(\mathbf{y})$ i.e. we want to maximise f. The function f is concave and piecewise linear and we will maximize it using coordinate ascend.

Given a line search direction $\mathbf{d}$ (along a single coordinate in cas of coordinate ascent, and along the gradient for gradient ascent) we compute the optimal step by solving the 1D problem

$$\alpha^* = argmax_\alpha f(\mathbf{y} + \alpha\mathbf{d}) \tag{6}$$

with $\mathbf{e}_i$ the vector that has zeros everywhere but a one at location i. Then we update $\mathbf{y}$ using $\mathbf{y} \leftarrow \mathbf{y} + \alpha^*\mathbf{d}$. The function $h(\alpha) = f(\mathbf{y} + \alpha\mathbf{d})$ is concave piecewise linear and we can compute its maximum as follows:

$$\bar{\mathbf{c}}(\alpha) = (\mathbf{c}^t + \mathbf{y}^t A + \alpha\mathbf{d}^t A)^t \tag{7}$$

$$h(\alpha) = -\mathbf{y}^t\mathbf{b} - \alpha\mathbf{d}^t\mathbf{b} + \sum_{j,c(\alpha)\bar{[}j]\neq 0} min(\bar{\mathbf{c}}(\alpha)[j]\mathbf{u}[j], \bar{\mathbf{c}}(\alpha)[j]\mathbf{l}[j]) \qquad (8)$$

$\bar{\mathbf{c}}(\alpha)$ is an affine function, $h(\alpha)$ is concave piecewise linear. The derivative of h is piecewise constant decreasing function. Its derivative writes

$$h'(\alpha) = -\mathbf{d}^t\mathbf{b} + \sum_j \mathbf{e}[j](\mathbf{u}[j] + (\mathbf{l}[j] - \mathbf{u}[j])H(\bar{\mathbf{c}}(\alpha)[j])) \qquad (9)$$

with $\mathbf{e} = \mathbf{d}^t A$ with H the heaviside step function Discontinuities of the derivative occurs when one component of $\bar{\mathbf{c}}(\alpha)$ changes sign. Let $s = \{j|\mathbf{e}_j \neq 0\}$ for all j in S. Let $\mathbf{v} = \mathbf{c} + \mathbf{y}^t A$. The sign of $\bar{\mathbf{c}}(\alpha)[j]$ changes when $\mathbf{v}[j] + \alpha\mathbf{e}[j] = 0$ i.e. When $\alpha = -\mathbf{v}[j]/\mathbf{e}[j]$ let $\mathbf{a}$ be the vector with $\mathbf{a}[j] = -\mathbf{v}[j]/\mathbf{e}[j]$

$$h'(\alpha) = -\mathbf{d}^t\mathbf{b} + \sum_{j,\mathbf{e}[j]>0} \mathbf{e}[j]\mathbf{u}[j] + \sum_{j,\mathbf{e}[j]<0} \mathbf{e}[j]\mathbf{l}[j] \sum_{j,a[j]<\alpha} |\mathbf{e}[j]|(\mathbf{l}[j] - \mathbf{u}[j]) \quad (10)$$

$$h'(\alpha) = -\mathbf{d}^t\mathbf{b} + \sum_{j,\mathbf{a}[j]<\alpha} min(\mathbf{e}[j]\mathbf{u}[j], \mathbf{e}[j]\mathbf{l}[j]) + \sum_{j,\mathbf{a}[j]>\alpha} max(\mathbf{e}[j]\mathbf{l}[j], \mathbf{e}[j]\mathbf{u}[j]) \qquad (11)$$

The function $h'(\alpha)$ is decreasing piecewise constant and can be efficiently computed by sorting the entries in $\mathbf{a}$ and performing left and right cumulative sums. We then look for α^* such that $h'(\alpha^*) = 0$. If there is a interval for which $h'(\alpha) = 0$ then we choose α^* by sampling a uniform distribution in that interval or by choosing the middle value.

The coordinate ascent method generalizes easily to the inequality form of the LP by enforcing positive of the dual variables associated to inequalities after each update. The gradient ascent formulation is hard to apply to the inequality form.

In comparison with dual gradient ascent, the dual coordinate ascent method may seem to have some advantages:

- invariant by individual rescaling of the rows of the constraints matrix A along with the corresponding entry in b
- no step parameter
- the computational cost of computing the optimal step length depends on the number of non zeros in $\mathbf{e}$, which is small when we use coordinate ascent and the matrix A is sparse

- we could update coordinates that are the most promising and test less often dual variables associate to inactive constraints.
- there is an explicit objective function beeing optimized and that can be use to compare solutions.

However it seems that the method gets often stuck due to the non differentiability of the dual function.

If we use a decreasing step length schedule with the dual gradient ascent, it does not seem to converge well either, not sure why. Maybe we could do some smoothing of the dual function by adding a small quadratic penalization $\|\mathbf{x}\|^2$ to $L(\mathbf{x}, \mathbf{y})$ with a weight that is decreased through iterations ? The optimal step computation might be a bit more difficult then? maybe instead of taking the optimal step , we can take a random step length in the interval that increases the dual.

2.2 Chambolle-Pock preconditioned primal-dual algorithm

We adapt the algorithm for solving linear program presented in [12] to a more generic formulation of linear programs

$$\mathbf{x}^* = argmin_{\mathbf{x}} \mathbf{c}^t\mathbf{x} \; s.t. \; A_e\mathbf{x} = \mathbf{b_e}, A_i\mathbf{x} \leq \mathbf{b_i}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \tag{12}$$

this can be rewritten as

$$\mathbf{x}^* = argmin_{\mathbf{x}} max_{\mathbf{y_e},\mathbf{y_i}} \mathbf{c}^t\mathbf{x} + \mathbf{y}_e^t(A_e\mathbf{x} - \mathbf{b}_e) + \mathbf{y}_i^t(A_i\mathbf{x} - \mathbf{b}_i) \; s.t. \; \mathbf{y}_i \geq 0, \; \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \tag{13}$$

adapting the algorithm in [12] we get

$$\begin{aligned}
\mathbf{x}^{k+1} &= proj_{[\mathbf{l},\mathbf{u}]}(\mathbf{x}^k - T(A_i^T\mathbf{y}_i^k + A_e^T\mathbf{y}_e^k + \mathbf{c})) && (14)\\
\mathbf{z}^k &= (1+\theta)\mathbf{x}^{k+1} - \theta\mathbf{x}^k && (15)\\
\mathbf{y}_e^{k+1} &= \mathbf{y}_e^k + \Sigma_e(A_e\mathbf{z}^k - \mathbf{b}_e) && (16)\\
\mathbf{y}_i^{k+1} &= proj_{[0,+\infty)}(\mathbf{y}_i^k + \Sigma_i(A_i\mathbf{z}^k - \mathbf{b}_i)) && (17)
\end{aligned}$$

where $\theta \in [0, 1]$ is an over-relaxation parameter and with the preconditioning diagonal matrices T,Σ_i and Σ_e defined by

$$\begin{aligned}
T[j,j] &= \big(\sum_i |A_e[i,j]|^{2-\alpha} + \sum_i A_i[j,j]^{2-\alpha}\big)^{-1} && (18)\\
\Sigma_i[i,i] &= \big(\sum_j |A_i[i,j]|^{\alpha}\big)^{-1} && (19)\\
\Sigma_e[i,i] &= \big(\sum_j |A_e[i,j]|^{\alpha}\big)^{-1} && (20)
\end{aligned}$$

with $\alpha \in [0, 2]$.

Some remarks : if a variables x_i is not part of any constraint other than the simple bound constraints then $T[i, i]$ is equal to ∞. This would be ok if T what not multiplied by c in the update of $\mathbf{x}$. It is a bit strange that $\mathbf{c}$ is also multiplied by T as it make the method not invariant by rescaling of $\mathbf{c}$. Similarly the update of $\mathbf{x}$ is not invariant by rescaling individual rows of A_i or A_e while the ones of y_e and y_i are if we take $\alpha = 1$. Maybe we should try to rescale each row of A_i, A_e and normalize $\mathbf{c}$?

2.2.1 accelerations

Some variable can be estimated quite reliably after a few iteration and do not change in the subsequent iterations and it seems to be a waste of computational power to evaluate the update on these variables at each iteration. We would be temped to keep these variables fixed for some iterations and same some computation in during these iterations. Also some of the constraint will keep being verified by these fixed variables and it is a waste of computational power to re-evaluate them at each iteration. Can we use sparse vectors and maybe add some auxiliary variables to avoid recalculation of values that do not change

2.3 Alternating Direction Method of Multipliers

2.4 Augmenting the positivity constraints

We first follow the method presented in section 5.2 in [3]. Xe reformulate the LP in the standard form

$$\underset{\mathbf{x}}{min}\, \mathbf{c}^t\mathbf{x} \text{ s.t. } A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0 \tag{21}$$

In order to deal with the constraints $A\mathbf{x} = \mathbf{b}$ and $\mathbf{x} \geq 0$ separatly we introduce copies of $\mathbf{x}$:

$$\underset{\mathbf{x}}{min}\, \mathbf{c}^t\mathbf{x} \text{ s.t. } A\mathbf{x} = \mathbf{b}, \mathbf{y} \geq 0, \mathbf{x} = \mathbf{y} \tag{22}$$

We introduce mutlipliers λ for the constraints $\mathbf{x} = \mathbf{y}$

$$\underset{\mathbf{x}}{min}\, \underset{\lambda}{max}\, \mathbf{c}^t\mathbf{x} + \lambda^t(\mathbf{x} - \mathbf{y}) + \gamma\|\mathbf{x} - \mathbf{y}\|^2 \text{ s.t } A\mathbf{x} = \mathbf{b}, \mathbf{y} \geq 0 \tag{23}$$

Using indicator function [.] that takes 0 is the condition in the bracket is true and ∞ else we can rewrite it as

$$\underset{\mathbf{x}}{min}\,\underset{\lambda}{max}\,\mathbf{c}^t\mathbf{x} + \lambda^t(\mathbf{x}-\mathbf{y}) + \gamma\|\mathbf{x}-\mathbf{y}\|^2 + [A\mathbf{x}=\mathbf{b}] + [\mathbf{y}\geq 0] \tag{24}$$

We can now use the Alternating Direction Method of Multipliers (ADMM) method. We minimize with respect to $\mathbf{x}$ that is quadratic under linear constraints,then with respect to $\mathbf{y}$ that is easy, then update the dual variables.

$$\begin{aligned}
\mathbf{x}_{t+1} &= argmin_{\mathbf{x}}\mathbf{c}^t\mathbf{x} + \lambda_t^T(\mathbf{x}-\mathbf{y}_t) + \gamma\|\mathbf{x}-\mathbf{y}_t\|^2 \text{ s.t } A\mathbf{x}=\mathbf{b} && (25)\\
\mathbf{y}_{t+1} &= proj_{[0,\infty)}(\mathbf{x}_{t+1} + \lambda/\gamma) && (26)\\
\lambda_{t+1} &= \lambda_{t+1} + \gamma(\mathbf{x}_{t+1} - \mathbf{y}_{t+1}) && (27)
\end{aligned}$$

we obtain $\mathbf{x}_{t+1}$ by solving the linear equations system

$$\begin{bmatrix} 2\gamma I_d & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_{t+1} \\ v \end{bmatrix} = \begin{bmatrix} -c^t - \lambda_t^T + 2\gamma\mathbf{y}_t \\ b \end{bmatrix} \tag{28}$$

with v lagrange multipliers. As we solve several linear problems with the same matrix, we may try to reuse some factorization (cholesky) but this problem may be too big to solve directly and we may have to use iterative solvers such as conjgate gradient. We could reuse the previous solution as a warm start (section 4.3.3 in [3]), we could try to compute a good preconditionner that will be reuse across problems (see discussion in section 4.2 in [3]) or reuse history of previous linear system resolutions to get a preconditionner. If we do not solve the linear system accuratly we may may loose convergence guaranties (this is briefly discussed in sections 3.4.4 and 4.3.2 in [3]).

2.5 Augmenting the linear equalities

unlike done in [3], we can also introduce mutlipliers for the linear equality constraints $A\mathbf{x}=\mathbf{b}$ with augmented lagrangian terms:

$$\underset{\mathbf{x}}{min}\,\underset{\lambda_1,\lambda_2}{max}\,\mathbf{b}^t\mathbf{x} + \lambda_1^t(A\mathbf{x}-\mathbf{b}) + \gamma_1\|A\mathbf{x}-\mathbf{b}\|^2 + \lambda_2^t(\mathbf{x}-\mathbf{y}) + \gamma_2\|\mathbf{x}-\mathbf{y}\|^2 \text{ s.t } \mathbf{y}\geq 0 \tag{29}$$

Using this formulation the update of $\mathbf{x}$ is an unconstrained quadratic minimization problem which can be solved by solving a simple linear system

However, in constract with the method in the previous section, approximate solution do not violate any constraints. This may make the method more tolerant with approximate resolution of the linear system when updating $\mathbf{x}$ as we could simply check sufficient decrease of the quadratic energy.

We can again use the Alternating Direction Method of Multipliers (ADMM). We minimize with respect to $\mathbf{x}$ that is quadratic but this time without linear constraints, then with respect to $\mathbf{y}$, which is easy, then update the dual variables.

When minimizing with respect to $\mathbf{x}$ we use n iteration of coordinate descent or other gradient descent with lambda fixed, or conjugate gradient, or diagonal quasi newton BFGS. Like with the method in the previous section we solve many similar linear systems (on at each iteration) and we could try to reuse computation across iterations.

Some questions:

- if we do one step of gradient descent when minimizing with respect to $\mathbf{x}$, is it then equivalent to chambolle-pock ?
- can we update γ_1 and γ_2 trough iteration? have a look at section 3.4.1 in [3] for a method of adaptative penalty parameter.
- is there a way to make γ_2 dependent of the distance to the constraint at each iteration (null if we are far enough) without making it infinite if we are close to the constraint ? update lambda with on step in the gradient direction and loop again. once we modified λ we could use the accumulated quasi newton inverse matrix ? with $\lambda = 0$ this is equivalent to the fixed penalization method

3 Exploiting the structure

the previous solvers do not exploit much of the structure of the problem. They exploit the sparsity but not the structure(block , toepliz blocks etc). By duplicating some variables using a dual decomposition method or an augmented Lagrangian method, we can obtain sub-problems that can be solve easily: for example linear system with tridiagonal matrices that can be solved easily in linear time using the Tridiagonal matrix algorithm a.k.a as the Thomas algorithm.

3.1 ADMM with vertical matrix decomposition

3.1.1 decomposition

We adapte the *Global Variable Consensus With regularization* method presented in section 7.1 in [3] to the linear program written in the standard form. Suppose the problem writes

$$min c^t x \text{ s.t.} Ax = b, x \geq 0 \tag{30}$$

we can decompose A horizontally into N submatrices $A = \begin{bmatrix} A_1 \\ \vdots \\ A_n \end{bmatrix}$ and the vector b into subvectors $b_1, \ldots, b_n$ with length of each subvector b_i matching the height of the submatrices A_i

$$min \frac{1}{N} \sum_{i=1}^{N} c^t x_i \text{ s.t.} \forall i : A_i x_i = b_i, \forall i > j : x_i = y, y \geq 0 \tag{31}$$

we can then enforce the equality constraints $x_i = y$ using an augmented lagrange formulation.

$$\begin{aligned} x_i^{t+1} &= argmin_x \frac{1}{N} c^t x + \lambda_i^{tT}(x - y^t) + \gamma \|x - y^t\|^2 \text{ s.t } A_i x = b_i && (32) \\ y^{t+1} &= proj_{[0,\infty)}(\frac{1}{N} \sum_i x_i^{t+1} - \gamma^{-1} \sum_i \lambda_i^{tT}) && (33) \\ \lambda_i^{t+1} &= \lambda_i^{t+1} + \gamma(x_i^{t+1} - y^{t+1}) && (34) \end{aligned}$$

It is sufficient to enforce equality only on the subset of coordinates of x_i for which columns in A_i are not empty , in order to avoid unnecessary inertia when updating y , which mean that parts of x_i are not used (if we see each constraint $A_i x_i = b_i$ as a factor we need to copy only the variable used by the factor). This is exactly what is done in the *Generalized Form Consensus Optimization* method presenting in section 7.2 in [3]. We use a slightly different formalization than [3] by introducing rectangular sparse matrices S_i with a single one on each colmuns and rows, where S_i that extract the component in x_i hat corresponds to non empty columns in A_i and we denote $z_i = S_i x_i$. We rewrite the minimization problem as

$$min \frac{1}{N} \sum_{i=1}^{N} c^t S_i^T z_i \text{ s.t.} \forall i : A_i S_i^T z_i = b_i, \forall i > j : z_i = S_i y, y \geq 0 \tag{35}$$

$$\begin{aligned} z_i^{t+1} &= argmin_z \frac{1}{N} c^t S_i^T z + \lambda_i^{tT}(z - S_i y^t) + \gamma \|z - S_i y^t\|^2 \text{ s.t } A_i S_i^T z = b_i && (36) \\ y^{t+1} &= proj_{[0,\infty)}(D \sum_i S_i^T z_i^{t+1} - \gamma^{-1} \sum_i \lambda_i^{tT}) && (37) \\ \lambda_i^{t+1} &= \lambda_i^{t+1} + \gamma(z_i^{t+1} - S_i y^{t+1}) && (38) \end{aligned}$$

with D a diagonal matrix defined by $(\sum_i S_i^T S_i)^{-1}$ whose k^{th} diagonal element is in the inverse of the number of time the k^{th} element of y as been copied in some z_i.

3.1.2 solving subproblems

At each iteration we need to solve N linear system of the form:

When we update z_i we need to solve a linear systems with matrix $A_i^t A_i$

- $A_i^t A_i$ small ((single scalar if A_i a a single row matrix,)
- $A_i^t A_i$ of size n by n and band limited with width k: we can use a Cholesky decomposition (computation is suppose posed to be in nk). If the system is a tridiagonal system it can be solved in linear time using Thomas algorithm, which is equivalent to Gaussian elimination.
- iterative solvers : conjugate gradient , jacobi, Gauss-Siedel , Sequential over relaxation.

the inequality constraint $x \geq 0$ is easily impose on x by re-projection on the positive quartant ? is it similar to the method used by loic ? the advantage is that we do not an iterative solver for the subproblems. there might be some link between the fact we can use Thomas algorithm for the subproblems involving tridiogonal matrices , and dynamic programming if we remove the quadratic terms and add the inequality constraints on each x_i Is there also a link for the combinatorial problem that can be solved using shortest patch in graph and some structure in the matrices ?

can we solve the 2D Poisson equation by decomposing into 1D horizontal and 1D vertical problems using ADMM ?

4 Integer constraints

A heuristic proposed in [15, 14] that is designed for problem where the matrix A contain only binary entries consists in doing dual coordinate ascent and adding small perturbation to the problem by modifying the cost vector c in order to obtain feasible integer solution in the primal. The method has the advantage that is does not require to solve the LP relaxation exactly to run a heuristic to get a integer feasible solution , and can find an feasible solution in a small amount of computation that needed to solve the relaxed LP. The paper [2] generalizes this method to more general matrices A with non integer and non positive entries. However it does seem to consist in a simple dual coordinate ascent uses some knapsack solver to solve sub-problems (a bit difficulat to read :( )

References

[1] P Aguiar and EP Xing. An augmented Lagrangian approach to constrained MAP inference. *... of the 28th ...*, 2011.

[2] Oliver Bastert, Benjamin Hummel, and Sven De Vries. Generalized Wedelin heuristic for integer programming. *INFORMS Journal on Computing*, 22(1):93–107, 2010.

[3] Stephen Boyd. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2010.

[4] Laurent Condat. A direct algorithm for 1-D total variation denoising. *IEEE Signal Processing Letters*, 20:1054–1057, 2013.

[5] Yu G Evtushenko, A I Golikov, and N Mollaverdy. Augmented Lagrangian method for large-scale linear programming problems. 20(October):1–10, 2005.

[6] Jacek Gondzio and Robert Sarkissian. Parallel interior-point solver for structured linear programs. *Mathematical Programming*, 584(January):561–584, 2003.

[7] Andreas Grothey. Exploiting Structure in Parallel Implementation of Interior Point Methods for Optimization 1. pages 1–25, 2007.

[8] Nachi Gupta and Raphael Hauser. Kalman Filtering with Equality and Inequality State Constraints. Technical Report 07, 2007.

[9] Geo Irving, Keith Pasko, and Martin Wicke. A linear time direct solver for convex tridiagonal quadratic programs with bound constraints. pages 1–6, 2015.

[10] Seung-jean Kim, K Koh, M Lustig, Stephen Boyd, and Dimitry Gorinevsky. An Interior-Point Method for Large Scale l1-Regularized Least Squares. *IEEE Journal of selected topics in signal processing*, 1(4):606–617, 2007.

[11] Jorge Nocedal and Steve J. Wright. *Numerical optimization.* Springer Series in Operations Research and Financial Engineering. Springer, Berlin, 2006. NEOS guide http://www-fp.mcs.anl.gov/otc/Guide/.

[12] T Pock and Antonin Chambolle. Diagonal preconditioning for first order primal-dual algorithms in convex optimization. *Computer Vision (ICCV), 2011 IEEE . . .*, 2011.

[13] Francisco Tom. Modular proximal optimization for multidimensional total-variation regularization . 2013.

[14] D Wedelin. An Algorithm for large scale 0-1 Integer Programming with Application to Airline Crew Scheduling. *Annals of Operations Research*, 57(December):283–301, 1995.

[15] Dag Wedelin. Revisiting the in-the-middle algorithm and heuristic for integer programming and the max-sum problem. (November):1–22, 2013.