Solving LP using random projections

Leo Liberti, Pierre-Louis Poirion, Vu Khac Ky [1,2]

CNRS LIX, Ecole Polytechnique, 91128 Palaiseau, France

Abstract

A celebrated result of Johnson and Lindenstrauss asserts that, in high enough dimensional spaces, Euclidean distances defined by a finite set of points are approximately preserved when these points are projected to a certain lower dimensional space. We show that the distance from a point to a convex set is another approximate invariant, and leverage this result to approximately solve linear programs with a logarithmic number of rows.

Keywords: Johnson-Lindenstrauss lemma, random projection.

1 Introduction

One of the computational "grand challenges" in Mathematical Programming is to solve ever larger Linear Programs (LP). We are currently able to routinely solve (sparse) LPs with a million variables and constraints. Developers of commercial solvers have seen customer LPs with up to a hundred million variables. What about a billion? This short paper is unfortunately *not* announcing such a breakthrough, but it possibly paves the way — if one is willing to accept an approximate solution with high probability.

[1] Email: {`liberti,poirion,vu`}`@lix.polytechnique.fr`

[2] Vu Khac Ky is supported by a Microsoft Research PhD grant.

We want to find approximate solution of LPs in standard form

$$\min\{cx \mid Ax = b \wedge x \geq 0\}, \tag{1}$$

with high probability, where A is an $m \times n$ matrix, $c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$. The general idea is as follows: we pre-multiply A and b by a certain $k \times m$ matrix T (sampled randomly from certain distributions), with $k \ll m$. T is guaranteed with high probability to approximately preserve Euclidean distances among the columns of A and b. Since the worst-case complexity LP methods depends on both n and m, a large decrease in the number of rows is likely to have a beneficial impact on efficiency, and to allow for solving larger instances.

Such *random projection* methods are at the heart of the proof of the Johnson-Lindenstrauss Lemma (JLL), which states that, for any finite set $X \subseteq \mathbb{R}^m$ with $|X| = n$ and $\varepsilon \in (0,1)$ there exists a k of order $O(\frac{1}{\varepsilon^2}\ln n)$ and a mapping $T : \mathbb{R}^m \to \mathbb{R}^k$ such that:

$$\forall x, y \in X \quad (1-\varepsilon)\|x-y\|_2 \leq \|Tx - Ty\|_2 \leq (1+\varepsilon)\|x-y\|_2. \tag{2}$$

From here onwards, norms will always be Euclidean unless specified otherwise.

Random projections have been used previously to address optimization and/or learning algorithms involving the Euclidean norm only (see e.g. [2,1]). This is their natural setting, since a set of Euclidean distances is rotationally independent and rotational independence plays a prominent role in the original proof in the JLL [3]. As far as we know, this is the first application of the approximate preservation of the orthant $x \geq 0$ (which is definitely *not* rotationally independent), and is therefore interesting in its own right from a theoretical point of view.

For a matrix A we denote the i-th row by A_i and the j-th column by A^j. For a vector v and an index set J, we let $v_J = (v_j \mid j \in J)$. Let $\mathscr{C}(A) = \mathsf{cone}(A^j \mid j \leq n)$. For a problem P let $\mathcal{F}(P)$ be its feasible region.

2 A randomized algorithm for large LPs

Our proposed algorithm is as follows.

1. **Sample** a $k \times m$ *random projector* matrix T.
2. **Solve** $TP \equiv \min\{cx \mid TAx = Tb \wedge x \geq 0\}$,
   let c' be its optimal objective function value.
3. **Retrieve** an approximately optimal solution x^* of P as follows:
   a. let $A'x = b'$ be the system $TAx = Tb \wedge cx = c'$,

let α be a uniform random vector in $\mathbb{R}^n$;

b. solve $TP_\alpha \equiv \min\{\alpha x \mid A'x = b' \wedge x \geq 0\}$,
let y' be its optimal dual vector and $y = T^\top y'$;

c. let J be the set of indices $j \leq n$ such that $yA^j = \alpha_j$,
set $x_i^* = 0$ for each $j \notin J$;

d. let $\bar{x}$ be the solution of the $k \times k$ system $(A^J)^\top A^J x_J = (A^J)^\top b$,
let $x_j^* = \bar{x}_j$ for each $j \in J$.

In the rest of this paper, we shall sketch the reason why this algorithm works.

3 The random projector

Among the many distributions that T can be sampled from, the simplest has each component of T sampled independently from $\mathcal{N}(0, \frac{1}{\sqrt{k}})$. Since T is a linear map, it obviously preserves feasibility. In the (yet unpublished) report [4], we prove that, if b, A^j are unit vectors for $j \leq n$ and $b \notin \mathscr{C}(A)$, then $\exists \mathcal{C} > 0$ such that:

$$\mathsf{Prob}(Tb \notin \mathscr{C}(TA)) \geq 1 - 2n(n+1)e^{-\mathcal{C}(\varepsilon^2 - \varepsilon^3)k}$$

for all $\varepsilon > 0$ in a certain "reasonable" interval. Since $b \in \mathscr{C}(A)$ iff $\exists x \geq 0$ s.t. $Ax = b$, our result shows that if P is infeasible then TP highly likely to be infeasible, and this probability can be made arbitrarily close to 1 as k grows. [3]

4 Solving the projected LP

Since $\mathcal{F}(P) = \mathcal{F}(TP)$ with high probability, a bisection argument shows that P and TP both have objective function values c' with high probability. Thus, we can find c' by simply solving TP using a standard LP solver. On the other hand, we can prove that the primal solution x' of TP is infeasible in P with probability 1, so we need a different strategy to compute the certificate.

5 Solution retrieval

Steps a-d in the algorithm of Sect. 2 provide a primal solution retrieval method via the dual LP using complementary slackness. The dual y' of P_α is such that $y'A' \leq \alpha$. Since $A' = (TAc)^\top$, we write $y' = (\bar{y}, y^c)$ so that we have $\bar{y}TA + y^c c \leq \alpha$ $(\star)$. Letting $y = (\bar{y}T, y^c)$ we have $y(Ac)^\top \leq \alpha$ $(\dagger)$, which

[3] I.e. as m grows, which, since P is in standard form, also means that n grows.

means that y is a valid dual solution to the problem $P_\alpha = \min\{\alpha x \mid Ax = b \wedge cx = c' \wedge x \geq 0\}$. By complementary slackness of TP_α, at least k of the n inequalities in ($\star$) are satisfied at equality (say those corresponding to the index set J), which means the same holds for (†). By complementary slackness of P_α, $\forall j \notin J$ we have $x^*_j = 0$. The nonzero components of x^* are those indexed by J, and we can find them by identifying the corresponding k columns of $Ax = b$ and then solving a $k \times k$ linear system.

6 Perspectives

So, how far are we down the road to solving large LPs? If we only consider dense, randomly generated feasibility problems $Ax = b \wedge x \geq 0$, the following table shows that this approach does actually save us some time.

Uniform	ε	$k \approx$	CPU savings	accuracy
$(0,1)$	0.1	$0.5m$	10%	100%
$(0,1)$	0.15	$0.25m$	90%	100%
$(0,1)$	0.2	$0.12m$	97%	100%
$(-1,1)$	0.1	$0.5m$	30%	50%
$(-1,1)$	0.15	$0.25m$	92%	0%
$(-1,1)$	0.2	$0.12m$	99.2%	0%

For sparse LPs, as expected, the issues concerning size, values of the constant $\mathcal{C}$, and values of ε (none of which we know how to estimate, much less compute) make it impossible to obtain any CPU time saving. For validation purposes, we ran a simple experiment on the `afiro` and `recipe` instances of the NetLib [5], and obtained a valid objective function value and primal solutions in around 10% and 20% of the total number of independent runs of our randomized algorithm.

References

[1] Ailon, N. and B. Chazelle, *Approximate nearest neighbors and fast Johnson-Lindenstrauss lemma*, in: *Proceedings of the Symposium on the Theory Of Computing*, STOC **06** (2006).

[2] Indyk, P. and A. Naor, *Nearest neighbor preserving embeddings*, ACM Transactions on Algorithms **3** (2007), p. Art. 31.

[3] Johnson, W. and J. Lindenstrauss, *Extensions of Lipschitz mappings into a Hilbert space*, in: G. Hedlund, editor, *Conference in Modern Analysis and Probability*, Contemporary Mathematics **26** (1984), pp. 189–206.

[4] Ky, V. K., P.-L. Poirion and L. Liberti, *Using the johnson-lindenstrauss lemma in linear and integer programming*, Technical Report 1507.00990v1[math.OC], arXiv (2015).

[5] NetLib, *LP instance library* (2015), `http://www.netlib.org/lp/`.