Barvinok's naive algorithm for dimensionality reduction

and its use in Distance Geometry

Leo Liberti, CNRS LIX Ecole Polytechnique

`liberti@lix.polytechnique.fr`

TAU18 Workshop 180424-26

Joint work with Vu Khac Ky

Outline

Barvinok's Naive Algorithm

Distance Geometry

Noisy distances

Computational results

Concentration of measure

From [Barvinok, 1997]

> *The value of a "well behaved" function at a random point of a "big" probability space X is "very close" to the mean value of the function.*

and

> *In a sense, measure concentration can be considered as an extension of the law of large numbers.*

Concentration of measure

Given Lipschitz function $f : X \to \mathbb{R}$ s.t.

$$\forall x, y \in X \quad |f(x) - f(y)| \leq L\|x - y\|_2$$

for some $L \geq 0$, there is *concentration of measure* if $\exists$ constants c, C s.t.

$$\forall \varepsilon > 0 \quad \mathsf{P}_x(|f(x) - \mathsf{E}(f)| > \varepsilon) \leq c\, e^{-C\varepsilon^2/L^2}$$

where $\mathsf{E}(\cdot)$ is w.r.t. given Borel measure μ over X

$\equiv$ *"discrepancy from mean is unlikely"*

Barvinok's theorem

Consider:

- **for each** $k \leq m$, **manifolds** $\mathcal{X}_k = \{x \in \mathbb{R}^n \mid x^\top Q^k x = a_k\}$
  where $m \leq \mathsf{poly}(n)$
- **feasibility problem** $F \equiv \left[\bigcap_{k \leq m} \mathcal{X}_k \overset{?}{\neq} \varnothing\right]$
- **SDP relaxation** $\forall x \leq m \; (Q^k \bullet X = a_k) \wedge X \succeq 0$ **with soln.** $\bar{X}$

Find an approximate rank-1 solution of F

Algorithm: $T \leftarrow \mathsf{factor}(\bar{X})$; $\quad y \sim \mathcal{N}^n(0,1)$; $\quad x' \leftarrow Ty$

Then $\exists c > 0, n_0 \in \mathbb{N}$ **such that** $\forall n \geq n_0$

$$\mathsf{Prob}\left(\forall k \leq m \quad \mathsf{dist}(x', \mathcal{X}_k) \leq c\sqrt{\|\bar{X}\|_2 \ln n}\right) \geq 0.9.$$

IDEA: since x' is "close" to each $\mathcal{X}_k$, try local descent!

Elements of Barvinok's formula

$$\textsf{Prob}\left(\forall k \le m \quad \textsf{dist}(x', \mathcal{X}_k) \le c\sqrt{\|\bar{X}\|_2 \ln n}\right) \ge 0.9.$$

- $\sqrt{\|\bar{X}\|_2}$ **arises from** T **(a factor of** $\bar{X}$**)**
- $\sqrt{\ln n}$ **ensures concentration of measure**
- 0.9 **follows by adjusting parameter values in union bounds**

Outline

Distance Geometry Problem

- **Given** $K \in \mathbb{N}_+$ **and** $G = (V, E, d)$ **with** $d : V \to \mathbb{Q}_+$, **determine if** $\exists$ **realization** $x : V \to \mathbb{R}^K$ **s.t.:**

$$\forall \{i, j\} \in E \quad \|x_i - x_j\|_2^2 = d_{ij}^2$$

- Inverse problem to: ***given** n **pts in** $\mathbb{R}^K$**, determine some of their pairwise distances and their adjacencies***
- $\Rightarrow$ **"Draw graph in $\mathbb{R}^K$, where edges $\equiv$ segments of corresp. length"**
- **Applications:** clock synchronization protocols, sensor network localization, protein conformation, nanostructures, autonomous underwater vehicles, rigidity, statics and more
- *Strongly **NP**-hard even if K fixed and* $\mathsf{ran}(d) = \{1, 2\}$

See e.g. [L. et al, SIAM Review 2014], [Dokmanić et al., IEEE Sig. Proc. Mag. 2015], [L. and Lavor, Springer 2017]

SDP formulation of the DGP

- **A feasibility problem:**

$$\begin{array}{rrcl} & \min F \bullet X & & \\ \forall \{i,j\} \in E & X_{ii} + X_{jj} - 2X_{ij} & = & d_{ij}^2 \\ & X & \succeq & 0 \end{array}$$

- F **depends on application; for protein conformation, try:**

$$F = \sum_{\{i,j\} \in E} (X_{ii} + X_{jj} - 2X_{ij}) + \nu \mathsf{Tr}(X)$$

where ν is small (try 0.001)

- **Issue: obtain realization $\bar{X}$ in $\mathbb{R}^n$ not $\mathbb{R}^K$**
  need dimension reduction

Barvinok's alg. for the DGP

- $\forall \{i,j\} \in E \quad \mathcal{X}_{ij} = \{x \in \mathbb{R}^{nK} \mid \|x_i - x_j\|_2^2 = d_{ij}^2\}$
- **DGP $\equiv$ "is $\bigcap_{\{i,j\}\in E} \mathcal{X}_{ij}$ non-empty?"**
- **SDP rel.** $X_{ii} + X_{jj} - 2X_{ij} = d_{ij}^2 \wedge X \succeq 0$ **with soln.** $\bar{X}$

- **Difference w.r.t. Barvinok:** $x \in \mathbb{R}^{nK}$
- **IDEA: sample** $y \sim \mathcal{N}^{nK}(0, \frac{1}{\sqrt{K}})$
- **Our result:** Extension of Barvinok's thm to rank K

$$\mathsf{Prob}\left(\forall k \le m \quad \mathsf{dist}(x', \mathcal{X}_k) \le c\sqrt{\|\bar{X}\|_2 \ln(nK)}\right) \ge 0.9.$$

Analysis improvement yields

$$\mathsf{Prob}\left(\forall k \le m \quad \mathsf{dist}(x', \mathcal{X}_k) \le c\sqrt{\|\bar{X}\|_2 \ln n}\right) \ge 0.9.$$

Proof structure

- Show that, on average, $\forall k \le m\ (Ty)^\top Q^k (Ty) = Q^K \bullet \bar{X} = a_k$
  - compute multivariate integrals
  - bilinear terms disappear because y normally distributed
  - decompose multivariate int. to a sum of univariate int.
- Exploit concentration of measure to show errors happen rarely
  - a couple of technical lemmata yielding bounds
  - $\Rightarrow$ bound Gaussian measure μ of ε-neighbourhoods of

$$A_i^- = \{y \in \mathbb{R}^{n\times K} \mid \mathcal{Q}^i(Ty) \le Q^i \bullet \bar{X}\}$$
$$A_i^+ = \{y \in \mathbb{R}^{n\times K} \mid \mathcal{Q}^i(Ty) \ge Q^i \bullet \bar{X}\}$$
$$A_i = \{y \in \mathbb{R}^{n\times K} \mid \mathcal{Q}^i(Ty) = Q^i \bullet \bar{X}\}.$$

  - use union bound for measure of $A_i^-(\varepsilon) \cap A_i^+(\varepsilon)$
  - show $A_i^-(\varepsilon) \cap A_i^+(\varepsilon) = A_i(\varepsilon)$
  - use union bound for measure of intersections of $A_i(\varepsilon)$
  - appropriate values for some parameters $\Rightarrow$ result

The heuristic

1. Solve SDP relaxation of DGP, get soln. $\bar{X}$
2. Barvinok's algorithm:
   $T \leftarrow \mathsf{factor}(\bar{X}), y \sim \mathcal{N}^{nK}(0, \frac{1}{\sqrt{K}}), x' \leftarrow Ty$
3. Use x' as starting point for a local NLP solver on formulation

$$\min_x \sum_{\{i,j\} \in E} \left(\|x_i - x_j\|_2^2 - d_{ij}^2 \right)^2$$

and return improved solution x

Outline

The Interval DGP

- **Distances never precise in applications**
- **DGP with constraints**

$$\forall \{i,j\} \in E \quad L_{ij}^2 \leq \|x_i - x_j\|^2 \leq U_{ij}^2$$

- **SDP formulation constraints**

$$\forall \{i,j\} \in E \quad L_{ij}^2 \leq X_{ii} + X_{jj} - 2X_{ij} \leq U_{ij}^2$$

- **NLP formulation**

$$\min \sum_{\{i,j\} \in E} \big((\max(L_{ij}^2 - \|x_i - x_j\|_2^2, 0)^2 + (\max(\|x_i - x_j\|_2^2 - U_{ij}^2, 0))^2 \big)$$

Our result

- **Extension of Barvinok's thm to intervals**

$$\mathsf{Prob}\left(\forall k \leq m \quad \mathsf{dist}(x', \mathcal{X}_k) \leq c\sqrt{\|\bar{X}\|_2 \ln m}\right) \geq 0.9.$$

- *Unfortunately, a worse bound: for most applications,* $\ln m \geq \ln n$

Outline

Evaluation metrics

- **LDE**$(x) = \max_{\{i,j\} \in E} | \, \|x_i - x_j\| - d_{ij} \, |$
- **MDE**$(x) = \frac{1}{|E|} \sum_{\{i,j\} \in E} | \, \|x_i - x_j\| - d_{ij} \, |$
- **CPU time**
- **For intervals: replace** $\|x_i - x_j\| - d_{ij}$ **by**

$$\max(L_{ij} - \|x_i - x_j\|_2, 0) + \max(\|x_i - x_j\|_2 - U_{ij}, 0)$$

Comparison with PCA

- Barvinok's heuristic: SDP + Barvinok + NLP
- Best-known dimensionality reduction algorithm: *Principal Component Analysis* (PCA)
  1. Decompose SDP soln. $\bar{X}$ into $P^\top \mathsf{diag}(\lambda) P$
  2. $\forall k > K$ let $\lambda_k \leftarrow 0$
  3. $x' \leftarrow P^\top \mathsf{diag}(\lambda) P$
- Compare against heuristic: SDP + PCA + NLP

Results on DGP

instance	*MDE* barvinok	*MDE* pca	*LDE* barvinok	*LDE* pca	*CPU* barvinok	*CPU* pca
`names`	**0.00**	0.11	**0.07**	1.00	39.33	**22.44**
`pept`	**0.00**	0.10	**0.03**	1.81	83.91	**56.65**
`C0020pdb`	**0.00**	0.12	**0.01**	2.72	76.73	**49.39**
`1guu-1`	0.03	**0.00**	0.26	**0.08**	370.73	**322.66**
`1guu-4000`	**0.03**	0.12	**0.73**	1.15	415.66	**397.87**
`1guu`	0.02	**0.01**	**0.29**	0.33	305.54	**267.52**
`res_5000`	**0.00**	0.15	**0.00**	2.24	84.19	**53.36**
`res_2000`	**0.00**	0.07	**0.00**	1.46	85.26	**53.39**
`res_0`	0.00	0.00	**0.00**	0.01	93.08	**62.64**
`res_3000`	**0.00**	0.01	**0.00**	1.08	88.51	**53.43**
`res_1000`	**0.00**	0.10	**0.00**	3.05	87.88	**52.98**
`res_2kxa`	**0.00**	0.15	**0.00**	2.92	764.34	**713.35**
`C0030pkl`	**0.00**	0.11	**0.07**	2.19	1178.73	**1024.86**

Most of the CPU time taken by the local NLP solver `scipy.optimize.root`

Results on iDGP

instance	*MDE* barvinok	*MDE* pca	*LDE* barvinok	*LDE* pca	*CPU* barvinok	*CPU* pca
names	0.04	**0.00**	2.11	**0.07**	53.91	**36.86**
pept	0.01	**0.00**	0.46	**0.40**	133.28	**99.60**
C0020pdb	0.02	**0.00**	1.64	**0.42**	112.38	**79.22**
1guu-1	0.02	**0.01**	1.09	**0.58**	500.64	**440.50**
1guu-4000	0.03	**0.02**	1.49	1.49	522.19	**461.53**
1guu	*memory overflow*					
res_5000	0.01	**0.00**	0.69	**0.08**	30764.21	**30465.16**
res_2000	0.01	**0.00**	1.78	**0.10**	33017.88	**32713.91**
res_0	0.00	0.00	0.11	0.11	22897.14	**22619.79**
res_3000	0.00	0.00	**0.05**	0.08	26095.91	**25846.81**
res_1000	0.00	0.00	**0.05**	0.07	27790.87	**27542.96**
res_2kxa	*memory overflow*					
C0030pkl	*memory overflow*					

Most of the CPU time taken by the local NLP solver `scipy.optimize.root`

Future directions

- What kind of local descent algorithm would be most appropriate?
- Improve performance on iDGP
  change SDP/NLP formulations
- Test this technique on a wider range of SDPs

[L., Ky Vu, *Barvinok's Naive Algorithm in Distance Geometry*, Op. Res. Lett., in revision]