Convex integer programming model to obtain dual bounds for sparse PCA

Santanu S. Dey[1] Rahul Mazumder[2] Guanyi Wang[1]

[1]Industrial and Systems Engineering, Georgia Institute of Technology,
[2]Operations Research Center, Massachusetts Institute of Technology

Discrete Optimization and Machine Learning

1
Introduction

Motivation: Why Sparse PCA

- The **principal component analysis (PCA)** problem

$$\max_{\|x\|_2=1} x^\top Ax,$$

where A is a positive semidefinte matrix, is one of the most widely used dimensionality reduction method in statistics and data science.

- In many applications, most of the components of the principle component (PC) direction are nonzero - this impedes **interpretability**.
- To enhance interpretability, it is desirable to incorporate cardinality (ℓ_0-norm) constraint for sparsity requirement as the **sparse principle component analysis (SPCA)** problem:

$$\lambda^k(A) \triangleq \max_x x^\top Ax \text{ s.t. } \|x\|_2 \leq 1, \|x\|_0 \leq k. \qquad \text{(SPCA)}$$

Motivation: Heuristics do not come with certificates

- Many heuristic algorithms have been proposed in the literature that use greedy methods, alternating methods and the related power methods to find a "**good**" feasible solution of SPCA problem.
- However, conditions under which (some of) these computationally friendlier methods can be shown to "work well", make very **strong** and often **unverifiable** assumptions on the problem data.
- Therefore, the performances of these heuristic methods are not clear **in general**.

1.1
Obtaining Dual Bounds

How to obtain certificates- Dual bounds

$$\begin{aligned} z^* := \quad & \max \quad f(x) \\ & \text{s.t.} \quad x \in S \end{aligned}$$

1. We want z^{UB} such that $z^{UB} \geq z^*$. (Why?)
2. Relaxation:

$$\begin{aligned} z^{UB} := \quad & \max \quad f(x) \ \text{g(x)} \\ & \text{s.t.} \quad x \in S \ \text{T} \end{aligned}$$

where (1) $g(x) \geq f(x)$ for all $x \in S$, (2) $S \subseteq T$

How to obtain certificates- Dual bounds

$$\begin{aligned} z^* := \quad & \max \quad f(x) \\ & \text{s.t.} \quad x \in S \end{aligned}$$

1. We want z^{UB} such that $z^{UB} \geq z^*$. (Why?)
2. Relaxation:

$$\begin{aligned} z^{UB} := \quad & \max \quad \cancel{f(x)}\ \text{g(x)} \\ & \text{s.t.} \quad x \in \cancel{S}\ \text{T} \end{aligned}$$

where (1) $g(x) \geq f(x)$ for all $x \in S$, (2) $S \subseteq T$

How to obtain certificates- Dual bounds

$$\begin{aligned} z^* := \quad & \max \quad f(x) \\ & \text{s.t.} \quad x \in S \end{aligned}$$

1. We want z^{UB} such that $z^{UB} \geq z^*$. (Why?)
2. Relaxation:

$$\begin{aligned} z^{UB} := \quad & \max \quad \cancel{f(x)}\ \text{g(x)} \\ & \text{s.t.} \quad x \in \cancel{S}\ \text{T} \end{aligned}$$

where (1) $g(x) \geq f(x)$ for all $x \in S$, (2) $S \subseteq T$

1.2
Standard Relaxation: SDP relaxation

Semi-definite programming (SDP) relaxation

$$\lambda^k(A) \triangleq \max_x x^\top Ax \text{ s.t. } \|x\|_2 \leq 1, \|x\|_0 \leq k. \qquad \text{(SPCA)}$$

SPCA is equivalent to a nonconvex problem ($X = xx^\top$):

$$\begin{aligned}\lambda^k(A) = \max\ & \operatorname{tr}(AX) \\ \text{s.t. } & \operatorname{tr}(X) = 1, \|X\|_0 \leq k^2, X \succeq 0, \operatorname{rank}(X) = 1.\end{aligned}$$

SDP relaxation: Relaxing by dropping rank constraint and replacing cardinality constraints with $\mathbf{1}^\top|X|\mathbf{1} \leq k$ gives the standard SDP relaxation:

$$\begin{aligned}\max\ & \operatorname{tr}(AX) \\ \text{s.t. } & \operatorname{tr}(X) = 1, \mathbf{1}^\top|X|\mathbf{1} \leq k, X \succeq 0. \qquad \text{(SDP)}\end{aligned}$$

Semi-definite programming (SDP) relaxation

$$\lambda^k(A) \triangleq \max_x x^\top Ax \text{ s.t. } \|x\|_2 \leq 1, \|x\|_0 \leq k. \qquad \text{(SPCA)}$$

SPCA is equivalent to a nonconvex problem ($X = xx^\top$):

$$\begin{aligned} \lambda^k(A) = \max \ & \operatorname{tr}(AX) \\ \text{s.t. } & \operatorname{tr}(X) = 1, \|X\|_0 \leq k^2, X \succeq 0, \operatorname{rank}(X) = 1. \end{aligned}$$

SDP relaxation: Relaxing by dropping rank constraint and replacing cardinality constraints with $\mathbf{1}^\top |X| \mathbf{1} \leq k$ gives the standard SDP relaxation:

$$\begin{aligned} \max \ & \operatorname{tr}(AX) \\ \text{s.t. } & \operatorname{tr}(X) = 1, \mathbf{1}^\top |X| \mathbf{1} \leq k, X \succeq 0. \end{aligned} \qquad \text{(SDP)}$$

2
Main results

Our contributions

- Present a framework that involves solving convex integer programs (IP) to produce **dual bounds**, as certificates of optimality, which make no restrictive/unverifiable assumptions on the data.
- Present **theoreticial worst-case results** on the quality of the dual bound provided by the convex IP.
- In practice, the dual bounds are significantly better than worst-case performance, and even better than the SDP bounds on some real-life instances.

Our contributions

- Present a framework that involves solving convex integer programs (IP) to produce **dual bounds**, as certificates of optimality, which make no restrictive/unverifiable assumptions on the data.
- Present **theoreticial worst-case results** on the quality of the dual bound provided by the convex IP.
- In practice, the dual bounds are significantly better than worst-case performance, and even better than the SDP bounds on some real-life instances.

Our contributions

- Present a framework that involves solving convex integer programs (IP) to produce **dual bounds**, as certificates of optimality, which make no restrictive/unverifiable assumptions on the data.
- Present **theoreticial worst-case results** on the quality of the dual bound provided by the convex IP.
- In practice, the dual bounds are significantly better than worst-case performance, and even better than the SDP bounds on some real-life instances.

2.1
Integer Programming relaxation

Integer programing relaxation: First attempt

$$\begin{aligned}
\max \quad & x^\top A x && (1)\\
\text{s.t.} \quad & \|x\|_2 \leq 1 && (2)\\
& -z_j \leq x_j \leq z_j \ \forall j \in [n] && (3)\\
& \sum_{j\in[n]} z_j \leq k && (4)\\
& x \in \mathbb{R}^n, z \in \{0,1\}^n && (5)
\end{aligned}$$

1. Too many binary variables.
2. Cardinality constraint usually challenging to deal with.
3. Objective is still non-convex
4. (Approx 100-200 variables QPs with cardinality constraint solved)

Integer programing relaxation: First attempt

$$\begin{aligned}
\max \quad & x^\top A x && (1)\\
\text{s.t.} \quad & \|x\|_2 \leq 1 && (2)\\
& -z_j \leq x_j \leq z_j \ \forall j \in [n] && (3)\\
& \sum_{j\in[n]} z_j \leq k && (4)\\
& x \in \mathbb{R}^n, z \in \{0,1\}^n && (5)
\end{aligned}$$

1. Too many binary variables.
2. Cardinality constraint usually challenging to deal with.
3. Objective is still non-convex
4. (Approx 100-200 variables QPs with cardinality constraint solved)

How to solve SPCA? ℓ_1-relaxation of SPCA problem

The constraints $\|x\|_2 = 1, \|x\|_0 \leq k$ implies that $\|x\|_1 \leq \sqrt{k}$. Thus, one obtains the so-called ℓ_1-norm relaxation of SPCA as:

$$\text{OPT}_{\ell_1} \triangleq \max_x x^\top Ax \text{ s.t. } \|x\|_2 \leq 1, \|x\|_1 \leq \sqrt{k}. \qquad (\ell_1\text{-relax})$$

Theorem

The objective value OPT_{ℓ_1} is upper bounded by a multiplicative factor ρ^2 away from $\lambda^k(A)$, i.e.,

$$\lambda^k(A) \leq OPT_{\ell_1} \leq \rho^2 \cdot \lambda^k(A),$$

with $\rho \leq 1 + \sqrt{\frac{k}{k+1}}$.

How to solve SPCA? ℓ_1-relaxation of SPCA problem

The constraints $\|x\|_2 = 1, \|x\|_0 \leq k$ implies that $\|x\|_1 \leq \sqrt{k}$. Thus, one obtains the so-called ℓ_1-norm relaxation of SPCA as:

$$\text{OPT}_{\ell_1} \triangleq \max_x x^\top Ax \text{ s.t. } \|x\|_2 \leq 1, \|x\|_1 \leq \sqrt{k}. \qquad (\ell_1\text{-relax})$$

Theorem

The objective value OPT_{ℓ_1} is upper bounded by a multiplicative factor ρ^2 away from $\lambda^k(A)$, i.e.,

$$\lambda^k(A) \leq OPT_{\ell_1} \leq \rho^2 \cdot \lambda^k(A),$$

with $\rho \leq 1 + \sqrt{\frac{k}{k+1}}$.

Quality of ℓ_1-relaxation of SPCA problem

The ℓ_1-relax has two advantages:

1. As shown in Theorem 1 above, ℓ_1-relax gives a **constant factor** bound on SPCA,
2. The **feasible region is convex** and all the non-convexity is in the objective function.

$$\text{OPT}_{\ell_1} \triangleq \underbrace{\max_x x^\top A x}_{\text{Not convex}} \text{ s.t. } \underbrace{\|x\|_2 \leq 1, \|x\|_1 \leq \sqrt{k}}_{\text{convex!}}. \qquad (\ell_1\text{-relax})$$

Convex integer programming: Upper bounding the objective with piecewise linear function

We use ℓ_1-relax as our basic relaxation and further relax it in the following steps:

1. By spectral decomposition $A = \sum_{i=1}^{n} \lambda_i v_i v_i^\top$, the objective function
$$x^\top A x = \sum_{i=1}^{n} \lambda_i (x^\top v_i)^2.$$
2. Let λ^L be a lower bound of $\lambda^k(A)$, split eigenvalues into two parts based on λ^L as $I_1 \triangleq \{i : \lambda_i > \lambda\}$, $I_2 \triangleq \{i : \lambda_i \leq \lambda\}$, then the objective can be written as
$$x^\top A x = \lambda^L + \underbrace{\sum_{i \in I_1} (\lambda_i - \lambda^L)(x^\top v_i)^2}_{\text{convex}} + \underbrace{\sum_{i \in I_2} (\lambda_i - \lambda^L)(x^\top v_i)^2}_{\text{concave}}$$
3. For convex part, linearize $(x^\top v_i)^2$ by a new variable ξ_i with SOS-2 constraints $<--$ needs binary variables

Convex integer programming: Upper bounding the objective with piecewise linear function

We use ℓ_1-relax as our basic relaxation and further relax it in the following steps:

1. By spectral decomposition $A = \sum_{i=1}^{n} \lambda_i v_i v_i^\top$, the objective function
$$x^\top A x = \sum_{i=1}^{n} \lambda_i (x^\top v_i)^2.$$

2. Let λ^L be a lower bound of $\lambda^k(A)$, split eigenvalues into two parts based on λ^L as $I_1 \triangleq \{i : \lambda_i > \lambda\}$, $I_2 \triangleq \{i : \lambda_i \leq \lambda\}$, then the objective can be written as
$$x^\top A x = \lambda^L + \underbrace{\sum_{i \in I_1} (\lambda_i - \lambda^L)(x^\top v_i)^2}_{\text{convex}} + \underbrace{\sum_{i \in I_2} (\lambda_i - \lambda^L)(x^\top v_i)^2}_{\text{concave}}$$

3. For convex part, linearize $(x^\top v_i)^2$ by a new variable ξ_i with SOS-2 constraints $<--$ needs binary variables

Convex integer programming: Upper bounding the objective with piecewise linear function

We use ℓ_1-relax as our basic relaxation and further relax it in the following steps:

1. By spectral decomposition $A = \sum_{i=1}^{n} \lambda_i v_i v_i^\top$, the objective function
$$x^\top A x = \sum_{i=1}^{n} \lambda_i (x^\top v_i)^2.$$

2. Let λ^L be a lower bound of $\lambda^k(A)$, split eigenvalues into two parts based on λ^L as $I_1 \triangleq \{i : \lambda_i > \lambda\}$, $I_2 \triangleq \{i : \lambda_i \leq \lambda\}$, then the objective can be written as
$$x^\top A x = \lambda^L + \underbrace{\sum_{i \in I_1} (\lambda_i - \lambda^L)(x^\top v_i)^2}_{\text{convex}} + \underbrace{\sum_{i \in I_2} (\lambda_i - \lambda^L)(x^\top v_i)^2}_{\text{concave}}$$

3. For convex part, linearize $(x^\top v_i)^2$ by a new variable ξ_i with SOS-2 constraints $<--$ needs binary variables.

Convex integer programming: Upper bounding the objective with piecewise linear function

We use ℓ_1-relax as our basic relaxation and further relax it in the following steps:

1. By spectral decomposition $A = \sum_{i=1}^{n} \lambda_i v_i v_i^\top$, the objective function
$$x^\top A x = \sum_{i=1}^{n} \lambda_i (x^\top v_i)^2.$$

2. Let λ^L be a lower bound of $\lambda^k(A)$, split eigenvalues into two parts based on λ^L as $I_1 \triangleq \{i : \lambda_i > \lambda\}$, $I_2 \triangleq \{i : \lambda_i \leq \lambda\}$, then the objective can be written as
$$x^\top A x = \lambda^L + \underbrace{\sum_{i \in I_1} (\lambda_i - \lambda^L)(x^\top v_i)^2}_{\text{convex}} + \underbrace{\sum_{i \in I_2} (\lambda_i - \lambda^L)(x^\top v_i)^2}_{\text{concave}}$$

3. For convex part, linearize $(x^\top v_i)^2$ by a new variable ξ_i with SOS-2 constraints $<--$ needs binary variables.

Convex integer programming: Model

Thus, a convex integer programming problem is obtained as follows:

$$\lambda^L + \max_{x,y,g,\xi,\eta,s} \sum_{i\in\{i:\lambda_i>\lambda^L\}} (\lambda_i - \lambda^L)\xi_i - s \qquad (\triangleq \mathrm{OPT}_{\text{convex-IP}})$$

$$\text{s.t.} \quad \begin{array}{l} g_i = x^\top v_i,\ i \in [n] \\ \left\{ \begin{array}{l} g_i = \sum_{j=-N}^{N} \gamma_i^j \eta_i^j,\ i \in \{i:\lambda_i > \lambda\} \\ \xi_i = \sum_{j=-N}^{N} (\gamma_i^j)^2 \eta_i^j,\ i \in \{i:\lambda_i > \lambda\} \\ (\eta_i^{-N}, \ldots, \eta_i^N) \in \text{SOS-2},\ i \in \{i:\lambda_i > \lambda\} \end{array} \right. \\ \left\{ \begin{array}{l} \sum_{i=1}^n x_i^2 \le 1 \\ \sum_{i\in\{i:\lambda_i>\lambda\}} \xi_i + \sum_{i\in\{i:\lambda_i\le\lambda\}} g_i^2 \le 1 + \frac{1}{4N^2} \sum_{i\in\{i:\lambda_i>\lambda\}} \theta_i^2 \end{array} \right. \\ \sum_{i=1}^n y_i \le \sqrt{k} \text{ and } y_i \ge x_i, y_i \ge -x_i\ i \in \{i:\lambda_i > \lambda\} \\ -\theta_i \le g_i \le \theta_i,\ i \in [n] \\ \sum_{i\in\{i:\lambda_i\le\lambda^l\}} -(\lambda_i - \lambda^l) g_i^2 \le s \end{array}$$

(Convex-IP)

Convex integer programming: Propositions

As a corollary of Theorem 1 we arrive at:

Proposition 1

The optimal objective value $\text{OPT}_{\text{convex-IP}}$ of Convex-IP is upper bounded by

$$\text{OPT}_{\text{convex-IP}} \leq \left(1 + \sqrt{\frac{k}{1+k}}\right)^2 \lambda^k(A) + \frac{1}{4N^2} \sum_{i \in \{i:\lambda_i > \lambda^L\}} (\lambda_i - \lambda^L)\theta_i^2.$$

Convex integer programming: Computation advantage

- The ℓ_0 constraint of SPCA using integer programing needs n binary variables.

$$x_i \leq z_i, \ \sum_i z_i \leq k, \ z \in \{0,1\}^n$$

- However, the number of binary variables in Convex-IP is $(2N+1) \cdot |\{i : \lambda_i > \lambda\}|$ which is typically significantly lesser than n.

Convex integer programming: Computation advantage

- The ℓ_0 constraint of SPCA using integer programing needs n binary variables.

$$x_i \leq z_i, \ \sum_i z_i \leq k, \ z \in \{0,1\}^n$$

- However, the number of binary variables in Convex-IP is $(2N+1) \cdot |\{i : \lambda_i > \lambda\}|$ which is typically significantly lesser than n.

Convex integer programming: Polynomial solvable

Therefore,

Proposition 3

Given the number of splitting points N and the size of set $\{i : \lambda_i > \lambda\}$, the Convex-IP problem can be solved in polynomial time.

Note that the polynomial time solvable for convex integer programming method **does not contradict known inapproximabiliy results for SPCA problem**.

Some more tricks

1. Massaging the matrix:

   - $$x^\top Ax = \lambda^L + \underbrace{\sum_{i \in I_1} (\lambda_i - \lambda^L)(x^\top v_i)^2}_{\text{convex}} + \underbrace{\sum_{i \in I_2} (\lambda_i - \lambda^L)(x^\top v_i)^2}_{\text{concave}}$$

   - $$x^\top Ax = \lambda^L + \underbrace{\sum_{i \in I_1} (\lambda_i - \lambda^L)(x^\top v_i)^2}_{\text{convex}} + \underbrace{\sum_{i \in I_2} (\bar{\lambda} - \lambda^L)(x^\top v_i)^2}_{\text{concave}}$$

   where $\bar{\lambda} \geq \lambda_i$ for all $i \in I_2$.

2. Cutting-planes: Let $x \in \mathbb{R}^n$. Let $|x_{i_1}| \geq |x_{i_2}| \geq \cdots \geq |x_{i_{n-1}}| \geq |x_{i_n}|$ where $\{i_1, i_2, \ldots, i_k\} \subseteq \{1, \ldots, n\}$. Then let v be the vector:

$$v_{i_j} = \begin{cases} |x_{i_j}| & \text{if } j \leq k \\ |x_{i_k}| & \text{if } j > k \end{cases} \tag{6}$$

Also let $b_{(v)} := \|(v_{i_1}, v_{i_2}, v_{i_3}, \ldots, v_{i_k})\|_2$. The the inequality

$$v^\top y \leq b_{(v)},$$

Some more tricks

1. Massaging the matrix:
   - $$x^\top Ax = \lambda^L + \underbrace{\sum_{i\in I_1}(\lambda_i - \lambda^L)(x^\top v_i)^2}_{\text{convex}} + \underbrace{\sum_{i\in I_2}(\lambda_i - \lambda^L)(x^\top v_i)^2}_{\text{concave}}$$
   - $$x^\top Ax = \lambda^L + \underbrace{\sum_{i\in I_1}(\lambda_i - \lambda^L)(x^\top v_i)^2}_{\text{convex}} + \underbrace{\sum_{i\in I_2}(\bar{\lambda} - \lambda^L)(x^\top v_i)^2}_{\text{concave}}$$

   where $\bar{\lambda} \geq \lambda_i$ for all $i \in I_2$.
2. Cutting-planes: Let $x \in \mathbb{R}^n$. Let $|x_{i_1}| \geq |x_{i_2}| \geq \cdots \geq |x_{i_{n-1}}| \geq |x_{i_n}|$ where $\{i_1, i_2, \ldots, i_k\} \subseteq \{1, \ldots, n\}$. Then let v be the vector:

$$v_{i_j} = \begin{cases} |x_{i_j}| & \text{if } j \leq k \\ |x_{i_k}| & \text{if } j > k \end{cases} \tag{6}$$

Also let $b_{(v)} := \|(v_{i_1}, v_{i_2}, v_{i_3}, \ldots, v_{i_k})\|_2$. The the inequality

$$v^\top y \leq b_{(v)}, \tag{7}$$

3
Numerical experiments

Data sets & Hardware and Software

In this section, we empirically compare the performances of Convex-IP method, Pert-Convex-IP method and Standard SDP relaxation method on three types of artificial examples:

- Spiked covariance recovery cases
- Synthetic cases
- Controlling sparsity cases

then test their performances on the three types of real examples:

- several biological data sets
- large-scale internet data set

All numerical experiments are implemented on MacBookPro13 with 2 GHz Intel Core i5 CPU and 8 GB 1867 MHz LPDDR3 Memory. Convex IPs were solved using Gurobi 7.0.2. SDPs were solved using Mosek 8.0.0.60.

Notations and settings

- A total time of 7200 seconds were given to each instance for running the convex IP (any extra time reported in the tables is due to running time of singular value decomposition and primal heuristics).
- We have run all our experiments with $k = 10$.
- For the Convex-IP method, we use the following parameter: $(I_{\text{pos}}, N) = (10, 3)$.
- For the Pert-Convex-IP method, let iter be the maximum number of iterations, we implement on the setting $(I_{\text{pos}}, N, \text{iter}) = (5, 3, 10)$.

Artificial data sets - I

Table: 'Spiked Covariance' Recovery

Case	LB-ℓ_0	Convex IP			SDP		
		UB	gap	Time	UB	gap	Time
Case 1 (200, 10)	511.95	511.99	0.007 %	76	511.959	0.001 %	1277
Case 2 (200, 10)	592.45	592.49	0.006 %	615	592.463	0.002 %	1458
Case 1 (300, 10)	414.04	414.17	0.03 %	642	NaN	-	-
Case 2 (300, 10)	568.56	568.65	0.016 %	82	NaN	-	-
Case 1 (400, 10)	478.24	478.41	0.04 %	793	NaN	-	-
Case 2 (400, 10)	426.91	427.15	0.06 %	181	NaN	-	-
Case 1 (500, 10)	256.82	257.37	0.21 %	1345	NaN	-	-
Case 2 (500, 10)	551.74	551.97	0.04 %	152	NaN	-	-
Case 1 (1000, 10)	315.16	317.00	0.57 %	1147	NaN	-	-
Case 2 (1000, 10)	383.44	384.73	0.34 %	2745	NaN	-	-

Artificial data sets - II

Table: 'Synthetic Example'

Case	**LB-ℓ_0**	**Convex IP**			**SDP**		
		UB	gap	Time	UB	gap	Time
Case 1 (200, 10)	5634.14	5642.24	0.14 %	38	5639.86	0.10 %	1092
Case 2 (200, 10)	7321.22	7330.92	0.13 %	23	7327.84	0.09 %	1086
Case 1 (300, 10)	4157.46	4168.86	0.27 %	83	NaN	-	-
Case 2 (300, 10)	5135.48	5147.40	0.23 %	62	NaN	-	-
Case 1 (400, 10)	6519.36	6533.74	0.22 %	98	NaN	-	-
Case 2 (400, 10)	5942.04	5963.72	0.36 %	56	NaN	-	-
Case 1 (500, 10)	5125.84	5145.36	0.38 %	149	NaN	-	-
Case 2 (500, 10)	5545.84	5567.36	0.39 %	50	NaN	-	-
Case 1 (1000, 10)	5116.08	5145.83	0.58 %	257	NaN	-	-
Case 2 (1000, 10)	6946.12	6973.33	0.39 %	323	NaN	-	-

Artificial data sets - III

Table: 'Controlling Sparsity'

Case	LB-ℓ_0	Convex IP			SDP		
		UB	gap	Time	UB	gap	Time
Case 1 (200, 10)	706	727	2.9 %	117	709	0.42 %	1360
Case 2 (200, 10)	680	704	3.53 %	176	688	1.2 %	1148
Case 1 (300, 10)	972	1010	3.91	135	NaN	-	-
Case 2 (300, 10)	976	1013	3.79 %	278	NaN	-	-
Case 1 (400, 10)	1239	1288	4.21 %	769	NaN	-	-
Case 2 (400, 10)	1207	1250	3.56 %	221	NaN	-	-
Case 1 (500, 10)	1498	1576	5.21 %	1026	NaN	-	-
Case 2 (500, 10)	1498	1560	4.14 %	251	NaN	-	-
Case 1 (1000, 10)	3948	6305	59.7 %	2206	NaN	-	-
Case 2 (1000, 10)	4002	6325	58.1 %	3270	NaN	-	-

Real data sets

Table: Biological and Internet Data

Case	LB-ℓ_0	Convex IP			SDP		
		UB	gap	Time	UB	gap	Time
Eisen-1 (79, 10)	17.33	17.35	0.12 %	63	17.71	2.2 %	15
Eisen-2 (118, 10)	11.71	11.87	1.4 %	96	11.94	2.0 %	52
Colon (500, 10)	2641	3373	27.7 %	708	NaN	-	-
Lymphoma (500, 10)	6008	8470	41 %	610	NaN	-	-
Reddit (2000, 10)	1523	1712	% 12.41	5932	NaN	-	-

Size of instances we solved

- **SDP:** Because of limitation of hardware and software, the SDP relaxation method cannot be implemented on the instances with input matrix of size greater than or equal to 300×300.
- **Convex IP:** The pertubed convex IP scales significantly better that the other methods. While we experimented with instances up to size 2000×2000, we believe this method will easily scale to larger instances.

Quality of dual bound

- **SDP vs {Convex IP}:** On some instances SDP obtained better dual bounds, but not in all. For example on both the real data sets Eisen-1 and Eisen-2, SDP bunds are worse.
- **Over all gaps for {Convex IP}:** Except for the random instances of type 'controlling sparsity' of size 1000×1000, and Lymphoma data set, in all other instances at least one method had a gap less that 10%.

Thank you!

Find paper on:
`https://www2.isye.gatech.edu/sdey30/publications.html`.

How to solve SPCA? Classical integer programming approach

The classical integer programming approach to solve SPCA problem would be to go to an extended space of product of $x-$variables combining with one binary variable per $x-$variable as follows:

$$\max \operatorname{tr}(AX) \text{ s.t. } x_i \leq z_i, \forall i \in [n]; \ \sum_{i=1}^{n} z_i \leq k; \ \|x\|_2 \leq 1;$$
$$\begin{bmatrix} 1 & x^\top \\ x & X \end{bmatrix} \succeq 0; \ \operatorname{rank}\left(\begin{bmatrix} 1 & x^\top \\ x & X \end{bmatrix}\right) = 1; \ z_i \in \{0,1\}^n.$$

It is easy to see that such a model is challenging due to (1) "**quadratic**" increasing in number of variables (X) (2) **rank constraints**, and (3) n **binary variables**.

Perturbation of convex IP: Construction

In order to reduce the running time of convex integer programming method, we do the following:

1. Recall that in convex integer programming, we split eigenvalues into two sets based on a threshold value λ.
2. Replace all the eigenvalues in I_2 with value $\bar{\lambda} > \max_{i \in I_2} \lambda_i$, i.e., the perturbed sample covariance matrix A now become

$$\bar{A} = \sum_{i \in I_1} \lambda_i v_i v_i^\top + \sum_{i \in I_2} \bar{\lambda} v_i v_i^\top .$$

Under this case, some convex quadratic constraint can be simplified by linear constraint.

Perturbation of convex IP: Construction

In order to reduce the running time of convex integer programming method, we do the following:

1. Recall that in convex integer programming, we split eigenvalues into two sets based on a threshold value λ.
2. Replace all the eigenvalues in I_2 with value $\bar{\lambda} > \max_{i \in I_2} \lambda_i$, i.e., the perturbed sample covariance matrix A now become

$$\bar{A} = \sum_{i \in I_1} \lambda_i v_i v_i^\top + \sum_{i \in I_2} \bar{\lambda} v_i v_i^\top.$$

Under this case, some convex quadratic constraint can be simplified by linear constraint.

Perturbation of convex IP: Model

Thus a simplifed convex IP corresponding to the perturbed version of the matrix is:

$$\lambda + \max_{x,y,g,\xi,\eta,s} \sum_{i\in\{i:\lambda_i>\lambda\}} (\lambda_i - \lambda)\xi_i - s \triangleq \mathrm{OPT}_{\text{pert-convex-IP}}$$

$$\begin{aligned}
& g_i = x^\top v_i,\ i \in \{i : \lambda_i > \lambda\} \\
& \begin{cases} g_i = \sum_{j=-N}^{N} \gamma_i^j \eta_i^j,\ i \in \{i : \lambda_i > \lambda\} \\ \xi_i = \sum_{j=-N}^{N} (\gamma_i^j)^2 \eta_i^j,\ i \in \{i : \lambda_i > \lambda\} \\ (\eta_i^{-N}, \ldots, \eta_i^N) \in \text{SOS-2},\ i \in \{i : \lambda_i > \lambda\} \end{cases} \\
\text{s.t.} \quad & \begin{cases} \sum_{i=1}^{n} x_i^2 \le 1 \\ \sum_{i\in\{i:\lambda_i>\lambda\}} g_i^2 \le 1 - \dfrac{s}{\lambda - \bar{\lambda}} \\ 1 - \dfrac{s}{\lambda - \bar{\lambda}} \le \sum_{i\in\{i:\lambda_i>\lambda\}} \xi_i \le 1 + \frac{1}{4N^2} \sum_{i\in\{i:\lambda_i>\lambda\}} \theta_i^2 - \dfrac{s}{\lambda - \bar{\lambda}} \end{cases} \\
& \sum_{i=1}^{n} y_i \le \sqrt{k} \text{ and } y_i \ge |x_i|,\ i \in \{i : \lambda_i > \lambda\} \\
& -\theta_i \le g_i \le \theta_i,\ i \in \{i : \lambda_i > \lambda\}.
\end{aligned}$$

(Pert-Convex-IP)

Perturbation of convex IP: Proposition

Proposition

The optimal objective value $\text{OPT}_{\text{pert-convex-IP}}$ of Pert-Convex-IP is upper bounded by

$$\text{OPT}_{\text{pert-convex-IP}} \leq \rho^2 \lambda^k(A) + \rho^2(\bar{\lambda} - \lambda_{\min}(A)) + \frac{1}{4N^2} \sum_{i \in \{i:\lambda_i > \lambda\}} (\lambda_i - \lambda)\theta_i^2.$$

- Note that in Pert-Convex-IP, we do not need the variables $g_i, i \in \{i : \lambda_i \leq \lambda\}$ which greatly reduces the number of variables since in general the value $\{i : \lambda_i \geq \lambda\} \ll n$.
- In practice, we note a significant reduction in running time, while the dual bound obtained from Pert-Convex-IP model remains reasonable.

Perturbation of convex IP: Proposition

Proposition

The optimal objective value $\text{OPT}_{\text{pert-convex-IP}}$ of Pert-Convex-IP is upper bounded by

$$\text{OPT}_{\text{pert-convex-IP}} \leq \rho^2 \lambda^k(A) + \rho^2(\bar{\lambda} - \lambda_{\min}(A)) + \frac{1}{4N^2} \sum_{i \in \{i:\lambda_i > \lambda\}} (\lambda_i - \lambda)\theta_i^2.$$

- Note that in Pert-Convex-IP, we do not need the variables $g_i, i \in \{i : \lambda_i \leq \lambda\}$ which greatly reduces the number of variables since in general the value $\{i : \lambda_i \geq \lambda\} \ll n$.
- In practice, we note a significant reduction in running time, while the dual bound obtained from Pert-Convex-IP model remains reasonable.

Refining the splitting points

- Since the Pert-Convex-IP model runs much faster than the Convex-IP model, we run the Pert-Convex-IP model **iteratively**.
- In each new iteration, we add one extra splitting point describing each ξ_i function.
- In particular, once we solve Pert-Convex-IP model, we add one splitting point at the optimal value of g_i.

Cutting planes

Cutting planes

Let $x \in \mathbb{R}^n$. Let $|x_{i_1}| \geq |x_{i_2}| \geq \cdots \geq |x_{i_{n-1}}| \geq |x_{i_n}|$ where $\{i_1, i_2, \ldots, i_k\} \subseteq \{1, \ldots, n\}$. Then let v be the vector:

$$v_{i_j} = \begin{cases} |x_{i_j}| & \text{if } j \leq k \\ |x_{i_k}| & \text{if } j > k \end{cases} \tag{8}$$

Also let $b_{(v)} := \|(v_{i_1}, v_{i_2}, v_{i_3}, \ldots, v_{i_k})\|_2$. The the inequality $v^\top y \leq b_{(v)}$, is a valid inequality SPCA. See[1]

We add these inequalities at the end of each iteration, where the seeding x for constructing v is chosen to be the optimal solution of the previous iteration.

[1] **kim2016cardinality**.