CVXALGGEO USER MANUAL

MEHDI GHASEMI

1. What is CvxAlgGeo?

CvxAlgGeo is a package written for Sage and published under GNU(GPL) to implement a fast method to compute a lower bound for multivariate polynomials over a basic closed semialgebraic set. It is mainly based on methods introduced in [2], [1] and [3]. Moreover, to provide computational tools for usual calculation based on semidefinite programming CvxAlgGeo implements some functionalities available on SosTools and GlOptiPoly for Matlab [4, 8].

At this stage, CvxAlgGeo contains implementations of two different methods for polynomial optimization:

1.1. **By Geometric Programming:** Let $f = \sum_\alpha f_\alpha \underline{X}^\alpha$ be a polynomials on $X_1, \dots, X_n$ of degree d, where $\underline{X}^\alpha$ is short for $X_1^{\alpha_1} \cdots X_n^{\alpha_n}$, and $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{N}^n$. The degree of a monomial $\underline{X}^\alpha$ is denoted by $|\alpha|$ that is defined by $|\alpha| = \alpha_1 + \cdots + \alpha_n$. Denote f_α where $\alpha = 2d(\delta_{1i}, \dots, \delta_{ni})$ by $f_{2d,i}$ and $f_{(0,\dots,0)} = f(\underline{0})$ by f_0 for simplicity. Degree of a polynomial f ($\deg f$) is the maximum degree of its monomials. We also denote the set of those exponents α for which the monomial $f_\alpha \underline{X}^\alpha$ is not a square by Δ_f, i.e.,

$$\Delta_f \coloneqq \{\alpha \in \mathbb{N}^n \ : \ f_\alpha < 0 \text{ or } \alpha_i \text{ is odd for some } 1 \le i \le n\}.$$

Let $\mathbf{g} = (g_1, \dots, g_m)$ be a tuple of polynomials and $K_{\mathbf{g}} = \{x \in \mathbb{R}^n : g_j(x) \ge 0, j = 1, \dots, m\}$. Denote the infimum of f over $K_{\mathbf{g}}$ by $f_{*,\mathbf{g}}$, $\Delta = \Delta_f \cup \Delta_{-g_1} \cup \cdots \cup \Delta_{-g_m}$ and $A = (a_{kl})$ be a $(m+1) \times (m+1)$ matrix such that $a_{0l} = 1$ if $l = 1$ and 0 otherwise. Define $h_k = \sum_{j=0}^m a_{jk} g_j$, $k = 0, \dots, m$ where $g_0 = -f$ and $H(\mu) = -\sum_{j=0}^m \mu h_j$. We can Decompose H as $H^+ - H^-$ where all the coefficients of H^+ and H^- are nonnegative. If ρ is an optimum value for the following *geometric program*, then $f_{gp,\mathbf{g}} = -h_0(0) - \rho$

Date: January 25, 2013.

Key words and phrases. sums of squares, semidefinite programming, geometric programming, optimization.

is a lower bound for the global minimum of f on $K_{\mathbf{g}}$.

$$
(1)\qquad \begin{cases}
Minimize \ \sum_{j=1}^{m}\mu_j h_j(0)^+ + \sum_{\alpha\in\Delta^{<d}}(d-|\alpha|)\left[\left(\frac{w_\alpha}{d}\right)^d\left(\frac{\alpha}{\mathbf{z}_\alpha}\right)^\alpha\right]^{1/(d-|\alpha|)} \\
\text{s.t. } \sum_{\alpha\in\Delta} z_{\alpha,i} \ \leq \ H(\mu)_{d,i}\,, \quad i=1,\dots,n \\
\left(\frac{\mathbf{z}_\alpha}{\alpha}\right)^\alpha \geq \left(\frac{w_\alpha}{d}\right)^d, \quad \alpha\in\Delta^{=d} \\
w_\alpha \geq \max\{H(\mu)_\alpha^+, H(\mu)_\alpha^-\}, \quad \alpha\in\Delta \\
\sum_{k=0}^{m} a_{jk}\mu_k \geq 0, \quad j=1,\dots,m
\end{cases}
$$

Here $h_j^+(0)=\max\{h_j(0),0\}$ [3, Theorem 4.1].

In [1], [2] and [3] the advantages and disadvantages of solving (1) instead of solving the corresponding semidefinite program is discussed.

1.2. **By Semidefinite Programming:** It is well known that a feasible value for the semidefinite program (2) is a lower bound for the polynomial f on a semialgebraic set $K=K_S$, where $S=\{g_1,\dots,g_m\}$ and $K_S=\{x\in\mathbb{R}^n : g_i(x)\geq 0, i=1,\dots,m\}$.

$$
(2)\qquad \mathrm{Prg}_k : \begin{cases}
\min\limits_{y} & \sum_\alpha f_\alpha y_\alpha \\
\text{subject to} & M_k(y)\geq 0 \\
 & M_{k-\deg g_i}(g_i y)\geq 0 \quad i=1,\dots,m
\end{cases}
$$

for $k\geq\max\{\deg f,\deg g_1,\dots,\deg g_m\}$. All the notations are as described in [5]. Current implementation is based on the description given on [6, Chapter 4 and 5] and [7].

2. CvxAlgGeo

CvxAlgGeo implements few tools for computations over real polynomial rings in Python for Sage using CvxOpt. The current version consists of two main classes.

(1) `GlOptGeoPrg`
(2) `SosTools`

In this section we describe how to use these programs.

2.1. GlOptGeoPrg(Prog, Rng [, H=A, Settings]).

Initialization: This class implements (1), requires two mandatory arguments, `Prog`, `Rng` and two optional arguments `H` and `Settings`.

- `Prog=[f, Cons]` is a list consists of two parts:

  - '`f`' is the polynomial which is subject to minimizations.
  - '`Cons=[g1, ..., gm]`' is a list of polynomials corresponding to the constraints $g_j \geq 0$, for $j = 1, \cdots, m$.

  They must be elements of the polynomial ring `Rng`, '`PolynomialRing(F,'x',n)`' where `n` is the number of variables and '`F`' is the base field.
- `H` is the matrix $A = (a_{kl})$.
- `Settings` is a dictionary with the following keys:
  - '`Iterations`' is the number of iterations the `solver` tries to find an optimum solution. Its default value is 150.
  - '`Details`' is by default set to `False`. Set `True` to see the detail output of the CvxOpt solver.
  - '`tryKKT`' is by default 20 that forces the solver to continue for the given number of steps, even if a singular KKT matrix occurred.
  - '`AutoMatrix`' accepts a boolean '`True`' or '`False`'. The value `True` leaves the choice of A to the program automatically.

Running: The method `GlOptGeoPrg.minimize()` solves (1) for `f` and `g1,...,gm`. The attribute `GlOptGeoPrg.fgp` also contains the output of `minimize`.

Output: The attribute `GlOptGeoPrg.Info` is a dictionary which gives detailed information on the results of the method `minimize`.

- '`gp`' is the lower bound obtained from (1).
- '`Wall`' contains the wall time consumed by the solver.
- '`CPU`' contains the CPU time consumed by the solver.
- '`status`' and '`Message`' give some informations about the running process.

2.2. **SosTools(Prg, Rng [, Settings]).**

This class is on its early stage and aims to provides the same functionality as SosTools does for Matlab.

Initialization: This class implements the semidefinite program (2).

- `Prg=[f,Cons]` is a list consists of two parts:
  - '`f`' is the polynomial which is subject to minimizations.
  - '`Cons=[g1, ..., gm]`' is a list of polynomials corresponding to the constraints $g_j \geq 0$, for $j = 1, \cdots, m$.

  They must be elements of the polynomial ring `Rng`, '`PolynomialRing(F,'x',n)`' where `n` is the number of variables and '`F`' is the base field.
- `Settings` is exactly similar to `GlOptGeoPrg` with an extra index '`Order`' which corresponds to the index k for Prg_k in (2). The default value for `Order` is set to be 0 which results in the minimum size for moment matrices.

Running:

- The method `SosTools.init_sdp()` initializes the semidefinite program corresponding to (2).
- The method `SosTools.minimize()` returns the an optimum value f_{sos}, for (2) if exists.
- The method `SosTools.decompose` returns a vector of polynomials $[p_1, \dots, p_s]$ such that $f = \sum_{i=1}^{s} p_i^2$.

Output: The attribute `SosTools.Info` returns the similar information as for `GlOptGeoPrg.Info`. But instead of '`gp`' it has the key '`min`' for f_{sos}. The Info key '`is sos`' is by default `False`, but after executing '`decompose`' it may become `True`.

2.3. **Example.**

```
## Initial settings ##
n = 3 # Number of variables
d = 4 # Maximum degree of polynomials
m = 2 # Number of constraints
num_monos = 4 # Max number of monomials
R = PolynomialRing(RR,'x',n);
X = R.gens();

##########################################

Prg=[]
Cns=[]
f=R.random_element(d,randint(1,num_monos))+sum(p^d for p in X)
Prg.append(f)
for i in range(m):
    g=R.random_element(d,num_monos)
    Cns.append(g);
Prg.append(Cns)

conf={'Details':False,'tryKKT':20, 'AutoMatrix':False, 'Order':1}
M=identity_matrix(QQ,m+1)

print Prg

A=GlOptGeoPrg(Prg, R, H=M, Settings=conf)
A.minimize()
```

```
show(A.H)
print A.Info
B=SosTools(Prg, R, Settings=conf)
B.init_sdp()
B.minimize()
print B.Info
```

The code generates random polynomials and computes f_{gp} and $f_{gp,\mathbf{g}}$: Here is the output:

```
[x0^4 + x1^4 + x2^4 - x2^3 + 122*x0,
[-x0^2*x1 - 4*x2^3 - x1, -x0^2*x1*x2 + 10*x1^3*x2 - 2*x1^2*x2]]

{'status': 'Optimal', 'Message': 'Optimal solution found by solver.',
'Wall': 0.2654759883880615, 'CPU': 0.1799999999999926, 'gp':-285.988059619120}

{'status': 'Optimal', 'min': -285.98744579692226, 'Wall':1.1604969501495361,
'Message': 'Feasible solution for moments of order 3',
'CPU': 0.8499999999999943, 'Size': [20, 84]}
```

References

[1] M. Ghasemi, J. B. Lasserre and M. Marshall, *Lower bounds on the global minimum of a polynomial*, to appear.

[2] M. Ghasemi and M. Marshall, *Lower bounds for polynomials using geometric programming*, SIAM J. Optim. 22(2) (460-473), 2012.

[3] M. Ghasemi and M. Marshall, *Lower Bounds for a Polynomial on a basic closed semialgebraic set using geometric programming*, in progress.

[4] D. Henrion and J. B. Lasserre, *Gloptipoly: global optimization over polynomials with matlab and sedumi*, ACM Transactions on Mathematical Software, 29(2):165195, 2003.

[5] J. B. Lasserre, *Global optimization with polynomials and the problem of moments*, SIAM J. Opt. 11(3)(796817), 2001.

[6] J. B. Lasserre, *Moments, Positive Polynomials and Their Applications*, Imperial College Press Optimization Series Vol. 1, 2010.

[7] M. Laurent, *Sums of squares, moment matrices and optimization over polynomials*, 2010.

[8] S. Prajna, A. Papachristodoulou, P. Seiler, and P. A. Parrilo, *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*, 2004

Department of Mathematics and Statistics,
University of Saskatchewan,
Saskatoon, SK. S7N 5E6, Canada
E-mail address: mehdi.ghasemi@usask.ca