Low-Rank Univariate Sum-of-Squares Has No Spurious Local Minima

Presented by Benoît Legat
Based on joint work with Chenyang Yuan and Pablo Parrilo

First-order methods

- Amenability to **parallelization**
- Affordable **per-iteration** computational cost
- Low **storage** requirements

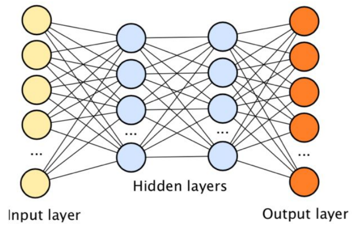
# nodes	PDLP	SCS	Gurobi Barrier	Gurobi Primal Simp.	Gurobi Dual Simp.
10^4	7.4 sec.	1.3 sec.	36 sec.	37 sec.	114 sec.
10^5	35 sec.	38 sec.	7.8 hr.	9.3 hr.	>24 hr.
10^6	11 min.	25 min.	OOM	>24 hr.	-
10^7	5.4 hr.	3.8 hr.	-	-	-

Applegate, David, et al. **Practical Large-Scale Linear Programming using Primal-Dual Hybrid Gradient**. *NeurIPS 2021.*

Deep Learning uses gradient-based solvers on large scale problems

Very successful on various classification and inference tasks

Solved with highly parallelized first-order methods



PyTorch

TensorFlow

Nonconvex factorization formulations

- Basin of attraction
  - Initialization
  - Iterative refinement
- Benign Global Landscape

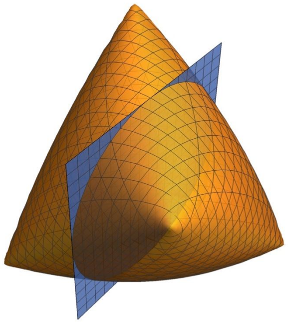
Require statistical/genericity conditions such as Restricted isometry property (RIP)

Matrix sensing, matrix completion, phase retrieval, blind deconvolution, ...

$$\underset{\boldsymbol{L}\in\mathbb{R}^{n_1\times r},\boldsymbol{R}\in\mathbb{R}^{n_2\times r}}{\text{minimize}} f(\boldsymbol{L},\boldsymbol{R}) = \frac{1}{4m}\sum_{i=1}^{m}\left(\langle \boldsymbol{A}_i, \boldsymbol{L}\boldsymbol{R}^\top\rangle - y_i\right)^2$$

Chi, Yuejie, Yue M. Lu, and Yuxin Chen. **Nonconvex optimization meets low-rank matrix factorization: An overview**. *IEEE Transactions on Signal Processing* 67.20 (2019): 5239-5269.

Semidefinite programming



Semidefinite programming (SDP) is a powerful and expressive convex optimization method

$n \times n$ positive semidefinite variable $X \succcurlyeq 0$ + m linear constraints

Applications: Optimal control, Lyapunov analysis, convex relaxations of combinatorial optimization, rank minimization and nuclear norm, ...

mosek

Typically solved with expensive interior point methods

- $O((mn + m^2)n^2)$ operations per iteration
- $O(\sqrt{n} \log(\varepsilon))$ iterations
- $O(m^2 + n^2)$ memory

First-order solver for nonconvex factorization formulation?

Introduction

Burer-Monteiro methods factor PSD constraint $X = UU^T$, then perform local optimization on resulting non-convex unconstrained problem

$$\langle A_i, X \rangle = b_i \quad \forall i$$
$$X \succeq 0$$

$$\longleftrightarrow$$

$$\min_U \sum_i (\langle A_i, UU^T \rangle - b_i)^2$$

Feasible — Optimum = 0

May get stuck in local optimum (explicit counterexamples where second-order critical point ≠ global minimum)

When is non-convexity benign?

Related work

For general SDP feasibility with m linear constraints, with the factorization $X = UU^T$, where U is a $n \times r$ matrix.

Second-order critical point ⇒ Global minimum (non-convexity benign) when:

- $r > n$ [Burer and Monteiro]
- $r = \Omega(\sqrt{m})$, but with smoothed analysis [Cifuentes and Moitra], generic constraints [Bhojanapalli, Boumal, Jain, Netrapalli], or determinant regularization [Burer and Monteiro], (necessary because of counterexamples)

Can we do better if the SDP has special structure?

Samuel Burer and Renato Monteiro. (2005) Local Minima and Convergence in Low-Rank Semidefinite Programming, Mathematical Programming.
Diego Cifuentes and Ankur Moitra. (2019) Polynomial time guarantees for the Burer-Monteiro method, arXiv:1912.01745.
Srinadh Bhojanapalli, Nicolas Boumal, Prateek Jain, Praneeth Netrapalli (2018) Smoothed analysis for low-rank solutions to semidefinite programs in quadratic penalty form, Conference on Learning Theory.

Sum of Squares Optimization

Given p(x), can we write it as a sum of squares? $p(x) = \sum_{i=1}^{r} u_i(x)^2$

Certifies that p(x) ≥ 0, and can be formulated as SDP

Focus on univariate trigonometric polynomials in this talk (methods can be generalized to multivariate case)

$$p(x) = a_0 + \sum_{k=1}^{d} a_k \cos(kx) + a_{-k} \sin(kx), \quad x \in [0, \pi]$$

Applications in signal processing, filter design and control

$$H(z) = C(zI - A)^{-1}B$$

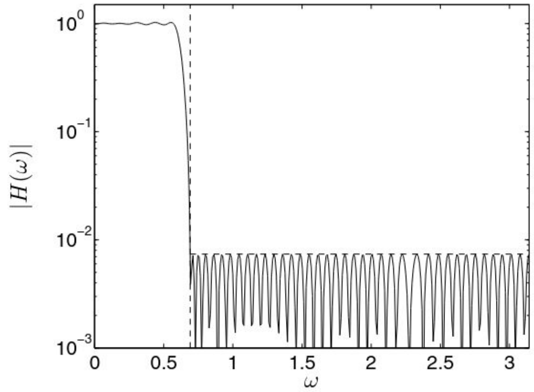
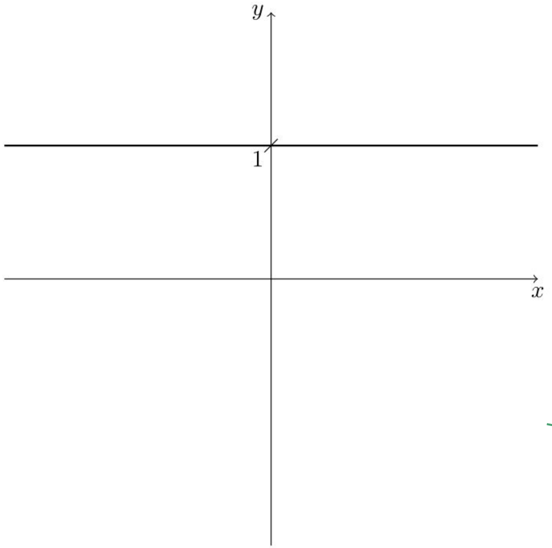


Image credit: Tae Roh and Lieven Vandenberghe. (2006) Discrete transforms, semidefinite programming and sum-of-squares representations of nonnegative polynomials. SIAM J. on Optimization.

Univariate to trigonometric basis

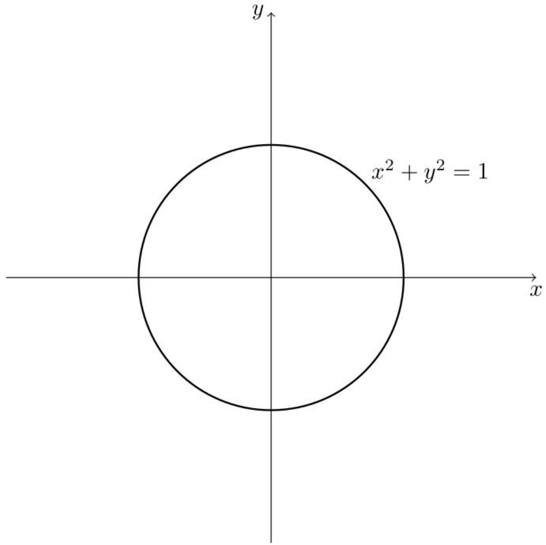


$$x^2 - 2x + 1$$

$$x^2 - 2xy + y^2$$

$$\cos(\alpha)^2 - 2\cos(\alpha)\sin(\alpha) + \sin(\alpha)^2$$

$$1 - \sin(2\alpha)$$



Linear transformation on *coefficients* :
Chebyshev basis

Section 1.5.1 : Dumitrescu, Bogdan. *Positive trigonometric polynomials and signal processing applications*. Vol. 103. Berlin: Springer, 2007.

Contributions

Find sum of squares decomposition of p(x) by solving

$$\min_u f(u) = \left\| \sum_{i=1}^{r} u_i(x)^2 - p(x) \right\|$$

For any norm on polynomials, if f(u) = 0, sum of squares decomposition agrees with p(x).

Theorem: when **r ≥ 2** (vs r = Ω(√m)) first-order methods find sum of squares decomposition for univariate polynomials (non-convexity benign)

If we choose right norm, ∇f(u) can be computed in O(d log d) time using fast fourier transforms (FFTs)

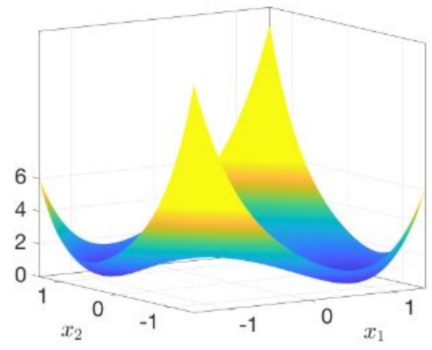


Image credit: Chi Yuejie, Lu Yue, Chen Yuxin. (2019) Nonconvex Optimization Meets Low-Rank Matrix Factorization: An Overview. IEEE Transactions on Signal Processing.

Sampled basis

Which inner product <p(x),q(x)> on polynomials to choose?

Given p(x), q(x) degree d, choose d+1 points x_k

$$\langle p(x), q(x)\rangle = \sum_{k=1}^{d+1} p(x_k)q(x_k), \quad \|p(x)\|^2 = \sum_{k=1}^{d+1} p(x_k)^2$$

Valid inner product: when x_k are distinct points, if $\|p(x)\|^2 = 0$ then $p(x) = 0$.

Sum of squares using a sampled/interpolation basis studied by [Löfberg and Parrilo] and [Cifuentes and Parrilo]

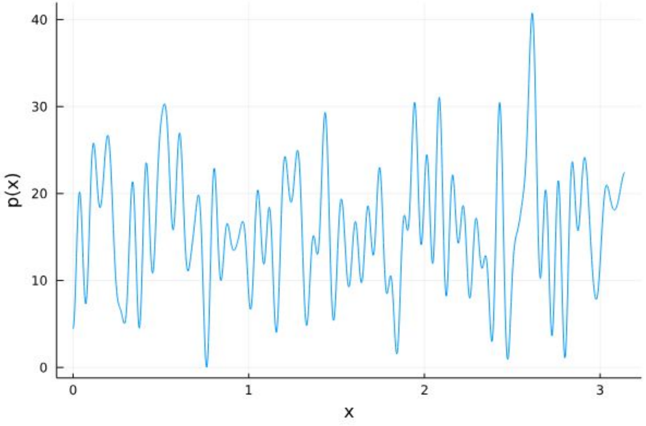
How should we choose x_k?

Diego Cifuentes and Pablo Parrilo. (2017) Sampling algebraic varieties for sum of squares programs, SIAM J. Optimization.
Johan Lofberg and Pablo Parrilo. (2004) From coefficients to samples: a new approach to SOS optimization, 43rd IEEE Conference on Decision and Control.

Numerical Implementation

Compute sum of squares decomposition of degree 2d trigonometric polynomial

$$p(x) = a_0 + \sum_{k=1}^{d} a_k \cos(kx) + a_{-k} \sin(kx)$$



Using basis vectors evaluated at 2d + 1 points

$$\langle p, q \rangle = \sum_{k=1}^{2d+1} p(x_k) q(x_k), \quad x_k = \frac{2k\pi}{2d+1}$$

$$B_k = \begin{bmatrix} 1 & \cos(x_k) & \cdots & \cos(\frac{d}{2}x_k) & \sin(x_k) & \cdots & \sin(\frac{d}{2}x_k) \end{bmatrix}^\top$$

Matrix-vector products in ∇f(U) can be computed by FFT

$$\nabla f(U) = U^T B \, \mathrm{diag}(\|U^T B_k\|^2 - p(x_k)) B^T$$

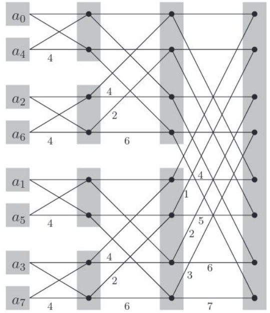
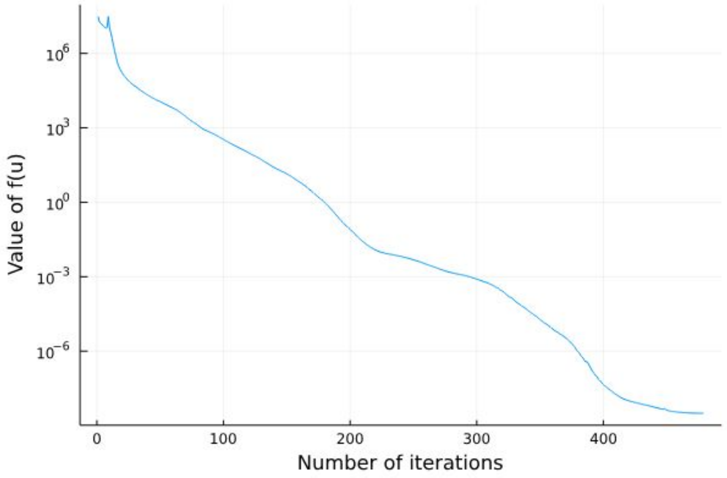


Image credit: Christos Papadimitriou, Sanjoy Dasgupta, and Umesh Vazirani. (2006) Algorithms

Results

Sum of squares decomposition for random trigonometric polynomial

Convergence rate for LBFGS with random initialization:



Degree	Time in seconds	Iterations
2,000	2 (1 – 2)	340 (306 – 384)
10,000	6 (5 – 6)	530 (497 – 592)
20,000	9 (8 – 10)	632 (587 – 695)
100,000	53 (46 – 59)	1126 (980 – 1248)
200,000	160 (139 – 174)	1375 (1212 – 1532)
1,000,000	1461 (1212 – 1532)	2303 (1934 – 2437)

Running times (stop at 10^{-7} relative error in U):

Use r = 4 with 4 cores.

Comparison with existing algorithms

Sturm sequence: Decide positivity of univariate polynomial of degree d in **$O(d^2)$**

Interior-point: Univariate Sum-of-Squares program of degree d in **$O(d^4)$** per iteration and **$O(\sqrt{d}\log(\varepsilon))$** iterations.
Infeasibility: Dual certificate.

Burer-Monteiro: **$O(d\log(d))$** per iteration for degree d.
Infeasibility: Projection to SOS cone.

Guarantee on number of iterations of Burer-Monteiro for univariate SOS ?

Proof Sketch

Assume that p(x) is a univariate polynomial and r = 2

$$f(u) = \left\|u_1(x)^2 + u_2(x)^2 - p(x)\right\|^2 = \|s(x) - p(x)\|^2$$

Given u such that $\nabla f(u)(v) = 0$ and $\nabla^2 f(u)(v,v) \geq 0$ for all v, show that $f(u) = 0$

We have inner product <p(x),q(x)> on polynomials with associated norm ||.||:

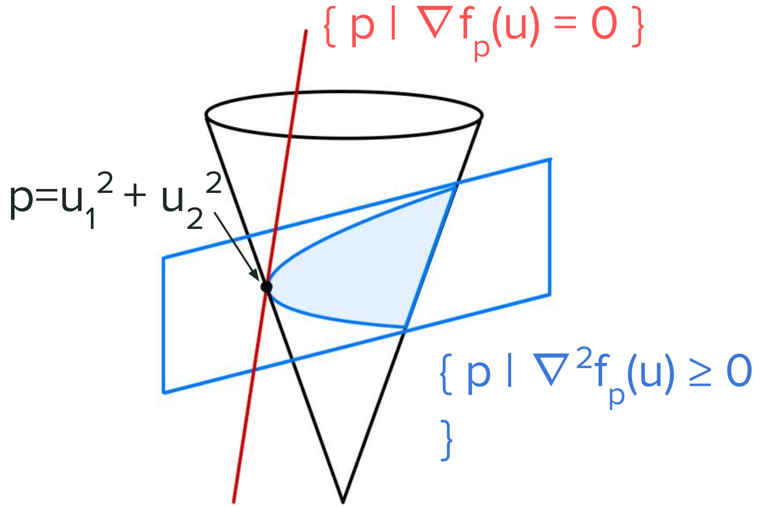
$$\nabla f(u)(v) \sim \left\langle \sum_{j=1}^{r} u_j(x) v_j(x), s(x) - p(x) \right\rangle = 0$$

$$\nabla^2 f(u)(v,v) \sim \langle \sum_{j=1}^{r} v_j(x)^2, s(x) - p(x) \rangle + 2 \left\| \sum_{j=1}^{r} u_j(x) v_j(x) \right\|^2 \geq 0$$

Proof Sketch

Geometrically, we want to show that the only intersection between set with zero gradient and PSD Hessian is when f(u) = 0.

For fixed u, these sets are convex!



Our proof can be interpreted as finding a certificate of this condition for every u and p.

Proof Sketch

$$\nabla f(u)(v) \sim \langle u_1(x)v_1(x) + u_2(x)v_2(x), s(x) - p(x)\rangle = 0$$

$$\nabla^2 f(u)(v,v) \sim \langle v_1(x)^2 + v_2(x)^2, s(x) - p(x)\rangle + 2\left\|u_1(x)v_1(x) + u_2(x)v_2(x)\right\|^2 \geq 0$$

Suppose u_1, u_2 coprime (true generically)

Bézout's lemma + gradient condition ⇒ exist v_1, v_2 s.t.

$$u_1(x)v_1(x) + u_2(x)v_2(x) = s(x) - p(x) \implies \left\|s(x) - p(x)\right\|^2 = 0$$

Suppose $u_1 = u_2$, choose $v_1 = v$ and $v_2 = -v$ in Hessian condition so for all v,

$$\langle v(x)^2, s(x) - p(x)\rangle \geq 0 \implies \langle p(x), s(x) - p(x)\rangle \geq 0$$

However, $\langle s(x), s(x) - p(x)\rangle = 0$ (gradient condition), so $\left\|s(x) - p(x)\right\|^2 = 0$

Interpolate between these two cases with the Positivstellensatz

Numerical Implementation

TrigPolys.jl: a new package for fast manipulation of trigonometric polynomials

```
function Base.:*(p1::TrigPoly, p2::TrigPoly)
    n = p1.n + p2.n
    interpolate(evaluate(pad_to(p1, n)) .* evaluate(pad_to(p2, n)))
end
```

```
p1 = random_trig_poly(10^6)
p2 = random_trig_poly(10^6)
@btime p1 * p2;
```

```
1.737 s (160 allocations: 778.21 MiB)
```

`evaluate`, `evaluateT` and `interpolate` uses FFTW.jl, enables fast computation of f(U):

```
f(u) = sum((evaluate(pad_to(u, p.n)).^2 - evaluate(p)).^2)
```

AutoGrad.jl enables automatic computation of $\nabla f(U)$

```
AutoGrad.@primitive evaluate(u::AbstractArray),dy,y evaluateT(dy)
fgrad = AutoGrad.grad(f)
```

Pass f(U), ∇f(U) to NLopt.jl to minimize f(U) with first-order optimization algorithms

Conclusion

When does it make sense to solve non-convex formulations of convex problems?

In our setting we can prove non-convexity does not hurt us

Also enables fast implementation in Julia