Online algorithms for two problems in high-dimensional convex geometry

Naren Sarayu Manoj (https://nsmanoj.com)

TTIC

2023 Nov 06

Modern input concerns for algorithmic data science

- Research statement – https://narenmanoj.github.io/nsm_statement.pdf
- Publication list – https://narenmanoj.github.io/nsm_publist.pdf

Modern input concerns for algorithmic data science

Unwieldy input

- Research statement – https://narenmanoj.github.io/nsm_statement.pdf
- Publication list – https://narenmanoj.github.io/nsm_publist.pdf

Modern input concerns for algorithmic data science

Unwieldy input

- Approximating convex polytopes in a stream (Makarychev, Manoj, and Ovsiankin [MMO22; MMO23], COLT 2022/under submission).

- Research statement – https://narenmanoj.github.io/nsm_statement.pdf
- Publication list – https://narenmanoj.github.io/nsm_publist.pdf

Modern input concerns for algorithmic data science

Unwieldy input

- Approximating convex polytopes in a stream (Makarychev, Manoj, and Ovsiankin [MMO22; MMO23], COLT 2022/under submission).
- Approximating large convex objective functions (Manoj and Ovsiankin [MO23], under submission).

- Research statement – https://narenmanoj.github.io/nsm_statement.pdf
- Publication list – https://narenmanoj.github.io/nsm_publist.pdf

Modern input concerns for algorithmic data science

Unwieldy input

- Approximating convex polytopes in a stream (Makarychev, Manoj, and Ovsiankin [MMO22; MMO23], COLT 2022/under submission).
- Approximating large convex objective functions (Manoj and Ovsiankin [MO23], under submission).
- Explaining classifier predictions on large inputs (Gupta and Manoj [GM23], SOSA 2023).

- Research statement – https://narenmanoj.github.io/nsm_statement.pdf
- Publication list – https://narenmanoj.github.io/nsm_publist.pdf

Modern input concerns for algorithmic data science

Unwieldy input

- Approximating convex polytopes in a stream (Makarychev, Manoj, and Ovsiankin [MMO22; MMO23], COLT 2022/under submission).
- Approximating large convex objective functions (Manoj and Ovsiankin [MO23], under submission).
- Explaining classifier predictions on large inputs (Gupta and Manoj [GM23], SOSA 2023).

Unexpected input

- Research statement – https://narenmanoj.github.io/nsm_statement.pdf
- Publication list – https://narenmanoj.github.io/nsm_publist.pdf

Modern input concerns for algorithmic data science

Unwieldy input

- Approximating convex polytopes in a stream (Makarychev, Manoj, and Ovsiankin [MMO22; MMO23], COLT 2022/under submission).
- Approximating large convex objective functions (Manoj and Ovsiankin [MO23], under submission).
- Explaining classifier predictions on large inputs (Gupta and Manoj [GM23], SOSA 2023).

Unexpected input

- Robust machine learning under backdoor poisoning attacks (Manoj and Blum [MB21], NeurIPS 2021).

- Research statement – https://narenmanoj.github.io/nsm_statement.pdf
- Publication list – https://narenmanoj.github.io/nsm_publist.pdf

Modern input concerns for algorithmic data science

Unwieldy input

- Approximating convex polytopes in a stream (Makarychev, Manoj, and Ovsiankin [MMO22; MMO23], COLT 2022/under submission).
- Approximating large convex objective functions (Manoj and Ovsiankin [MO23], under submission).
- Explaining classifier predictions on large inputs (Gupta and Manoj [GM23], SOSA 2023).

Unexpected input

- Robust machine learning under backdoor poisoning attacks (Manoj and Blum [MB21], NeurIPS 2021).
- Learning from out-of-list feedback (Blum, Gupta, Li, Manoj, Saha, and Yang [BGLMSY23], under submission).

- Research statement – https://narenmanoj.github.io/nsm_statement.pdf
- Publication list – https://narenmanoj.github.io/nsm_publist.pdf

Modern input concerns for algorithmic data science

Unwieldy input

- Approximating convex polytopes in a stream (Makarychev, Manoj, and Ovsiankin [MMO22; MMO23], COLT 2022/under submission).
- Approximating large convex objective functions (Manoj and Ovsiankin [MO23], under submission).
- Explaining classifier predictions on large inputs (Gupta and Manoj [GM23], SOSA 2023).

Unexpected input

- Robust machine learning under backdoor poisoning attacks (Manoj and Blum [MB21], NeurIPS 2021).
- Learning from out-of-list feedback (Blum, Gupta, Li, Manoj, Saha, and Yang [BGLMSY23], under submission).
- Generalization of short-program interpolators (Manoj and Srebro [MS23], COLT 2023).

- Research statement – https://narenmanoj.github.io/nsm_statement.pdf
- Publication list – https://narenmanoj.github.io/nsm_publist.pdf

Modern input concerns for algorithmic data science

Unwieldy input

- Approximating convex polytopes in a stream (Makarychev, Manoj, and Ovsiankin [MMO22; MMO23], COLT 2022/under submission).
- Approximating large convex objective functions (Manoj and Ovsiankin [MO23], under submission).
- Explaining classifier predictions on large inputs (Gupta and Manoj [GM23], SOSA 2023).

Unexpected input

- Robust machine learning under backdoor poisoning attacks (Manoj and Blum [MB21], NeurIPS 2021).
- Learning from out-of-list feedback (Blum, Gupta, Li, Manoj, Saha, and Yang [BGLMSY23], under submission).
- Generalization of short-program interpolators (Manoj and Srebro [MS23], COLT 2023).

- Research statement – https://narenmanoj.github.io/nsm_statement.pdf
- Publication list – https://narenmanoj.github.io/nsm_publist.pdf

Table of contents

Streaming Algorithms for Ellipsoidal Approximation of Convex Polytopes

Conference of Learning Theory (COLT) 2022

Yury Makarychev, *NSM*, Max Ovsiankin

Symmetric ellipsoidal approximations

Basic problem

Given a convex body X, find an ellipsoid $\mathcal{E}$ such that ${}^{\mathcal{E}}/_{\alpha} \subseteq X \subseteq \mathcal{E}$ for a small α.

Symmetric ellipsoidal approximations

Basic problem

Given a convex body X, find an ellipsoid $\mathcal{E}$ such that ${}^{\mathcal{E}}/_{\alpha} \subseteq X \subseteq \mathcal{E}$ for a small α.

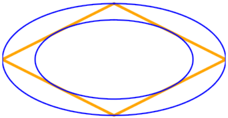


Figure: ${}^{\mathcal{E}}/\sqrt{2} \subseteq X \subseteq \mathcal{E}$

Symmetric ellipsoidal approximations

Basic problem

Given a convex body X, find an ellipsoid $\mathcal{E}$ such that $\mathcal{E}/\alpha \subseteq X \subseteq \mathcal{E}$ for a small α.

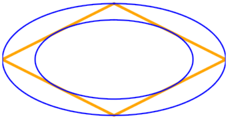


Figure: $\mathcal{E}/\sqrt{2} \subseteq X \subseteq \mathcal{E}$

Such an $\mathcal{E}$ has many uses, including...

Symmetric ellipsoidal approximations

Basic problem

Given a convex body X, find an ellipsoid $\mathcal{E}$ such that ${}^{\mathcal{E}}/_{\alpha} \subseteq X \subseteq \mathcal{E}$ for a small α.

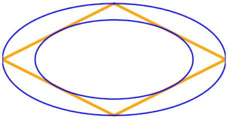


Figure: ${}^{\mathcal{E}}/_{\sqrt{2}} \subseteq X \subseteq \mathcal{E}$

Such an $\mathcal{E}$ has many uses, including...

- Obstacle detection in robotics [RB97]

Symmetric ellipsoidal approximations

Basic problem

Given a convex body X, find an ellipsoid $\mathcal{E}$ such that ${}^{\mathcal{E}}/_{\alpha} \subseteq X \subseteq \mathcal{E}$ for a small α.

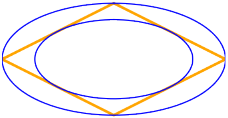
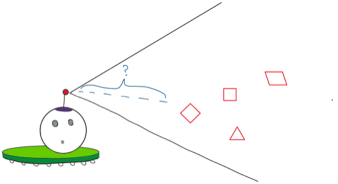


Figure: ${}^{\mathcal{E}}/_{\sqrt{2}} \subseteq X \subseteq \mathcal{E}$

Such an $\mathcal{E}$ has many uses, including...

- Obstacle detection in robotics [RB97]



Symmetric ellipsoidal approximations

Basic problem

Given a convex body X, find an ellipsoid $\mathcal{E}$ such that $\mathcal{E}/\alpha \subseteq X \subseteq \mathcal{E}$ for a small α.

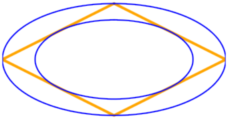
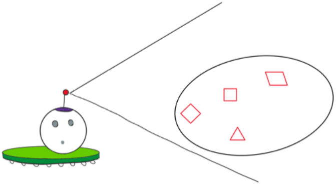


Figure: $\mathcal{E}/\sqrt{2} \subseteq X \subseteq \mathcal{E}$

Such an $\mathcal{E}$ has many uses, including...

- Obstacle detection in robotics [RB97]



Symmetric ellipsoidal approximations

Basic problem

Given a convex body X, find an ellipsoid $\mathcal{E}$ such that $\mathcal{E}/\alpha \subseteq X \subseteq \mathcal{E}$ for a small α.

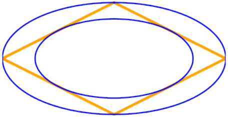
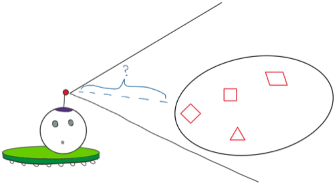


Figure: $\mathcal{E}/\sqrt{2} \subseteq X \subseteq \mathcal{E}$

Such an $\mathcal{E}$ has many uses, including...

- Obstacle detection in robotics [RB97]



Symmetric ellipsoidal approximations

Basic problem

Given a convex body X, find an ellipsoid $\mathcal{E}$ such that $\mathcal{E}/\alpha \subseteq X \subseteq \mathcal{E}$ for a small α.

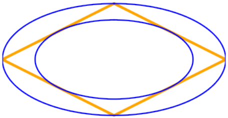
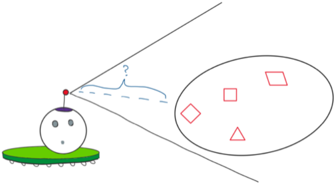


Figure: $\mathcal{E}/\sqrt{2} \subseteq X \subseteq \mathcal{E}$

Such an $\mathcal{E}$ has many uses, including...

- Obstacle detection in robotics [RB97]
- Online learning and optimization [LLS19]



Symmetric ellipsoidal approximations

Basic problem

Given a convex body X, find an ellipsoid $\mathcal{E}$ such that $\mathcal{E}/\alpha \subseteq X \subseteq \mathcal{E}$ for a small α.

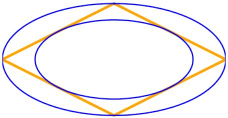
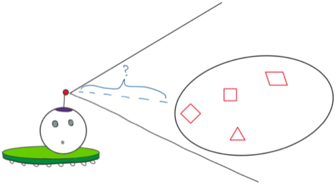


Figure: $\mathcal{E}/\sqrt{2} \subseteq X \subseteq \mathcal{E}$

Such an $\mathcal{E}$ has many uses, including...

- Obstacle detection in robotics [RB97]
- Online learning and optimization [LLS19]
- Succinctly representing a convex body (d^2 floats)



Offline solution

Symmetric ellipsoidal approximation – offline

Given a symmetric convex body $X \subseteq \mathbb{R}^d$, compute an ellipsoid $\mathcal{E}$ with $\mathcal{E}/\alpha \subseteq X \subseteq \mathcal{E}$ that minimizes α. Let α be $\mathcal{E}$'s *approximation factor*.

Offline solution

Symmetric ellipsoidal approximation – offline

Given a symmetric convex body $X \subseteq \mathbb{R}^d$, compute an ellipsoid $\mathcal{E}$ with $^{\mathcal{E}}/_{\alpha} \subseteq X \subseteq \mathcal{E}$ that minimizes α. Let α be $\mathcal{E}$'s *approximation factor*.

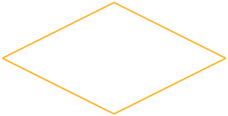


Figure: X

Offline solution

Symmetric ellipsoidal approximation – offline

Given a symmetric convex body $X \subseteq \mathbb{R}^d$, compute an ellipsoid $\mathcal{E}$ with $\mathcal{E}/\alpha \subseteq X \subseteq \mathcal{E}$ that minimizes α. Let α be $\mathcal{E}$'s *approximation factor*.

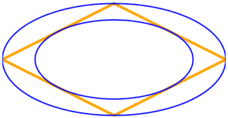


Figure: $\mathcal{E}/\sqrt{2} \subseteq X \subseteq \mathcal{E}$

Offline solution

Symmetric ellipsoidal approximation – offline

Given a symmetric convex body $X \subseteq \mathbb{R}^d$, compute an ellipsoid $\mathcal{E}$ with $\mathcal{E}/\alpha \subseteq X \subseteq \mathcal{E}$ that minimizes α. Let α be $\mathcal{E}$'s *approximation factor*.

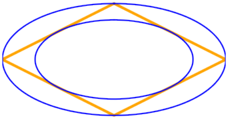


Figure: $\mathcal{E}/\sqrt{2} \subseteq X \subseteq \mathcal{E}$

Theorem of John [Joh48]

If $\mathcal{E}$ is the minimum volume ellipsoid covering X, then we can always achieve $\alpha \leq \sqrt{d}$.

Offline solution

Symmetric ellipsoidal approximation – offline

Given a symmetric convex body $X \subseteq \mathbb{R}^d$, compute an ellipsoid $\mathcal{E}$ with $\mathcal{E}/\alpha \subseteq X \subseteq \mathcal{E}$ that minimizes α. Let α be $\mathcal{E}$'s *approximation factor*.

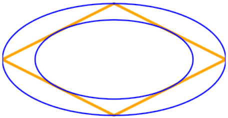


Figure: $\mathcal{E}/\sqrt{2} \subseteq X \subseteq \mathcal{E}$

Theorem of John [Joh48]

If $\mathcal{E}$ is the minimum volume ellipsoid covering X, then we can always achieve $\alpha \leq \sqrt{d}$.

There exists X for which the approximation factor $\sqrt{d}$ is tight for *any* ellipsoidal approximation for X (e.g. cross polytope, hypercube).

Offline solution

Symmetric ellipsoidal approximation – offline

Given a symmetric convex body $X \subseteq \mathbb{R}^d$, compute an ellipsoid $\mathcal{E}$ with ${}^{\mathcal{E}}/_{\alpha} \subseteq X \subseteq \mathcal{E}$ that minimizes α. Let α be $\mathcal{E}$'s *approximation factor*.

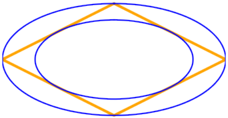


Figure: ${}^{\mathcal{E}}/_{\sqrt{2}} \subseteq X \subseteq \mathcal{E}$

Theorem of John [Joh48]

If $\mathcal{E}$ is the minimum volume ellipsoid covering X, then we can always achieve $\alpha \leq \sqrt{d}$.

There exists X for which the approximation factor $\sqrt{d}$ is tight for *any* ellipsoidal approximation for X (e.g. cross polytope, hypercube). For symmetric polytopes with n linear constraints, John's Ellipsoid can be approximated in time $\widetilde{O}\left(nd^2\right)$ [CCLY19].

Streaming/online ellipsoidal approximations

Problem

Given a symmetric convex body $X = \operatorname{conv}(\pm \boldsymbol{x}_1, \ldots, \pm \boldsymbol{x}_n)$ as a stream of points, find an ellipsoid $\mathcal{E}$ with $\mathcal{E}/\alpha \subseteq X \subseteq \mathcal{E}$ that minimizes α.

Streaming/online ellipsoidal approximations

Problem

Given a symmetric convex body $X = \operatorname{conv}(\pm \boldsymbol{x}_1, \ldots, \pm \boldsymbol{x}_n)$ as a stream of points, find an ellipsoid $\mathcal{E}$ with $\mathcal{E}/\alpha \subseteq X \subseteq \mathcal{E}$ that minimizes α.

$\boxed{\pm \boldsymbol{x}_1}$

Streaming/online ellipsoidal approximations

Problem

Given a symmetric convex body $X = \mathrm{conv}(\pm\boldsymbol{x}_1, \ldots, \pm\boldsymbol{x}_n)$ as a stream of points, find an ellipsoid $\mathcal{E}$ with $\mathcal{E}/\alpha \subseteq X \subseteq \mathcal{E}$ that minimizes α.

$$\boxed{\pm\boldsymbol{x}_1} \rightarrow \boxed{\pm\boldsymbol{x}_2}$$

Streaming/online ellipsoidal approximations

Problem

Given a symmetric convex body $X = \operatorname{conv}(\pm\boldsymbol{x}_1, \ldots, \pm\boldsymbol{x}_n)$ as a stream of points, find an ellipsoid $\mathcal{E}$ with ${}^{\mathcal{E}}/_{\alpha} \subseteq X \subseteq \mathcal{E}$ that minimizes α.

$$\boxed{\pm\boldsymbol{x}_1} \to \boxed{\pm\boldsymbol{x}_2} \to \cdots \to \boxed{\pm\boldsymbol{x}_n}$$

Streaming/online ellipsoidal approximations

Problem

Given a symmetric convex body $X = \mathrm{conv}(\pm \boldsymbol{x}_1, \ldots, \pm \boldsymbol{x}_n)$ as a stream of points, find an ellipsoid $\mathcal{E}$ with $\mathcal{E}/\alpha \subseteq X \subseteq \mathcal{E}$ that minimizes α.

$$\boxed{\pm \boldsymbol{x}_1} \to \boxed{\pm \boldsymbol{x}_2} \to \cdots \to \boxed{\pm \boldsymbol{x}_n}$$

Motivation – Suppose we want to summarize a dataset in a resource-constrained environment. Applications:

Streaming/online ellipsoidal approximations

Problem

Given a symmetric convex body $X = \mathrm{conv}(\pm\boldsymbol{x}_1, \ldots, \pm\boldsymbol{x}_n)$ as a stream of points, find an ellipsoid $\mathcal{E}$ with $\mathcal{E}/\alpha \subseteq X \subseteq \mathcal{E}$ that minimizes α.

$$\boxed{\pm\boldsymbol{x}_1} \rightarrow \boxed{\pm\boldsymbol{x}_2} \rightarrow \cdots \rightarrow \boxed{\pm\boldsymbol{x}_n}$$

Motivation – Suppose we want to summarize a dataset in a resource-constrained environment. Applications:

- Obstacle detection in robotics where the robot is processing using a mobile device

Streaming/online ellipsoidal approximations

Problem

Given a symmetric convex body $X = \operatorname{conv}(\pm \boldsymbol{x}_1, \ldots, \pm \boldsymbol{x}_n)$ as a stream of points, find an ellipsoid $\mathcal{E}$ with $\mathcal{E}/\alpha \subseteq X \subseteq \mathcal{E}$ that minimizes α.

$$\boxed{\pm \boldsymbol{x}_1} \to \boxed{\pm \boldsymbol{x}_2} \to \cdots \to \boxed{\pm \boldsymbol{x}_n}$$

Motivation – Suppose we want to summarize a dataset in a resource-constrained environment. Applications:

- Obstacle detection in robotics where the robot is processing using a mobile device
- A dataset is receiving updates on-the-fly, and a user needs to maintain a running ellipsoidal approximation of the dataset

Streaming/online ellipsoidal approximations

Problem

Given a symmetric convex body $X = \operatorname{conv}(\pm \boldsymbol{x}_1, \ldots, \pm \boldsymbol{x}_n)$ as a stream of points, find an ellipsoid $\mathcal{E}$ with $\mathcal{E}/\alpha \subseteq X \subseteq \mathcal{E}$ that minimizes α.

$$\boxed{\pm \boldsymbol{x}_1} \to \boxed{\pm \boldsymbol{x}_2} \to \cdots \to \boxed{\pm \boldsymbol{x}_n}$$

Motivation – Suppose we want to summarize a dataset in a resource-constrained environment. Applications:

- Obstacle detection in robotics where the robot is processing using a mobile device
- A dataset is receiving updates on-the-fly, and a user needs to maintain a running ellipsoidal approximation of the dataset
- We want our algorithms to be efficient:

Streaming/online ellipsoidal approximations

Problem

Given a symmetric convex body $X = \operatorname{conv}(\pm \boldsymbol{x}_1, \ldots, \pm \boldsymbol{x}_n)$ as a stream of points, find an ellipsoid $\mathcal{E}$ with $\mathcal{E}/\alpha \subseteq X \subseteq \mathcal{E}$ that minimizes α.

$$\boxed{\pm \boldsymbol{x}_1} \to \boxed{\pm \boldsymbol{x}_2} \to \cdots \to \boxed{\pm \boldsymbol{x}_n}$$

Motivation – Suppose we want to summarize a dataset in a resource-constrained environment. Applications:

- Obstacle detection in robotics where the robot is processing using a mobile device
- A dataset is receiving updates on-the-fly, and a user needs to maintain a running ellipsoidal approximation of the dataset
- We want our algorithms to be efficient:
  - Cannot store too many points at once;

Streaming/online ellipsoidal approximations

Problem

Given a symmetric convex body $X = \operatorname{conv}(\pm \boldsymbol{x}_1, \ldots, \pm \boldsymbol{x}_n)$ as a stream of points, find an ellipsoid $\mathcal{E}$ with $\mathcal{E}/\alpha \subseteq X \subseteq \mathcal{E}$ that minimizes α.

$$\boxed{\pm \boldsymbol{x}_1} \to \boxed{\pm \boldsymbol{x}_2} \to \cdots \to \boxed{\pm \boldsymbol{x}_n}$$

Motivation – Suppose we want to summarize a dataset in a resource-constrained environment. Applications:

- Obstacle detection in robotics where the robot is processing using a mobile device
- A dataset is receiving updates on-the-fly, and a user needs to maintain a running ellipsoidal approximation of the dataset
- We want our algorithms to be efficient:
  - Cannot store too many points at once;
  - Update time in each iteration must be fast.

Table of contents

Aspect ratio of convex body

Aspect ratio of convex body

Let R be the smallest value and r be the largest value such that:

$$r \cdot B_2^d \subseteq X \subseteq R \cdot B_2^d$$

The *aspect ratio* of X is $\kappa(X) := {}^R/_r$.

Aspect ratio of convex body

Let R be the smallest value and r be the largest value such that:

$$r \cdot B_2^d \subseteq X \subseteq R \cdot B_2^d$$

The *aspect ratio* of X is $\kappa(X) := {}^{R}\!/_{r}$.

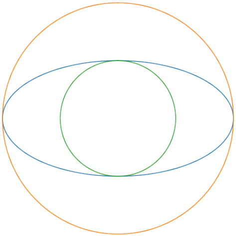


Figure: $1 \cdot B_2^d \subseteq \mathcal{E} \subseteq 2 \cdot B_2^d \Rightarrow \kappa(\mathcal{E}) = {}^{2}\!/_{1}$

Aspect ratio of convex body

Aspect ratio of convex body

Let R be the smallest value and r be the largest value such that:

$$r \cdot B_2^d \subseteq X \subseteq R \cdot B_2^d$$

The *aspect ratio* of X is $\kappa(X) := {}^{R}/_{r}$.

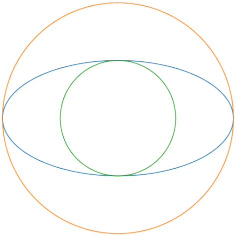


Figure: $1 \cdot B_2^d \subseteq \mathcal{E} \subseteq 2 \cdot B_2^d \Rightarrow \kappa(\mathcal{E}) = {}^{2}/_{1}$

Example

- For an ellipsoid $\mathcal{E}$ with major axis $\lambda_{\max}$ and minor axis $\lambda_{\min}$, we have $\kappa(\mathcal{E}) = {}^{\lambda_{\max}}/_{\lambda_{\min}}$;

Aspect ratio of convex body

Aspect ratio of convex body

Let R be the smallest value and r be the largest value such that:

$$r \cdot B_2^d \subseteq X \subseteq R \cdot B_2^d$$

The *aspect ratio* of X is $\kappa(X) := {}^R/_r$.

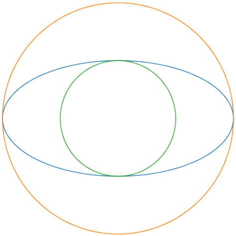


Figure: $1 \cdot B_2^d \subseteq \mathcal{E} \subseteq 2 \cdot B_2^d \Rightarrow \kappa(\mathcal{E}) = {}^2/_1$

Example

- For an ellipsoid $\mathcal{E}$ with major axis $\lambda_{\max}$ and minor axis $\lambda_{\min}$, we have $\kappa(\mathcal{E}) = {}^{\lambda_{\max}}/_{\lambda_{\min}}$;
- $\kappa(B_1^d) = \sqrt{d}$;

Aspect ratio of convex body

Aspect ratio of convex body

Let R be the smallest value and r be the largest value such that:

$$r \cdot B_2^d \subseteq X \subseteq R \cdot B_2^d$$

The *aspect ratio* of X is $\kappa(X) := {}^{R}/_{r}$.

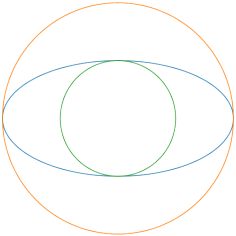


Figure: $1 \cdot B_2^d \subseteq \mathcal{E} \subseteq 2 \cdot B_2^d \Rightarrow \kappa(\mathcal{E}) = {}^{2}/_{1}$

Example

- For an ellipsoid $\mathcal{E}$ with major axis $\lambda_{\max}$ and minor axis $\lambda_{\min}$, we have $\kappa(\mathcal{E}) = {}^{\lambda_{\max}}/_{\lambda_{\min}}$;
- $\kappa(B_1^d) = \sqrt{d}$;
- $\kappa(B_2^d) = 1$

Greedy algorithm

Assumption

The algorithm is told a value of r such that $r \cdot B_2^d \subseteq X$.

Greedy algorithm

Assumption

The algorithm is told a value of r such that $r \cdot B_2^d \subseteq X$.

Initialization – Let $\mathcal{E}_0 = r \cdot B_2^d$.

Greedy algorithm

Assumption

The algorithm is told a value of r such that $r \cdot B_2^d \subseteq X$.

Initialization – Let $\mathcal{E}_0 = r \cdot B_2^d$.

Update rule – given ellipsoid $\mathcal{E}_{t-1}$ and new point $\pm \boldsymbol{x}_t$, let $\mathcal{E}_t$ be the minimum volume origin-centered ellipsoid covering $\mathcal{E}_{t-1}$ and $\pm \boldsymbol{x}_t$.

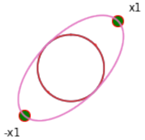
Greedy algorithm

Assumption

The algorithm is told a value of r such that $r \cdot B_2^d \subseteq X$.

Initialization – Let $\mathcal{E}_0 = r \cdot B_2^d$.

Update rule – given ellipsoid $\mathcal{E}_{t-1}$ and new point $\pm \boldsymbol{x}_t$, let $\mathcal{E}_t$ be the minimum volume origin-centered ellipsoid covering $\mathcal{E}_{t-1}$ and $\pm \boldsymbol{x}_t$.



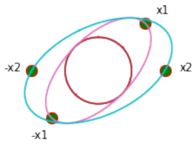
Greedy algorithm

Assumption

The algorithm is told a value of r such that $r \cdot B_2^d \subseteq X$.

Initialization – Let $\mathcal{E}_0 = r \cdot B_2^d$.

Update rule – given ellipsoid $\mathcal{E}_{t-1}$ and new point $\pm \boldsymbol{x}_t$, let $\mathcal{E}_t$ be the minimum volume origin-centered ellipsoid covering $\mathcal{E}_{t-1}$ and $\pm \boldsymbol{x}_t$.

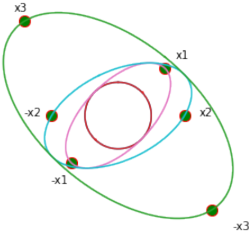


Assumption

The algorithm is told a value of r such that $r \cdot B_2^d \subseteq X$.

Initialization – Let $\mathcal{E}_0 = r \cdot B_2^d$.

Update rule – given ellipsoid $\mathcal{E}_{t-1}$ and new point $\pm \boldsymbol{x}_t$, let $\mathcal{E}_t$ be the minimum volume origin-centered ellipsoid covering $\mathcal{E}_{t-1}$ and $\pm \boldsymbol{x}_t$.

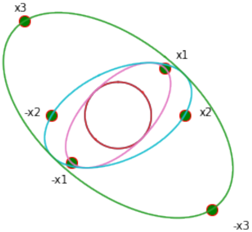


Assumption

The algorithm is told a value of r such that $r \cdot B_2^d \subseteq X$.

Initialization – Let $\mathcal{E}_0 = r \cdot B_2^d$.

Update rule – given ellipsoid $\mathcal{E}_{t-1}$ and new point $\pm \boldsymbol{x}_t$, let $\mathcal{E}_t$ be the minimum volume origin-centered ellipsoid covering $\mathcal{E}_{t-1}$ and $\pm \boldsymbol{x}_t$.



Formally, if $\mathcal{E}_t = \left\{ \boldsymbol{x} \ : \ \|\mathbf{A}_t \boldsymbol{x}\|_2 \leq 1 \right\}$ for invertible $\mathbf{A}_t$, then:

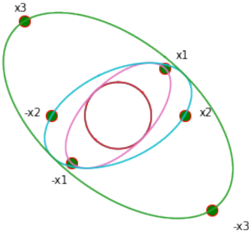
Greedy algorithm

Assumption

The algorithm is told a value of r such that $r \cdot B_2^d \subseteq X$.

Initialization – Let $\mathcal{E}_0 = r \cdot B_2^d$.

Update rule – given ellipsoid $\mathcal{E}_{t-1}$ and new point $\pm \boldsymbol{x}_t$, let $\mathcal{E}_t$ be the minimum volume origin-centered ellipsoid covering $\mathcal{E}_{t-1}$ and $\pm \boldsymbol{x}_t$.



Formally, if $\mathcal{E}_t = \left\{ \boldsymbol{x} \;:\; \|\mathbf{A}_t \boldsymbol{x}\|_2 \leq 1 \right\}$ for invertible $\mathbf{A}_t$, then:

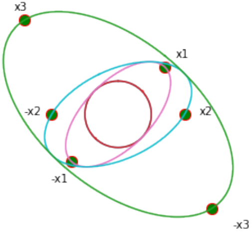
$$\mathbf{A}_t = \mathbf{A}_{t-1} - \left(1 - \frac{1}{\|\mathbf{A}_{t-1}\boldsymbol{x}_t\|_2}\right) \left(\frac{\mathbf{A}_{t-1}\boldsymbol{x}_t}{\|\mathbf{A}_{t-1}\boldsymbol{x}_t\|_2}\right) \left(\frac{\mathbf{A}_{t-1}\boldsymbol{x}_t}{\|\mathbf{A}_{t-1}\boldsymbol{x}_t\|_2}\right)^T \mathbf{A}_{t-1}$$

Assumption

The algorithm is told a value of r such that $r \cdot B_2^d \subseteq X$.

Initialization – Let $\mathcal{E}_0 = r \cdot B_2^d$.

Update rule – given ellipsoid $\mathcal{E}_{t-1}$ and new point $\pm \boldsymbol{x}_t$, let $\mathcal{E}_t$ be the minimum volume origin-centered ellipsoid covering $\mathcal{E}_{t-1}$ and $\pm \boldsymbol{x}_t$.



Formally, if $\mathcal{E}_t = \left\{ \boldsymbol{x} \ : \ \|\mathbf{A}_t \boldsymbol{x}\|_2 \leq 1 \right\}$ for invertible $\mathbf{A}_t$, then:

$$\mathbf{A}_t = \mathbf{A}_{t-1} - \underbrace{\left(1 - \frac{1}{\|\mathbf{A}_{t-1}\boldsymbol{x}_t\|_2}\right)}_{\text{scalar}} \underbrace{\left(\frac{\mathbf{A}_{t-1}\boldsymbol{x}_t}{\|\mathbf{A}_{t-1}\boldsymbol{x}_t\|_2}\right)}_{\text{vector}} \underbrace{\left(\frac{\mathbf{A}_{t-1}\boldsymbol{x}_t}{\|\mathbf{A}_{t-1}\boldsymbol{x}_t\|_2}\right)^T}_{\text{vector}} \mathbf{A}_{t-1}$$

Main result

Main result

Let r, R be such that $r \cdot B_2^d \subseteq X \subseteq R \cdot B_2^d$.

Main result

Main result

Let r, R be such that $r \cdot B_2^d \subseteq X \subseteq R \cdot B_2^d$.
The greedy algorithm gives an $O\left(\sqrt{d \log\left({}^{R}/_{r} + 1\right)}\right)$-approximation for the problem. Each iteration of the algorithm runs in time $O\left(d^2\right)$ and stores $O(d^2)$ floating-point numbers.

Main result

Main result

Let r, R be such that $r \cdot B_2^d \subseteq X \subseteq R \cdot B_2^d$.
The greedy algorithm gives an $O\left(\sqrt{d \log\left({}^{R}\!/_{r} + 1\right)}\right)$-approximation for the problem. Each iteration of the algorithm runs in time $O\left(d^2\right)$ and stores $O(d^2)$ floating-point numbers.

Optimality:

Main result

Main result

Let r, R be such that $r \cdot B_2^d \subseteq X \subseteq R \cdot B_2^d$.
The greedy algorithm gives an $O\left(\sqrt{d \log\left({}^{R}\!/_{r} + 1\right)}\right)$-approximation for the problem. Each iteration of the algorithm runs in time $O\left(d^2\right)$ and stores $O(d^2)$ floating-point numbers.

Optimality:

- Compared to the optimal offline solution, we lose only an excess $O(\sqrt{\log\left({}^{R}\!/_{r} + 1\right)})$ factor.

Main result

Main result

Let r, R be such that $r \cdot B_2^d \subseteq X \subseteq R \cdot B_2^d$.
The greedy algorithm gives an $O\left(\sqrt{d \log\left({}^{R}/_{r} + 1\right)}\right)$-approximation for the problem. Each iteration of the algorithm runs in time $O\left(d^2\right)$ and stores $O(d^2)$ floating-point numbers.

Optimality:

- Compared to the optimal offline solution, we lose only an excess $O(\sqrt{\log\left({}^{R}/_{r} + 1\right)})$ factor.
- We have strong evidence to suggest that the $\sqrt{\log\left({}^{R}/_{r} + 1\right)}$ extra loss is necessary.

Proof strategy

$\mathcal{E}$ is an α-approximation if $\mathcal{E}/\alpha \subseteq \operatorname{conv}(\pm \boldsymbol{x}_1, \ldots, \pm \boldsymbol{x}_n) \subseteq \mathcal{E}$.

Define:

$$E_{R/r} := \{\mathcal{E} \supseteq X \ : \ \mathcal{E} \text{ has aspect ratio at most } R/r\}$$

$$\tag{2}$$

Proof strategy

$\mathcal{E}$ is an α-approximation if ${}^{\mathcal{E}}/_{\alpha} \subseteq \operatorname{conv}(\pm \boldsymbol{x}_1, \ldots, \pm \boldsymbol{x}_n) \subseteq \mathcal{E}$.

Define:

$$E_{R/r} := \{\mathcal{E} \supseteq X \ : \ \mathcal{E} \text{ has aspect ratio at most } {}^{R}/_{r}\}$$

It is enough to show:

$$\tag{2}$$

Proof strategy

$\mathcal{E}$ is an α-approximation if ${}^{\mathcal{E}}/_{\alpha} \subseteq \operatorname{conv}(\pm \boldsymbol{x}_1, \ldots, \pm \boldsymbol{x}_n) \subseteq \mathcal{E}$.

Define:

$$E_{R/r} := \{\mathcal{E} \supseteq X \ : \ \mathcal{E} \text{ has aspect ratio at most } {}^{R}/_{r}\}$$

It is enough to show:

$$\mathcal{E}_n \subseteq C\sqrt{d \log\left({}^{R}/_{r} + 1\right)} \cdot \mathcal{E}, \text{ for all } \mathcal{E} \in E_{R/r} \tag{1}$$

$$\tag{2}$$

Proof strategy

$\mathcal{E}$ is an α-approximation if ${}^{\mathcal{E}}/_{\alpha} \subseteq \operatorname{conv}(\pm \boldsymbol{x}_1, \ldots, \pm \boldsymbol{x}_n) \subseteq \mathcal{E}$.

Define:

$$E_{R/r} := \{\mathcal{E} \supseteq X \ : \ \mathcal{E} \text{ has aspect ratio at most } {}^{R}/_{r}\}$$

It is enough to show:

$$\mathcal{E}_n \subseteq C\sqrt{d \log\left({}^{R}/_{r} + 1\right)} \cdot \mathcal{E}, \text{ for all } \mathcal{E} \in E_{R/r} \tag{1}$$

$$\bigcap_{\mathcal{E} \in E_{R/r}} \mathcal{E} \subseteq \sqrt{2} \cdot X \tag{2}$$

Proof strategy

$\mathcal{E}$ is an α-approximation if ${}^{\mathcal{E}}/_{\alpha} \subseteq \operatorname{conv}(\pm \boldsymbol{x}_1, \ldots, \pm \boldsymbol{x}_n) \subseteq \mathcal{E}$.

Define:

$$E_{R/r} := \{\mathcal{E} \supseteq X \ : \ \mathcal{E} \text{ has aspect ratio at most } {}^{R}/_{r}\}$$

It is enough to show:

$$\mathcal{E}_n \subseteq C\sqrt{d \log\left({}^{R}/_{r} + 1\right)} \cdot \mathcal{E}, \text{ for all } \mathcal{E} \in E_{R/r} \qquad (1)$$

$$\bigcap_{\mathcal{E} \in E_{R/r}} \mathcal{E} \subseteq \sqrt{2} \cdot X \qquad (2)$$

Then:

$$X \underset{\text{update rule}}{\subseteq} \mathcal{E}_n \underset{(1)}{\subseteq} \alpha \cdot \left(\bigcap_{\mathcal{E} \in E_{R/r}} \mathcal{E} \right) \underset{(2)}{\subseteq} \alpha\sqrt{2} \cdot X$$

Proof strategy

$\mathcal{E}$ is an α-approximation if ${}^{\mathcal{E}}/_{\alpha} \subseteq \operatorname{conv}(\pm \boldsymbol{x}_1, \ldots, \pm \boldsymbol{x}_n) \subseteq \mathcal{E}$.

Define:

$$E_{R/r} := \{\mathcal{E} \supseteq X \ : \ \mathcal{E} \text{ has aspect ratio at most } {}^{R}/_{r}\}$$

It is enough to show:

$$\mathcal{E}_n \subseteq C\sqrt{d \log\left({}^{R}/_{r} + 1\right)} \cdot \mathcal{E}, \text{ for all } \mathcal{E} \in E_{R/r} \tag{1}$$

$$\bigcap_{\mathcal{E} \in E_{R/r}} \mathcal{E} \subseteq \sqrt{2} \cdot X \tag{2}$$

Then:

$$X \underset{\text{update rule}}{\subseteq} \mathcal{E}_n \underset{(1)}{\subseteq} \alpha \cdot \left(\bigcap_{\mathcal{E} \in E_{R/r}} \mathcal{E} \right) \underset{(2)}{\subseteq} \alpha\sqrt{2} \cdot X$$

1. (1) follows from a potential function argument.

Proof strategy

$\mathcal{E}$ is an α-approximation if $\mathcal{E}/\alpha \subseteq \operatorname{conv}(\pm \boldsymbol{x}_1, \ldots, \pm \boldsymbol{x}_n) \subseteq \mathcal{E}$.

Define:

$$E_{R/r} := \{\mathcal{E} \supseteq X \ : \ \mathcal{E} \text{ has aspect ratio at most } R/r\}$$

It is enough to show:

$$\mathcal{E}_n \subseteq C\sqrt{d \log\left(R/r + 1\right)} \cdot \mathcal{E}, \text{ for all } \mathcal{E} \in E_{R/r} \tag{1}$$

$$\bigcap_{\mathcal{E} \in E_{R/r}} \mathcal{E} \subseteq \sqrt{2} \cdot X \tag{2}$$

Then:

$$X \underset{\text{update rule}}{\subseteq} \mathcal{E}_n \underset{(1)}{\subseteq} \alpha \cdot \left(\bigcap_{\mathcal{E} \in E_{R/r}} \mathcal{E} \right) \underset{(2)}{\subseteq} \alpha\sqrt{2} \cdot X$$

1. (1) follows from a potential function argument.
2. (2) can be verified directly.

Proof strategy

$\mathcal{E}$ is an α-approximation if ${}^{\mathcal{E}}/_{\alpha} \subseteq \operatorname{conv}(\pm \boldsymbol{x}_1, \ldots, \pm \boldsymbol{x}_n) \subseteq \mathcal{E}$.

Define:

$$E_{R/r} := \{\mathcal{E} \supseteq X \ : \ \mathcal{E} \text{ has aspect ratio at most } {}^{R}/_{r}\}$$

It is enough to show:

$$\mathcal{E}_n \subseteq C\sqrt{d \log\left({}^{R}/_{r} + 1\right)} \cdot \mathcal{E}, \text{ for all } \mathcal{E} \in E_{R/r} \tag{1}$$

$$\bigcap_{\mathcal{E} \in E_{R/r}} \mathcal{E} \subseteq \sqrt{2} \cdot X \tag{2}$$

Then:

$$X \underset{\text{update rule}}{\subseteq} \mathcal{E}_n \underset{(1)}{\subseteq} \alpha \cdot \left(\bigcap_{\mathcal{E} \in E_{R/r}} \mathcal{E} \right) \underset{(2)}{\subseteq} \alpha\sqrt{2} \cdot X$$

1. (1) follows from a potential function argument. Let's get an overview of how this works.
2. (2) can be verified directly.

Part 1 – Identifying an ellipsoid to track

Consider an ellipsoid $\mathcal{E}^\star \supseteq X$ such that $\kappa(\mathcal{E}^\star) \leq R/r$. Pick a matrix $\mathbf{J}^\star$ such that $\mathcal{E}^\star = \left\{x \ : \ \|\mathbf{J}^\star \boldsymbol{x}\|_2 \leq 1\right\}$.

Part 1 – Identifying an ellipsoid to track

Consider an ellipsoid $\mathcal{E}^\star \supseteq X$ such that $\kappa(\mathcal{E}^\star) \leq {}^R\!/_r$. Pick a matrix $\mathbf{J}^\star$ such that $\mathcal{E}^\star = \left\{x \;:\; \|\mathbf{J}^\star \boldsymbol{x}\|_2 \leq 1\right\}$.

Recall that $\mathbf{A}_n$ is such that $\mathcal{E}_n = \left\{\boldsymbol{x} \;:\; \|\mathbf{A}_n \boldsymbol{x}\|_2 \leq 1\right\}$.

Part 1 – Identifying an ellipsoid to track

Consider an ellipsoid $\mathcal{E}^\star \supseteq X$ such that $\kappa(\mathcal{E}^\star) \leq {}^R\!/_r$. Pick a matrix $\mathbf{J}^\star$ such that $\mathcal{E}^\star = \left\{x \ : \ \|\mathbf{J}^\star \boldsymbol{x}\|_2 \leq 1\right\}$.

Recall that $\mathbf{A}_n$ is such that $\mathcal{E}_n = \left\{\boldsymbol{x} \ : \ \|\mathbf{A}_n \boldsymbol{x}\|_2 \leq 1\right\}$.

Observation

If $\sigma_{\max}\left(\mathbf{J}^\star \mathbf{A}_n^{-1}\right) \leq \alpha$, then for all $\boldsymbol{x} \in \mathbb{R}^d$ we have $\|(\mathbf{J}^\star/\alpha)\,\boldsymbol{x}\|_2 \leq \|\mathbf{A}_n \boldsymbol{x}\|_2$.

Part 1 – Identifying an ellipsoid to track

Consider an ellipsoid $\mathcal{E}^\star \supseteq X$ such that $\kappa(\mathcal{E}^\star) \leq {}^R\!/_r$. Pick a matrix $\mathbf{J}^\star$ such that $\mathcal{E}^\star = \left\{x \ : \ \|\mathbf{J}^\star \boldsymbol{x}\|_2 \leq 1\right\}$.

Recall that $\mathbf{A}_n$ is such that $\mathcal{E}_n = \left\{\boldsymbol{x} \ : \ \|\mathbf{A}_n \boldsymbol{x}\|_2 \leq 1\right\}$.

Observation

If $\sigma_{\max}\left(\mathbf{J}^\star \mathbf{A}_n^{-1}\right) \leq \alpha$, then for all $\boldsymbol{x} \in \mathbb{R}^d$ we have $\|(\mathbf{J}^\star/\alpha)\,\boldsymbol{x}\|_2 \leq \|\mathbf{A}_n \boldsymbol{x}\|_2$. In other words, $\mathcal{E}_n \subseteq \alpha \cdot \mathcal{E}^\star$.

Consider an ellipsoid $\mathcal{E}^\star \supseteq X$ such that $\kappa(\mathcal{E}^\star) \leq R/r$. Pick a matrix $\mathbf{J}^\star$ such that $\mathcal{E}^\star = \left\{x \ : \ \|\mathbf{J}^\star \boldsymbol{x}\|_2 \leq 1\right\}$.

Recall that $\mathbf{A}_n$ is such that $\mathcal{E}_n = \left\{\boldsymbol{x} \ : \ \|\mathbf{A}_n \boldsymbol{x}\|_2 \leq 1\right\}$.

Observation

If $\sigma_{\max}\left(\mathbf{J}^\star \mathbf{A}_n^{-1}\right) \leq \alpha$, then for all $\boldsymbol{x} \in \mathbb{R}^d$ we have $\|(\mathbf{J}^\star/\alpha)\,\boldsymbol{x}\|_2 \leq \|\mathbf{A}_n \boldsymbol{x}\|_2$. In other words, $\mathcal{E}_n \subseteq \alpha \cdot \mathcal{E}^\star$.

Interpretation – $\mathbf{J}^\star \mathbf{A}_n^{-1}$ describes the image of $\mathcal{E}_n$ where we transform the space such that $\mathcal{E}^\star$ corresponds to the unit ball.

Part 1 – Identifying an ellipsoid to track

Consider an ellipsoid $\mathcal{E}^\star \supseteq X$ such that $\kappa(\mathcal{E}^\star) \leq {}^R\!/_r$. Pick a matrix $\mathbf{J}^\star$ such that $\mathcal{E}^\star = \left\{x \ : \ \|\mathbf{J}^\star \boldsymbol{x}\|_2 \leq 1\right\}$.

Recall that $\mathbf{A}_n$ is such that $\mathcal{E}_n = \left\{\boldsymbol{x} \ : \ \|\mathbf{A}_n \boldsymbol{x}\|_2 \leq 1\right\}$.

Observation

If $\sigma_{\max}\left(\mathbf{J}^\star \mathbf{A}_n^{-1}\right) \leq \alpha$, then for all $\boldsymbol{x} \in \mathbb{R}^d$ we have $\|({}^{\mathbf{J}^\star}\!/_\alpha)\, \boldsymbol{x}\|_2 \leq \|\mathbf{A}_n \boldsymbol{x}\|_2$. In other words, $\mathcal{E}_n \subseteq \alpha \cdot \mathcal{E}^\star$.

Interpretation – $\mathbf{J}^\star \mathbf{A}_n^{-1}$ describes the image of $\mathcal{E}_n$ where we transform the space such that $\mathcal{E}^\star$ corresponds to the unit ball.

Upshot – it's sufficient to show that $\sigma_{\max}\left(\mathbf{J}^\star \mathbf{A}_n^{-1}\right) \lesssim \sqrt{d \log\left({}^R\!/_r + 1\right)}$.

Goal – Identify some useful $\Phi(\mathcal{E}_t)$ and claim that
$\sigma_{\max}\left(\mathbf{J}^\star \cdot A_n^{-1}\right)^2 \leq \Phi(\mathcal{E}_n) \leq \Phi(\mathcal{E}_{n-1}) \leq \cdots \leq \Phi(\mathcal{E}_0) \lesssim d \log\left({}^{R}/_{r} + 1\right)$.

Defining the potential

Part 2 – Potential functions

Goal – Identify some useful $\Phi(\mathcal{E}_t)$ and claim that
$\sigma_{\max}\left(\mathbf{J}^\star \cdot A_n^{-1}\right)^2 \leq \Phi(\mathcal{E}_n) \leq \Phi(\mathcal{E}_{n-1}) \leq \cdots \leq \Phi(\mathcal{E}_0) \lesssim d \log\left({}^{R}/_{r} + 1\right)$.

Defining the potential

Let $\mathbf{A}_t$ be an invertible transformation such that $\mathcal{E}_t = \left\{ \boldsymbol{x} \;:\; \|\mathbf{A}_t \boldsymbol{x}\|_2 \leq 1 \right\}$.

Part 2 – Potential functions

Goal – Identify some useful $\Phi(\mathcal{E}_t)$ and claim that
$\sigma_{\max}\left(\mathbf{J}^\star \cdot A_n^{-1}\right)^2 \leq \Phi(\mathcal{E}_n) \leq \Phi(\mathcal{E}_{n-1}) \leq \cdots \leq \Phi(\mathcal{E}_0) \lesssim d \log\left({}^{R}/_{r} + 1\right)$.

Defining the potential

Let $\mathbf{A}_t$ be an invertible transformation such that $\mathcal{E}_t = \left\{\boldsymbol{x} : \|\mathbf{A}_t \boldsymbol{x}\|_2 \leq 1\right\}$. Then:

$$\Phi_{\mathbf{J}^\star}(\mathcal{E}_t) := \left\|\mathbf{J}^\star \mathbf{A}_t^{-1}\right\|_F^2 - 2 \log\det\left(\mathbf{J}^\star \mathbf{A}_t^{-1}\right)$$

Part 2 – Potential functions

Goal – Identify some useful $\Phi(\mathcal{E}_t)$ and claim that
$\sigma_{\max}\left(\mathbf{J}^\star \cdot A_n^{-1}\right)^2 \leq \Phi(\mathcal{E}_n) \leq \Phi(\mathcal{E}_{n-1}) \leq \cdots \leq \Phi(\mathcal{E}_0) \lesssim d \log\left({}^R\!/_r + 1\right)$.

Defining the potential

Let $\mathbf{A}_t$ be an invertible transformation such that $\mathcal{E}_t = \left\{\boldsymbol{x} : \|\mathbf{A}_t\boldsymbol{x}\|_2 \leq 1\right\}$. Then:

$$\Phi_{\mathbf{J}^\star}(\mathcal{E}_t) := \left\|\mathbf{J}^\star\mathbf{A}_t^{-1}\right\|_F^2 - 2\log\det\left(\mathbf{J}^\star\mathbf{A}_t^{-1}\right) = \sum_{i=1}^{d} \sigma_{i,t}^2 - \log\left(\sigma_{i,t}^2\right)$$

Interpretation – the singular values of $\mathbf{J}^\star\mathbf{A}_t^{-1}$ correspond to the lengths of the axes of $\mathbf{J}^\star \cdot \mathcal{E}_t$. Thus, $\Phi_{\mathbf{J}^\star}(\mathcal{E}_n) \gtrsim \sigma_{\max}\left(\mathbf{J}^\star\mathbf{A}_n^{-1}\right)^2$.

Goal – Identify some useful $\Phi(\mathcal{E}_t)$ and claim that
$\sigma_{\max}\left(\mathbf{J}^\star \cdot A_n^{-1}\right)^2 \le \Phi(\mathcal{E}_n) \le \Phi(\mathcal{E}_{n-1}) \le \cdots \le \Phi(\mathcal{E}_0) \lesssim d \log\left({}^R/_r + 1\right)$.

Defining the potential

Let $\mathbf{A}_t$ be an invertible transformation such that $\mathcal{E}_t = \left\{\boldsymbol{x} : \|\mathbf{A}_t\boldsymbol{x}\|_2 \le 1\right\}$. Then:

$$\Phi_{\mathbf{J}^\star}(\mathcal{E}_t) := \left\|\mathbf{J}^\star\mathbf{A}_t^{-1}\right\|_F^2 - 2\log\det\left(\mathbf{J}^\star\mathbf{A}_t^{-1}\right) = \sum_{i=1}^{d} \sigma_{i,t}^2 - \log\left(\sigma_{i,t}^2\right)$$

Interpretation – the singular values of $\mathbf{J}^\star\mathbf{A}_t^{-1}$ correspond to the lengths of the axes of $\mathbf{J}^\star \cdot \mathcal{E}_t$. Thus, $\Phi_{\mathbf{J}^\star}(\mathcal{E}_n) \gtrsim \sigma_{\max}\left(\mathbf{J}^\star\mathbf{A}_n^{-1}\right)^2$.

Next, we show:

Part 2 – Potential functions

Goal – Identify some useful $\Phi(\mathcal{E}_t)$ and claim that
$\sigma_{\max}\left(\mathbf{J}^\star \cdot A_n^{-1}\right)^2 \leq \Phi(\mathcal{E}_n) \leq \Phi(\mathcal{E}_{n-1}) \leq \cdots \leq \Phi(\mathcal{E}_0) \lesssim d \log\left({}^{R}/{}_{r} + 1\right)$.

Defining the potential

Let $\mathbf{A}_t$ be an invertible transformation such that $\mathcal{E}_t = \left\{\boldsymbol{x} : \|\mathbf{A}_t \boldsymbol{x}\|_2 \leq 1\right\}$. Then:

$$\Phi_{\mathbf{J}^\star}(\mathcal{E}_t) := \left\|\mathbf{J}^\star \mathbf{A}_t^{-1}\right\|_F^2 - 2 \log\det\left(\mathbf{J}^\star \mathbf{A}_t^{-1}\right) = \sum_{i=1}^{d} \sigma_{i,t}^2 - \log\left(\sigma_{i,t}^2\right)$$

Interpretation – the singular values of $\mathbf{J}^\star \mathbf{A}_t^{-1}$ correspond to the lengths of the axes of $\mathbf{J}^\star \cdot \mathcal{E}_t$. Thus, $\Phi_{\mathbf{J}^\star}(\mathcal{E}_n) \gtrsim \sigma_{\max}\left(\mathbf{J}^\star \mathbf{A}_n^{-1}\right)^2$.

Next, we show:

$$\frac{\left\|\mathbf{J}^\star \mathbf{A}_t^{-1}\right\|_F^2 - \left\|\mathbf{J}^\star \mathbf{A}_{t-1}^{-1}\right\|_F^2}{2\log\det\left(\mathbf{J}^\star \mathbf{A}_t^{-1}\right) - 2\log\det\left(\mathbf{J}^\star \mathbf{A}_{t-1}^{-1}\right)} = \frac{1 - \frac{1}{\left\|\mathbf{A}_{t-1}\boldsymbol{x}_t\right\|_2^2}}{2\log\left\|\mathbf{A}_{t-1}\boldsymbol{x}_t\right\|_2} \cdot \left\|\mathbf{J}^\star \boldsymbol{x}_t\right\|^2$$

Part 2 – Potential functions

Goal – Identify some useful $\Phi(\mathcal{E}_t)$ and claim that
$\sigma_{\max}\left(\mathbf{J}^\star \cdot A_n^{-1}\right)^2 \leq \Phi(\mathcal{E}_n) \leq \Phi(\mathcal{E}_{n-1}) \leq \cdots \leq \Phi(\mathcal{E}_0) \lesssim d \log\left(R/r + 1\right)$.

Defining the potential

Let $\mathbf{A}_t$ be an invertible transformation such that $\mathcal{E}_t = \left\{ \boldsymbol{x} : \|\mathbf{A}_t \boldsymbol{x}\|_2 \leq 1 \right\}$. Then:

$$\Phi_{\mathbf{J}^\star}(\mathcal{E}_t) := \left\|\mathbf{J}^\star \mathbf{A}_t^{-1}\right\|_F^2 - 2 \log\det\left(\mathbf{J}^\star \mathbf{A}_t^{-1}\right) = \sum_{i=1}^{d} \sigma_{i,t}^2 - \log\left(\sigma_{i,t}^2\right)$$

Interpretation – the singular values of $\mathbf{J}^\star \mathbf{A}_t^{-1}$ correspond to the lengths of the axes of $\mathbf{J}^\star \cdot \mathcal{E}_t$. Thus, $\Phi_{\mathbf{J}^\star}(\mathcal{E}_n) \gtrsim \sigma_{\max}\left(\mathbf{J}^\star \mathbf{A}_n^{-1}\right)^2$.

Next, we show:

$$\frac{\left\|\mathbf{J}^\star \mathbf{A}_t^{-1}\right\|_F^2 - \left\|\mathbf{J}^\star \mathbf{A}_{t-1}^{-1}\right\|_F^2}{2\log\det\left(\mathbf{J}^\star \mathbf{A}_t^{-1}\right) - 2\log\det\left(\mathbf{J}^\star \mathbf{A}_{t-1}^{-1}\right)} = \frac{1 - \frac{1}{\left\|\mathbf{A}_{t-1}\boldsymbol{x}_t\right\|_2^2}}{2\log\left\|\mathbf{A}_{t-1}\boldsymbol{x}_t\right\|_2} \cdot \left\|\mathbf{J}^\star \boldsymbol{x}_t\right\|^2 \leq 1$$

Part 2 – Potential functions

Goal – Identify some useful $\Phi(\mathcal{E}_t)$ and claim that
$\sigma_{\max}\left(\mathbf{J}^\star \cdot A_n^{-1}\right)^2 \le \Phi(\mathcal{E}_n) \le \Phi(\mathcal{E}_{n-1}) \le \cdots \le \Phi(\mathcal{E}_0) \lesssim d \log\left({}^R/_r + 1\right)$.

Defining the potential

Let $\mathbf{A}_t$ be an invertible transformation such that $\mathcal{E}_t = \left\{\boldsymbol{x} : \|\mathbf{A}_t\boldsymbol{x}\|_2 \le 1\right\}$. Then:

$$\Phi_{\mathbf{J}^\star}(\mathcal{E}_t) := \left\|\mathbf{J}^\star\mathbf{A}_t^{-1}\right\|_F^2 - 2\log\det\left(\mathbf{J}^\star\mathbf{A}_t^{-1}\right) = \sum_{i=1}^{d} \sigma_{i,t}^2 - \log\left(\sigma_{i,t}^2\right)$$

Interpretation – the singular values of $\mathbf{J}^\star\mathbf{A}_t^{-1}$ correspond to the lengths of the axes of $\mathbf{J}^\star \cdot \mathcal{E}_t$. Thus, $\Phi_{\mathbf{J}^\star}(\mathcal{E}_n) \gtrsim \sigma_{\max}\left(\mathbf{J}^\star\mathbf{A}_n^{-1}\right)^2$.

Next, we show:

$$\frac{\left\|\mathbf{J}^\star\mathbf{A}_t^{-1}\right\|_F^2 - \left\|\mathbf{J}^\star\mathbf{A}_{t-1}^{-1}\right\|_F^2}{2\log\det\left(\mathbf{J}^\star\mathbf{A}_t^{-1}\right) - 2\log\det\left(\mathbf{J}^\star\mathbf{A}_{t-1}^{-1}\right)} = \frac{1 - \frac{1}{\left\|\mathbf{A}_{t-1}\boldsymbol{x}_t\right\|_2^2}}{2\log\left\|\mathbf{A}_{t-1}\boldsymbol{x}_t\right\|_2} \cdot \left\|\mathbf{J}^\star\boldsymbol{x}_t\right\|^2 \le 1$$

$$\Rightarrow \Phi_{\mathbf{J}^\star}(\mathcal{E}_t) \le \Phi_{\mathbf{J}^\star}(\mathcal{E}_{t-1})$$

Defining the potential

Let $\mathbf{A}_t$ be an invertible transformation such that $\mathcal{E}_t = \left\{x : \|\mathbf{A}_t \boldsymbol{x}\|_2 \leq 1\right\}$. Then:

$$\Phi_{\mathbf{J}^\star}(\mathcal{E}_t) := \left\|\mathbf{J}^\star \mathbf{A}_t^{-1}\right\|_F^2 - 2 \log\det\left(\mathbf{J}^\star \mathbf{A}_t^{-1}\right)$$

Part 2 – Potential functions

Defining the potential

Let $\mathbf{A}_t$ be an invertible transformation such that $\mathcal{E}_t = \left\{x : \|\mathbf{A}_t \boldsymbol{x}\|_2 \leq 1\right\}$. Then:

$$\Phi_{\mathbf{J}^\star}(\mathcal{E}_t) := \left\|\mathbf{J}^\star \mathbf{A}_t^{-1}\right\|_F^2 - 2 \log\det\left(\mathbf{J}^\star \mathbf{A}_t^{-1}\right)$$

We have:

$$\sigma_{\max}\left(\mathbf{J}^\star \mathbf{A}_n^{-1}\right)^2 \lesssim \Phi_{\mathbf{J}^\star}(\mathcal{E}_n) \leq \cdots \leq \Phi_{\mathbf{J}^\star}(\mathcal{E}_0)$$

Defining the potential

Let $\mathbf{A}_t$ be an invertible transformation such that $\mathcal{E}_t = \left\{x : \left\|\mathbf{A}_t \boldsymbol{x}\right\|_2 \leq 1\right\}$. Then:

$$\Phi_{\mathbf{J}^\star}(\mathcal{E}_t) := \left\|\mathbf{J}^\star \mathbf{A}_t^{-1}\right\|_F^2 - 2 \log \det \left(\mathbf{J}^\star \mathbf{A}_t^{-1}\right)$$

We have:

$$\sigma_{\max}\left(\mathbf{J}^\star \mathbf{A}_n^{-1}\right)^2 \lesssim \Phi_{\mathbf{J}^\star}(\mathcal{E}_n) \leq \cdots \leq \Phi_{\mathbf{J}^\star}(\mathcal{E}_0)$$

Suffices to control the RHS.

Part 3 – Evaluating initial potential function value

1. Establish $\mathbf{A}_0^{-1} = r\mathbf{I}$.

Part 3 – Evaluating initial potential function value

1. Establish $\mathbf{A}_0^{-1} = r\mathbf{I}$.
2. Notice that $\mathbf{J}^\star \mathbf{A}_0^{-1} = r\mathbf{J}^\star$.

Part 3 – Evaluating initial potential function value

1. Establish $\mathbf{A}_0^{-1} = r\mathbf{I}$.
2. Notice that $\mathbf{J}^\star \mathbf{A}_0^{-1} = r\mathbf{J}^\star$.
3. Establish:

$$\sigma_i\left(\mathbf{J}^\star \mathbf{A}_0^{-1}\right) = r\sigma_i\left(\mathbf{J}^\star\right) \in \left[\left({}^r\!/\!_R\right)^2, 1\right]$$

Part 3 – Evaluating initial potential function value

1. Establish $\mathbf{A}_0^{-1} = r\mathbf{I}$.
2. Notice that $\mathbf{J}^\star \mathbf{A}_0^{-1} = r\mathbf{J}^\star$.
3. Establish:

$$\sigma_i\left(\mathbf{J}^\star \mathbf{A}_0^{-1}\right) = r\sigma_i\left(\mathbf{J}^\star\right) \in \left[\left({}^{r}/_{R}\right)^2, 1\right]$$

4. Calculate $\Phi_{\mathbf{J}^\star}(\mathcal{E}_0)$:

Part 3 – Evaluating initial potential function value

1. Establish $\mathbf{A}_0^{-1} = r\mathbf{I}$.
2. Notice that $\mathbf{J}^\star \mathbf{A}_0^{-1} = r\mathbf{J}^\star$.
3. Establish:

$$\sigma_i\left(\mathbf{J}^\star \mathbf{A}_0^{-1}\right) = r\sigma_i\left(\mathbf{J}^\star\right) \in \left[\left({}^r\!/_R\right)^2, 1\right]$$

4. Calculate $\Phi_{\mathbf{J}^\star}(\mathcal{E}_0)$:

$$\begin{aligned}\Phi_{\mathbf{J}^\star}(\mathcal{E}_0) &= \left\|\mathbf{J}^\star \mathbf{A}_t^{-1}\right\|_F^2 - 2\log\det\left(\mathbf{J}^\star \mathbf{A}_t^{-1}\right) \\ &= \sum_{i=1}^{d}\left(\sigma_i\left(\mathbf{J}^\star \mathbf{A}_0^{-1}\right)^2 - \log\left(\sigma_i\left(\mathbf{J}^\star \mathbf{A}_0^{-1}\right)^2\right)\right) \\ &\lesssim d\log\left({}^R\!/_r + 1\right)\end{aligned}$$

Part 3 – Evaluating initial potential function value

1. Establish $\mathbf{A}_0^{-1} = r\mathbf{I}$.
2. Notice that $\mathbf{J}^\star \mathbf{A}_0^{-1} = r\mathbf{J}^\star$.
3. Establish:

$$\sigma_i\left(\mathbf{J}^\star \mathbf{A}_0^{-1}\right) = r\sigma_i\left(\mathbf{J}^\star\right) \in \left[\left({}^r\!/_R\right)^2, 1\right]$$

4. Calculate $\Phi_{\mathbf{J}^\star}(\mathcal{E}_0)$:

$$\begin{aligned}\Phi_{\mathbf{J}^\star}(\mathcal{E}_0) &= \left\|\mathbf{J}^\star \mathbf{A}_t^{-1}\right\|_F^2 - 2\log\det\left(\mathbf{J}^\star \mathbf{A}_t^{-1}\right) \\ &= \sum_{i=1}^{d}\left(\sigma_i\left(\mathbf{J}^\star \mathbf{A}_0^{-1}\right)^2 - \log\left(\sigma_i\left(\mathbf{J}^\star \mathbf{A}_0^{-1}\right)^2\right)\right) \\ &\lesssim d\log\left({}^R\!/_r + 1\right)\end{aligned}$$

We now have:

$$\sigma_{\max}\left(\mathbf{J}^\star \mathbf{A}_n^{-1}\right)^2 \lesssim \Phi_{\mathbf{J}^\star}(\mathcal{E}_n) \leq \cdots \leq \Phi_{\mathbf{J}^\star}(\mathcal{E}_0) \lesssim d\log\left({}^R\!/_r + 1\right)$$

Part 3 – Evaluating initial potential function value

1. Establish $\mathbf{A}_0^{-1} = r\mathbf{I}$.
2. Notice that $\mathbf{J}^\star \mathbf{A}_0^{-1} = r\mathbf{J}^\star$.
3. Establish:

$$\sigma_i\left(\mathbf{J}^\star \mathbf{A}_0^{-1}\right) = r\sigma_i\left(\mathbf{J}^\star\right) \in \left[\left({}^{r}\!/_{R}\right)^2, 1\right]$$

4. Calculate $\Phi_{\mathbf{J}^\star}(\mathcal{E}_0)$:

$$\begin{aligned}\Phi_{\mathbf{J}^\star}(\mathcal{E}_0) &= \left\|\mathbf{J}^\star \mathbf{A}_t^{-1}\right\|_F^2 - 2\log\det\left(\mathbf{J}^\star \mathbf{A}_t^{-1}\right) \\ &= \sum_{i=1}^{d}\left(\sigma_i\left(\mathbf{J}^\star \mathbf{A}_0^{-1}\right)^2 - \log\left(\sigma_i\left(\mathbf{J}^\star \mathbf{A}_0^{-1}\right)^2\right)\right) \\ &\lesssim d\log\left({}^{R}\!/_{r} + 1\right)\end{aligned}$$

We now have:

$$\sigma_{\max}\left(\mathbf{J}^\star \mathbf{A}_n^{-1}\right)^2 \lesssim \Phi_{\mathbf{J}^\star}(\mathcal{E}_n) \leq \cdots \leq \Phi_{\mathbf{J}^\star}(\mathcal{E}_0) \lesssim d\log\left({}^{R}\!/_{r} + 1\right)$$

This is enough!

Table of contents

Tracking the minimum volume outer ellipsoid

Natural question – how closely does our solution track the minimum volume outer ellipsoid for X?

Tracking the minimum volume outer ellipsoid

Natural question – how closely does our solution track the minimum volume outer ellipsoid for X?

Inapproximability of minimum volume ellipsoid

Let J be the minimum volume outer ellipsoid covering X.
Every one-pass algorithm that must output a sequence of ellipsoids $\mathcal{E}_i$ satisfying $\mathcal{E}_i \subseteq \mathcal{E}_j$ for all $i \leq j$ must have a $\sqrt{d}$-approximation factor *with respect to* J.

Tracking the minimum volume outer ellipsoid

Natural question – how closely does our solution track the minimum volume outer ellipsoid for X?

Inapproximability of minimum volume ellipsoid

Let J be the minimum volume outer ellipsoid covering X.
Every one-pass algorithm that must output a sequence of ellipsoids $\mathcal{E}_i$ satisfying $\mathcal{E}_i \subseteq \mathcal{E}_j$ for all $i \leq j$ must have a $\sqrt{d}$-approximation factor *with respect to* J.

Upshot – No "monotone" one-pass algorithm can have an output close to the true minimum-volume outer ellipsoid for every sequence of inputs.

Hadamard basis

For a dimension d, we say there exists a Hadamard basis for $\mathbb{R}^d$ if we can find a collection of d vectors $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d\}$ such that $\boldsymbol{v}_i \in \{\pm 1\}^d$ and for all $i \neq j$, we have $\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle = 0$.

Hard Instance – Outline

Hadamard basis

For a dimension d, we say there exists a Hadamard basis for $\mathbb{R}^d$ if we can find a collection of d vectors $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d\}$ such that $\boldsymbol{v}_i \in \{\pm 1\}^d$ and for all $i \neq j$, we have $\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle = 0$.

Let d be some dimension for which there exists a Hadamard basis $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d$.

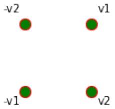
Hadamard basis

For a dimension d, we say there exists a Hadamard basis for $\mathbb{R}^d$ if we can find a collection of d vectors $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d\}$ such that $\boldsymbol{v}_i \in \{\pm 1\}^d$ and for all $i \neq j$, we have $\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle = 0$.

Let d be some dimension for which there exists a Hadamard basis $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d$.

Phase 1

Give the points $\boldsymbol{v}_1/\sqrt{d}, \ldots, \boldsymbol{v}_d/\sqrt{d}$.



Hard Instance – Outline

Hadamard basis

For a dimension d, we say there exists a Hadamard basis for $\mathbb{R}^d$ if we can find a collection of d vectors $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d\}$ such that $\boldsymbol{v}_i \in \{\pm 1\}^d$ and for all $i \neq j$, we have $\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle = 0$.

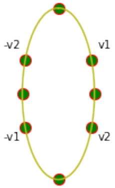
Let d be some dimension for which there exists a Hadamard basis $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d$.

Phase 1

Give the points ${}^{\boldsymbol{v}_1}/_{\sqrt{d}}, \ldots, {}^{\boldsymbol{v}_d}/_{\sqrt{d}}$.

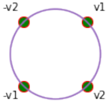
Phase 2

The instance selects $i \in [d]$ and $\varepsilon \in (0, d-1)$. Let $\boldsymbol{w}_i = \boldsymbol{e}_i \cdot {}^1/_{\sqrt{d-\varepsilon}}$ and $\boldsymbol{w}_j = \boldsymbol{e}_j \cdot \sqrt{{}^{d-1}/_{\varepsilon}}$, for all $j \neq i$. Give the points $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_d$.



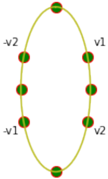
Hard Instance – Analysis Outline

- At the end of Phase 1, the minimum volume outer ellipsoid is B_2^d.



- At the end of Phase 1, the minimum volume outer ellipsoid is B_2^d.
- At the end of Phase 2, the final minimum volume outer ellipsoid is:

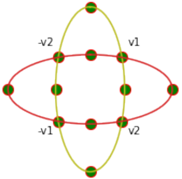
$$\left\{ \boldsymbol{x} \ : \ 1 \geq \frac{\boldsymbol{x}_i^2}{\left({}^1\!/\!\sqrt{d-\varepsilon}\right)^2} + \sum_{j \neq i} \frac{\boldsymbol{x}_j^2}{\left(\sqrt{d - {}^1\!/\!\varepsilon}\right)^2} \right\}$$



Hard Instance – Analysis Outline

- At the end of Phase 1, the minimum volume outer ellipsoid is B_2^d.
- At the end of Phase 2, the final minimum volume outer ellipsoid is:

$$\left\{ x \ : \ 1 \geq \frac{x_i^2}{\left(1/\sqrt{d-\varepsilon}\right)^2} + \sum_{j \neq i} \frac{x_j^2}{\left(\sqrt{d - 1/\varepsilon}\right)^2} \right\}$$



Key observation – it is not possible to output an ellipsoid at the end of Phase 1 such that all outcomes of Phase 2 yield approximation factor $< \sqrt{d - \varepsilon}$.

Conclusion – Streaming Algorithms for Ellipsoidal Approximation of Convex Polytopes

1. We gave a **simple, nearly-optimal streaming algorithm** to calculate an ellipsoidal approximation for a centrally symmetric convex body.

Conclusion – Streaming Algorithms for Ellipsoidal Approximation of Convex Polytopes

1. We gave a **simple, nearly-optimal streaming algorithm** to calculate an ellipsoidal approximation for a centrally symmetric convex body.
2. Our algorithm's space complexity is independent of the length of the stream.

Conclusion – Streaming Algorithms for Ellipsoidal Approximation of Convex Polytopes

1. We gave a **simple, nearly-optimal streaming algorithm** to calculate an ellipsoidal approximation for a centrally symmetric convex body.
2. Our algorithm's space complexity is independent of the length of the stream.
3. Paper – `https://arxiv.org/abs/2206.07250`.

Table of contents

Dueling convex optimization with a monotone adversary

under conference review

Avrim Blum, Meghal Gupta, Gene Li, *NSM*, Aadirupa Saha, Chloe Yang

Preference-based feedback

Consider a *recommendation* problem – we give a user some items and ask them to choose their favorite. Over time, we want to learn their preferences.

Consider a *recommendation* problem – we give a user some items and ask them to choose their favorite. Over time, we want to learn their preferences.

$$\text{Round} \qquad \text{Recommender} \longleftrightarrow \left\{ \begin{array}{l} \text{Item } 1 \\ \text{Item } 2 \\ \ldots \\ \text{Item } m \end{array} \right\} \longleftrightarrow \text{User}$$

Preference-based feedback

Consider a *recommendation* problem – we give a user some items and ask them to choose their favorite. Over time, we want to learn their preferences.

$$\text{Round 1} \qquad\qquad \text{Recommender} \longleftrightarrow \left\{ \begin{array}{l} \text{Item 1} \\ \text{Item 2} \\ \ldots \\ \text{Item } m \end{array} \right\} \longleftrightarrow \text{User}$$

Preference-based feedback

Consider a *recommendation* problem – we give a user some items and ask them to choose their favorite. Over time, we want to learn their preferences.

$$\text{Round 2} \qquad \text{Recommender} \longleftrightarrow \left\{ \begin{array}{l} \text{Item 1} \\ \text{Item 2} \\ \dots \\ \text{Item } m \end{array} \right\} \longleftrightarrow \text{User}$$

Preference-based feedback

Consider a *recommendation* problem – we give a user some items and ask them to choose their favorite. Over time, we want to learn their preferences.

$$\text{Round 3} \qquad\qquad \text{Recommender} \longleftrightarrow \left\{ \begin{array}{l} \text{Item } 1 \\ \text{Item } 2 \\ \dots \\ \text{Item } m \end{array} \right\} \longleftrightarrow \text{User}$$

$\text{Item } m+1$

Modeling the problem

A natural way to formulate this problem is through *dueling convex optimization with a monotone adversary*.

Dueling optimization with a monotone adversary

Modeling the problem

A natural way to formulate this problem is through *dueling convex optimization with a monotone adversary*.

Dueling optimization with a monotone adversary

Let $\mathcal{X} \subseteq \mathbb{R}^d$ and $f \colon \mathcal{X} \to \mathbb{R}$ be a cost function with an unknown global minimum $\boldsymbol{x}^\star$.

Modeling the problem

A natural way to formulate this problem is through *dueling convex optimization with a monotone adversary*.

Dueling optimization with a monotone adversary

Let $\mathcal{X} \subseteq \mathbb{R}^d$ and $f\colon \mathcal{X} \to \mathbb{R}$ be a cost function with an unknown global minimum $\boldsymbol{x}^\star$. A learner interacts with an adversary over rounds $t = 1, 2, \ldots,$ where each round is of the following form.

Modeling the problem

A natural way to formulate this problem is through *dueling convex optimization with a monotone adversary*.

Dueling optimization with a monotone adversary

Let $\mathcal{X} \subseteq \mathbb{R}^d$ and $f\colon \mathcal{X} \to \mathbb{R}$ be a cost function with an unknown global minimum $\boldsymbol{x}^\star$. A learner interacts with an adversary over rounds $t = 1, 2, \ldots,$ where each round is of the following form.

1. The learner proposes two points $\boldsymbol{x}_t^{(1)}, \boldsymbol{x}_t^{(2)} \in \mathcal{X}$.

Modeling the problem

A natural way to formulate this problem is through *dueling convex optimization with a monotone adversary*.

Dueling optimization with a monotone adversary

Let $\mathcal{X} \subseteq \mathbb{R}^d$ and $f : \mathcal{X} \to \mathbb{R}$ be a cost function with an unknown global minimum $\boldsymbol{x}^\star$. A learner interacts with an adversary over rounds $t = 1, 2, \ldots,$ where each round is of the following form.

1. The learner proposes two points $\boldsymbol{x}_t^{(1)}, \boldsymbol{x}_t^{(2)} \in \mathcal{X}$.
2. The adversary responds with a point $\boldsymbol{r}_t$ that satisfies $f(\boldsymbol{r}_t) \leq \min\left\{ f\left(\boldsymbol{x}_t^{(1)}\right), f\left(\boldsymbol{x}_t^{(2)}\right) \right\}$.

Modeling the problem

A natural way to formulate this problem is through *dueling convex optimization with a monotone adversary*.

Dueling optimization with a monotone adversary

Let $\mathcal{X} \subseteq \mathbb{R}^d$ and $f : \mathcal{X} \to \mathbb{R}$ be a cost function with an unknown global minimum $\boldsymbol{x}^\star$. A learner interacts with an adversary over rounds $t = 1, 2, \ldots,$ where each round is of the following form.

1. The learner proposes two points $\boldsymbol{x}_t^{(1)}, \boldsymbol{x}_t^{(2)} \in \mathcal{X}$.
2. The adversary responds with a point $\boldsymbol{r}_t$ that satisfies
   $f(\boldsymbol{r}_t) \leq \min\left\{f\left(\boldsymbol{x}_t^{(1)}\right), f\left(\boldsymbol{x}_t^{(2)}\right)\right\}$.

The goal is to design algorithms that:

Modeling the problem

A natural way to formulate this problem is through *dueling convex optimization with a monotone adversary*.

Dueling optimization with a monotone adversary

Let $\mathcal{X} \subseteq \mathbb{R}^d$ and $f\colon \mathcal{X} \to \mathbb{R}$ be a cost function with an unknown global minimum $\boldsymbol{x}^\star$. A learner interacts with an adversary over rounds $t = 1, 2, \ldots,$ where each round is of the following form.

1. The learner proposes two points $\boldsymbol{x}_t^{(1)}, \boldsymbol{x}_t^{(2)} \in \mathcal{X}$.
2. The adversary responds with a point $\boldsymbol{r}_t$ that satisfies $f(\boldsymbol{r}_t) \leq \min\left\{f\left(\boldsymbol{x}_t^{(1)}\right), f\left(\boldsymbol{x}_t^{(2)}\right)\right\}$.

The goal is to design algorithms that:

1. for $\varepsilon > 0$, minimize the number of iterations to find a point $\boldsymbol{x}$ for which $f(\boldsymbol{x}) - f(\boldsymbol{x}^\star) \leq \varepsilon$;

Modeling the problem

A natural way to formulate this problem is through *dueling convex optimization with a monotone adversary*.

Dueling optimization with a monotone adversary

Let $\mathcal{X} \subseteq \mathbb{R}^d$ and $f\colon \mathcal{X} \to \mathbb{R}$ be a cost function with an unknown global minimum $\boldsymbol{x}^\star$. A learner interacts with an adversary over rounds $t = 1, 2, \ldots,$ where each round is of the following form.

1. The learner proposes two points $\boldsymbol{x}_t^{(1)}, \boldsymbol{x}_t^{(2)} \in \mathcal{X}$.
2. The adversary responds with a point $\boldsymbol{r}_t$ that satisfies $f(\boldsymbol{r}_t) \le \min\left\{ f\left(\boldsymbol{x}_t^{(1)}\right), f\left(\boldsymbol{x}_t^{(2)}\right)\right\}$.

The goal is to design algorithms that:

1. for $\varepsilon > 0$, minimize the number of iterations to find a point $\boldsymbol{x}$ for which $f(\boldsymbol{x}) - f(\boldsymbol{x}^\star) \le \varepsilon$;
2. minimize the cost $\left(\max\left\{ f\left(\boldsymbol{x}_t^{(1)}\right), f\left(\boldsymbol{x}_t^{(2)}\right)\right\} - f(\boldsymbol{x}^\star)\right)$.

Modeling the problem

A natural way to formulate this problem is through *dueling convex optimization with a monotone adversary*.

Dueling optimization with a monotone adversary

Let $\mathcal{X} \subseteq \mathbb{R}^d$ and $f \colon \mathcal{X} \to \mathbb{R}$ be a cost function with an unknown global minimum $\boldsymbol{x}^\star$. A learner interacts with an adversary over rounds $t = 1, 2, \ldots,$ where each round is of the following form.

1. The learner proposes two points $\boldsymbol{x}_t^{(1)}, \boldsymbol{x}_t^{(2)} \in \mathcal{X}$.
2. The adversary responds with a point $\boldsymbol{r}_t$ that satisfies $f(\boldsymbol{r}_t) \leq \min\left\{f\left(\boldsymbol{x}_t^{(1)}\right), f\left(\boldsymbol{x}_t^{(2)}\right)\right\}$.

The goal is to design algorithms that:

1. for $\varepsilon > 0$, minimize the number of iterations to find a point $\boldsymbol{x}$ for which $f(\boldsymbol{x}) - f(\boldsymbol{x}^\star) \leq \varepsilon$;
2. minimize the total cost $\sum_{t=1}^{\infty}\left(\max\left\{f\left(\boldsymbol{x}_t^{(1)}\right), f\left(\boldsymbol{x}_t^{(2)}\right)\right\} - f(\boldsymbol{x}^\star)\right)$.

Original problem (without the monotone adversary) was studied by Jamieson, Nowak, and Recht [JNR12] and Saha, Koren, and Mansour [SKM22].

Challenges with out-of-list feedback

The out-of-list feedback makes it challenging to analyze certain natural algorithms.

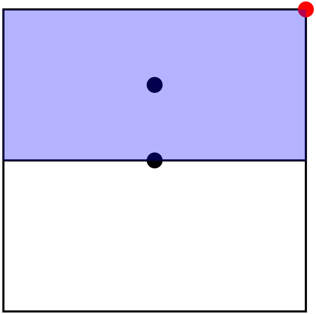
Challenges with out-of-list feedback

The out-of-list feedback makes it challenging to analyze certain natural algorithms.

Special case – $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2$.

Binary search

Choose query points so that we eliminate a constant fraction of the search space in each round.



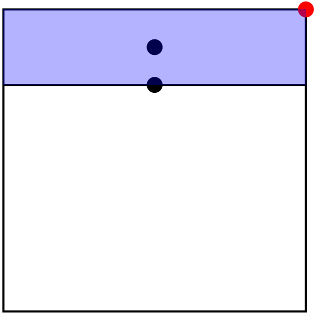
Challenges with out-of-list feedback

The out-of-list feedback makes it challenging to analyze certain natural algorithms.

Special case – $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2$.

Binary search

Choose query points so that we eliminate a constant fraction of the search space in each round.
Challenge – the volume of the set can be arbitrarily small, while the diameter can still be a constant.



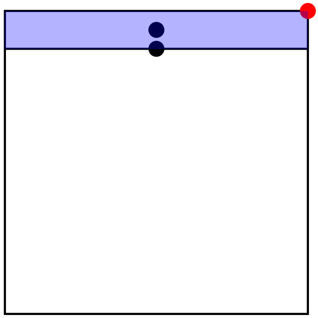
Challenges with out-of-list feedback

The out-of-list feedback makes it challenging to analyze certain natural algorithms.

Special case – $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2$.

Binary search

Choose query points so that we eliminate a constant fraction of the search space in each round.
Challenge – the volume of the set can be arbitrarily small, while the diameter can still be a constant.



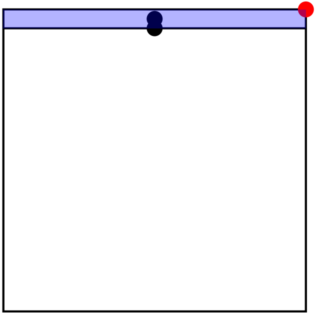
Challenges with out-of-list feedback

The out-of-list feedback makes it challenging to analyze certain natural algorithms.

Special case – $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2$.

Binary search

Choose query points so that we eliminate a constant fraction of the search space in each round.
Challenge – the volume of the set can be arbitrarily small, while the diameter can still be a constant.



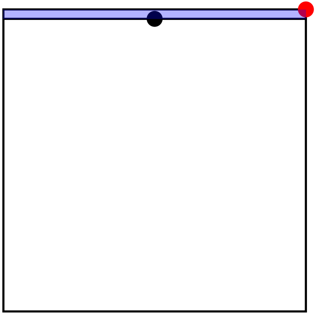
Challenges with out-of-list feedback

The out-of-list feedback makes it challenging to analyze certain natural algorithms.

Special case – $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2$.

Binary search

Choose query points so that we eliminate a constant fraction of the search space in each round.
Challenge – the volume of the set can be arbitrarily small, while the diameter can still be a constant.



Challenges with out-of-list feedback

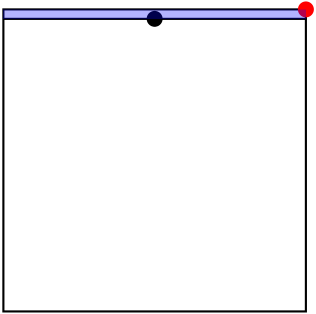
The out-of-list feedback makes it challenging to analyze certain natural algorithms.

Special case – $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2$.

Binary search

Choose query points so that we eliminate a constant fraction of the search space in each round.
Challenge – the volume of the set can be arbitrarily small, while the diameter can still be a constant.

Coordinate descent



Challenges with out-of-list feedback

The out-of-list feedback makes it challenging to analyze certain natural algorithms.

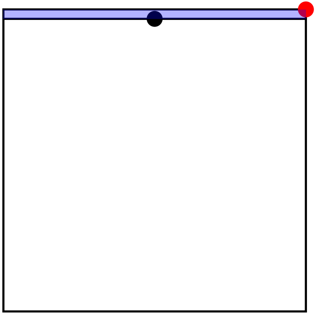
Special case – $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2$.

Binary search

Choose query points so that we eliminate a constant fraction of the search space in each round.
Challenge – the volume of the set can be arbitrarily small, while the diameter can still be a constant.

Coordinate descent

Fix a coordinate index i, hold all other coordinates constant, and check two variations of the ith coordinate. Repeat.



Challenges with out-of-list feedback

The out-of-list feedback makes it challenging to analyze certain natural algorithms.

Special case – $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2$.

Binary search

Choose query points so that we eliminate a constant fraction of the search space in each round.
Challenge – the volume of the set can be arbitrarily small, while the diameter can still be a constant.

Coordinate descent

Fix a coordinate index i, hold all other coordinates constant, and check two variations of the ith coordinate. Repeat.
Challenge – the adversary can return a point that does not reveal any information about the ith coordinate.

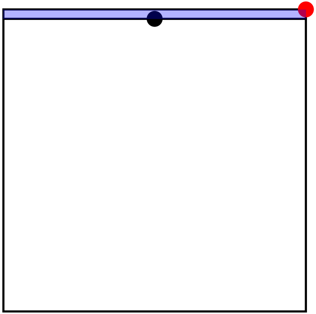


Table of contents

Our results

Recall that

$$\mathrm{cost}_t := \max_{i \in \{1,2\}} f\left(\boldsymbol{x}_t^{(i)}\right) - f(\boldsymbol{r}_t).$$

Main result

Our results

Recall that

$$\text{cost}_t := \max_{i \in \{1,2\}} f\left(\boldsymbol{x}_t^{(i)}\right) - f(\boldsymbol{r}_t).$$

Main result

There is a *simple, fast* algorithm that can handle the out-of-list feedback for dueling optimization for certain natural functions that, with high probability:

Our results

Recall that

$$\text{cost}_t := \max_{i \in \{1,2\}} f\left(\boldsymbol{x}_t^{(i)}\right) - f(\boldsymbol{r}_t).$$

Main result

There is a *simple, fast* algorithm that can handle the out-of-list feedback for dueling optimization for certain natural functions that, with high probability:

- finds an ε-optimal point in $O^\star\left(d \log\left(1/\varepsilon\right)^2\right)$ iterations;

Our results

Recall that

$$\mathrm{cost}_t := \max_{i \in \{1,2\}} f\left(\boldsymbol{x}_t^{(i)}\right) - f(\boldsymbol{r}_t).$$

Main result

There is a *simple, fast* algorithm that can handle the out-of-list feedback for dueling optimization for certain natural functions that, with high probability:

- finds an ε-optimal point in $O^\star\left(d \log\left({}^1/_\varepsilon\right)^2\right)$ iterations;
- accumulates total cost $O^\star(d)$.

Our results

Recall that

$$\text{cost}_t := \max_{i \in \{1,2\}} f\left(\boldsymbol{x}_t^{(i)}\right) - f(\boldsymbol{r}_t).$$

Main result

There is a *simple, fast* algorithm that can handle the out-of-list feedback for dueling optimization for certain natural functions that, with high probability:

- finds an ε-optimal point in $O^\star\left(d \log\left(1/\varepsilon\right)^2\right)$ iterations;
- accumulates total cost $O^\star(d)$.

Functions we can handle

Our results

Recall that

$$\mathrm{cost}_t := \max_{i \in \{1,2\}} f\left(\boldsymbol{x}_t^{(i)}\right) - f(\boldsymbol{r}_t).$$

Main result

There is a *simple, fast* algorithm that can handle the out-of-list feedback for dueling optimization for certain natural functions that, with high probability:

- finds an ε-optimal point in $O^\star\left(d \log\left({}^1\!/\!_\varepsilon\right)^2\right)$ iterations;
- accumulates total cost $O^\star(d)$.

Functions we can handle

- $f(\boldsymbol{x}) = -\langle \boldsymbol{x}, \boldsymbol{x}^\star \rangle$ when $\|\boldsymbol{x}\|_2, \|\boldsymbol{x}^\star\|_2 = 1$.

Our results

Recall that

$$\text{cost}_t := \max_{i \in \{1,2\}} f\left(\boldsymbol{x}_t^{(i)}\right) - f(\boldsymbol{r}_t).$$

Main result

There is a *simple, fast* algorithm that can handle the out-of-list feedback for dueling optimization for certain natural functions that, with high probability:

- finds an ε-optimal point in $O^\star\left(d \log\left(1/\varepsilon\right)^2\right)$ iterations;
- accumulates total cost $O^\star(d)$.

Functions we can handle

- $f(\boldsymbol{x}) = -\langle \boldsymbol{x}, \boldsymbol{x}^\star \rangle$ when $\|\boldsymbol{x}\|_2, \|\boldsymbol{x}^\star\|_2 = 1$.
- $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2$ when $\|\boldsymbol{x}\|_2 \leq 1$.

Our results

Recall that

$$\text{cost}_t := \max_{i \in \{1,2\}} f\left(\boldsymbol{x}_t^{(i)}\right) - f(\boldsymbol{r}_t).$$

Main result

There is a *simple, fast* algorithm that can handle the out-of-list feedback for dueling optimization for certain natural functions that, with high probability:

- finds an ε-optimal point in $O^\star\left(d \log\left(1/\varepsilon\right)^2\right)$ iterations;
- accumulates total cost $O^\star(d)$.

Functions we can handle

- $f(\boldsymbol{x}) = -\langle \boldsymbol{x}, \boldsymbol{x}^\star \rangle$ when $\|\boldsymbol{x}\|_2, \|\boldsymbol{x}^\star\|_2 = 1$.
- $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2$ when $\|\boldsymbol{x}\|_2 \leq 1$.
- (more generally) $f(\boldsymbol{x})$ is β-smooth and α-PŁ.

Our results

Recall that

$$\text{cost}_t := \max_{i \in \{1,2\}} f\left(\boldsymbol{x}_t^{(i)}\right) - f(\boldsymbol{r}_t).$$

Main result

There is a *simple, fast* algorithm that can handle the out-of-list feedback for dueling optimization for certain natural functions that, with high probability:

- finds an ε-optimal point in $O^\star\left(d \log\left({}^1\!/\!_\varepsilon\right)^2\right)$ iterations;
- accumulates total cost $O^\star(d)$.

Functions we can handle

- $f(\boldsymbol{x}) = -\langle \boldsymbol{x}, \boldsymbol{x}^\star \rangle$ when $\|\boldsymbol{x}\|_2, \|\boldsymbol{x}^\star\|_2 = 1$.
- $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2$ when $\|\boldsymbol{x}\|_2 \leq 1$. This talk.
- (more generally) $f(\boldsymbol{x})$ is β-smooth and α-PŁ.

Algorithm sketch

Algorithm sketch

1. Initialize $\boldsymbol{x}_0 = 0$ and $\varepsilon_0 \sim {}^1\!/\!\sqrt{d}$.

Algorithm sketch

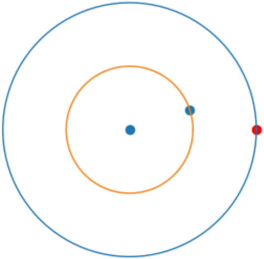
1. Initialize $\boldsymbol{x}_0 = 0$ and $\varepsilon_0 \sim {}^1/\sqrt{d}$.
2. for $t = 1, 2, \ldots$:

Algorithm sketch

1. Initialize $\boldsymbol{x}_0 = 0$ and $\varepsilon_0 \sim {}^1/\sqrt{d}$.
2. for $t = 1, 2, \ldots$:
   2.1 In iteration t, guess $\boldsymbol{x}_t$ and $\boldsymbol{x}_t + \varepsilon_t \cdot \mathrm{Unif}(\mathbb{S}^{d-1})$.

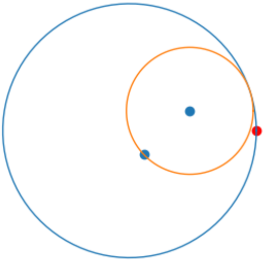
Algorithm sketch

1. Initialize $\boldsymbol{x}_0 = 0$ and $\varepsilon_0 \sim 1/\sqrt{d}$.
2. for $t = 1, 2, \ldots$:
   2.1 In iteration t, guess $\boldsymbol{x}_t$ and $\boldsymbol{x}_t + \varepsilon_t \cdot \mathrm{Unif}(\mathbb{S}^{d-1})$.
   2.2 Update $\boldsymbol{x}_{t+1} = \boldsymbol{r}_t$ ($\boldsymbol{r}_t$ is the adversary's response).



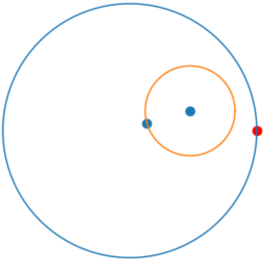
Algorithm sketch

1. Initialize $\boldsymbol{x}_0 = 0$ and $\varepsilon_0 \sim {}^1\!/\!\sqrt{d}$.
2. for $t = 1, 2, \ldots$:
   2.1 In iteration t, guess $\boldsymbol{x}_t$ and $\boldsymbol{x}_t + \varepsilon_t \cdot \mathrm{Unif}(\mathbb{S}^{d-1})$.
   2.2 Update $\boldsymbol{x}_{t+1} = \boldsymbol{r}_t$ ($\boldsymbol{r}_t$ is the adversary's response).



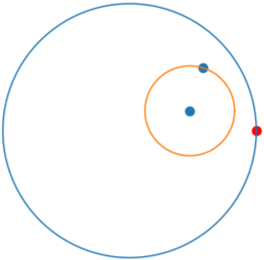
Algorithm sketch

1. Initialize $\boldsymbol{x}_0 = 0$ and $\varepsilon_0 \sim 1/\sqrt{d}$.
2. for $t = 1, 2, \ldots$:
   2.1 In iteration t, guess $\boldsymbol{x}_t$ and $\boldsymbol{x}_t + \varepsilon_t \cdot \mathrm{Unif}(\mathbb{S}^{d-1})$.
   2.2 Update $\boldsymbol{x}_{t+1} = \boldsymbol{r}_t$ ($\boldsymbol{r}_t$ is the adversary's response).
3. After sufficiently many steps, set $\varepsilon_{t+1} = \varepsilon_t/\sqrt{2}$.



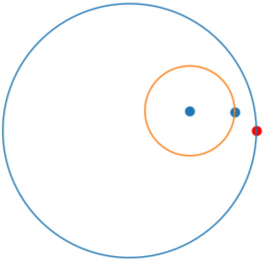
Algorithm sketch

1. Initialize $\boldsymbol{x}_0 = 0$ and $\varepsilon_0 \sim {}^1\!/\!\sqrt{d}$.
2. for $t = 1, 2, \ldots$:
   2.1 In iteration t, guess $\boldsymbol{x}_t$ and $\boldsymbol{x}_t + \varepsilon_t \cdot \mathrm{Unif}(\mathbb{S}^{d-1})$.
   2.2 Update $\boldsymbol{x}_{t+1} = \boldsymbol{r}_t$ ($\boldsymbol{r}_t$ is the adversary's response).
3. After sufficiently many steps, set $\varepsilon_{t+1} = \varepsilon_t / \sqrt{2}$.



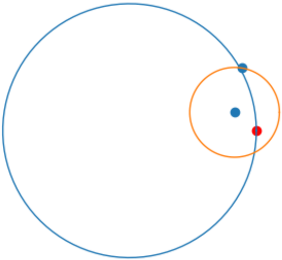
Algorithm sketch

1. Initialize $\mathbf{x}_0 = 0$ and $\varepsilon_0 \sim {}^1/\sqrt{d}$.
2. for $t = 1, 2, \ldots$:
   - 2.1 In iteration t, guess $\mathbf{x}_t$ and $\mathbf{x}_t + \varepsilon_t \cdot \text{Unif}(\mathbb{S}^{d-1})$.
   - 2.2 Update $\mathbf{x}_{t+1} = \mathbf{r}_t$ ($\mathbf{r}_t$ is the adversary's response).
3. After sufficiently many steps, set $\varepsilon_{t+1} = \varepsilon_t/\sqrt{2}$.



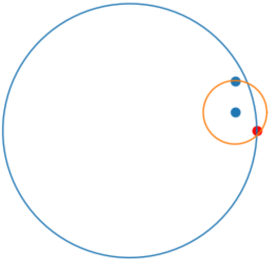
Algorithm sketch

1. Initialize $\boldsymbol{x}_0 = 0$ and $\varepsilon_0 \sim {}^1\!/\!\sqrt{d}$.
2. for $t = 1, 2, \ldots$:
    2.1 In iteration t, guess $\boldsymbol{x}_t$ and $\boldsymbol{x}_t + \varepsilon_t \cdot \mathrm{Unif}(\mathbb{S}^{d-1})$.
    2.2 Update $\boldsymbol{x}_{t+1} = \boldsymbol{r}_t$ ($\boldsymbol{r}_t$ is the adversary's response).
3. After sufficiently many steps, set $\varepsilon_{t+1} = \varepsilon_t / \sqrt{2}$.



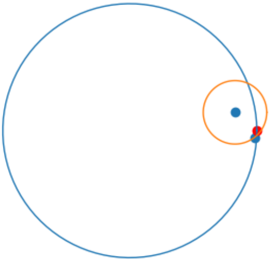
Algorithm sketch

1. Initialize $\boldsymbol{x}_0 = 0$ and $\varepsilon_0 \sim {}^{1}/\sqrt{d}$.
2. for $t = 1, 2, \ldots$:
   2.1 In iteration t, guess $\boldsymbol{x}_t$ and $\boldsymbol{x}_t + \varepsilon_t \cdot \mathrm{Unif}(\mathbb{S}^{d-1})$.
   2.2 Update $\boldsymbol{x}_{t+1} = \boldsymbol{r}_t$ ($\boldsymbol{r}_t$ is the adversary's response).
3. After sufficiently many steps, set $\varepsilon_{t+1} = \varepsilon_t / \sqrt{2}$.



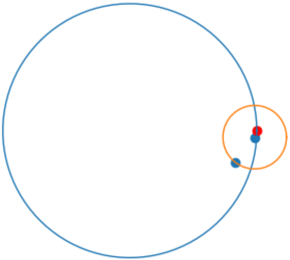
Algorithm sketch

1. Initialize $\boldsymbol{x}_0 = 0$ and $\varepsilon_0 \sim {}^1\!/\!\sqrt{d}$.
2. for $t = 1, 2, \ldots$:
   2.1 In iteration t, guess $\boldsymbol{x}_t$ and $\boldsymbol{x}_t + \varepsilon_t \cdot \mathrm{Unif}(\mathbb{S}^{d-1})$.
   2.2 Update $\boldsymbol{x}_{t+1} = \boldsymbol{r}_t$ ($\boldsymbol{r}_t$ is the adversary's response).
3. After sufficiently many steps, set $\varepsilon_{t+1} = \varepsilon_t / \sqrt{2}$.



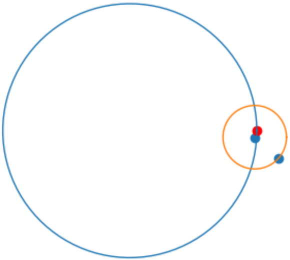
Algorithm sketch

1. Initialize $\boldsymbol{x}_0 = 0$ and $\varepsilon_0 \sim {}^1\!/\!\sqrt{d}$.
2. for $t = 1, 2, \ldots$:
   2.1 In iteration t, guess $\boldsymbol{x}_t$ and $\boldsymbol{x}_t + \varepsilon_t \cdot \mathrm{Unif}(\mathbb{S}^{d-1})$.
   2.2 Update $\boldsymbol{x}_{t+1} = \boldsymbol{r}_t$ ($\boldsymbol{r}_t$ is the adversary's response).
3. After sufficiently many steps, set $\varepsilon_{t+1} = \varepsilon_t / \sqrt{2}$.



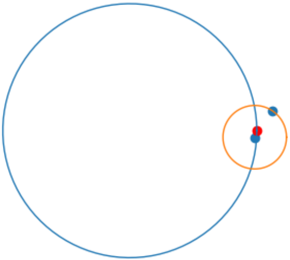
Algorithm sketch

1. Initialize $\boldsymbol{x}_0 = 0$ and $\varepsilon_0 \sim 1/\sqrt{d}$.
2. for $t = 1, 2, \ldots$:
   2.1 In iteration t, guess $\boldsymbol{x}_t$ and $\boldsymbol{x}_t + \varepsilon_t \cdot \mathrm{Unif}(\mathbb{S}^{d-1})$.
   2.2 Update $\boldsymbol{x}_{t+1} = \boldsymbol{r}_t$ ($\boldsymbol{r}_t$ is the adversary's response).
3. After sufficiently many steps, set $\varepsilon_{t+1} = \varepsilon_t/\sqrt{2}$.



Algorithm sketch

1. Initialize $\boldsymbol{x}_0 = 0$ and $\varepsilon_0 \sim {}^1\!/\!\sqrt{d}$.
2. for $t = 1, 2, \ldots$:
   2.1 In iteration t, guess $\boldsymbol{x}_t$ and $\boldsymbol{x}_t + \varepsilon_t \cdot \mathrm{Unif}(\mathbb{S}^{d-1})$.
   2.2 Update $\boldsymbol{x}_{t+1} = \boldsymbol{r}_t$ ($\boldsymbol{r}_t$ is the adversary's response).
3. After sufficiently many steps, set $\varepsilon_{t+1} = \varepsilon_t / \sqrt{2}$.



Analysis sketch

We need to prove a few properties.

Analysis sketch

We need to prove a few properties.

1. The algorithm makes progress (decreases its value of $f(\boldsymbol{x}^\star) - f(\boldsymbol{x}_t)$) in every iteration.

Analysis sketch

We need to prove a few properties.

1. The algorithm makes progress (decreases its value of $f(\boldsymbol{x}^\star) - f(\boldsymbol{x}_t)$) in every iteration.
   1.1 This should hold for a fixed step size ε_t for some number of iterations;

Analysis sketch

We need to prove a few properties.

1. The algorithm makes progress (decreases its value of $f(\boldsymbol{x}^\star) - f(\boldsymbol{x}_t)$) in every iteration.
   1.1 This should hold for a fixed step size ε_t for some number of iterations;
   1.2 Once we think we have made enough improvement, we need to decrease ε_t appropriately.

Analysis sketch

We need to prove a few properties.

1. The algorithm makes progress (decreases its value of $f(\boldsymbol{x}^\star) - f(\boldsymbol{x}_t)$) in every iteration.
   1.1 This should hold for a fixed step size ε_t for some number of iterations;
   1.2 Once we think we have made enough improvement, we need to decrease ε_t appropriately.
2. The probability that the algorithm fails to make enough progress is small, even totaled over infinitely many rounds.

Analysis sketch

We need to prove a few properties.

1. The algorithm makes progress (decreases its value of $f(\boldsymbol{x}^\star) - f(\boldsymbol{x}_t)$) in every iteration. Let's try to understand this.
    1.1 This should hold for a fixed step size ε_t for some number of iterations;
    1.2 Once we think we have made enough improvement, we need to decrease ε_t appropriately.
2. The probability that the algorithm fails to make enough progress is small, even totaled over infinitely many rounds.

How do we make progress?

How do we make progress?

Inner product with a random vector

Let $\boldsymbol{g} \sim \mathrm{Unif}(\mathbb{S}_2^{d-1})$ and let $\boldsymbol{y} \in \mathbb{S}_2^{d-1}$ be fixed. Then $\Pr_{\boldsymbol{g}}\left[\langle \boldsymbol{g}, \boldsymbol{y} \rangle \geq \frac{1}{2\sqrt{d}}\right] \geq \frac{1}{8}$.

How do we make progress?

Inner product with a random vector

Let $\boldsymbol{g} \sim \mathrm{Unif}(\mathbb{S}_2^{d-1})$ and let $\boldsymbol{y} \in \mathbb{S}_2^{d-1}$ be fixed. Then $\Pr_{\boldsymbol{g}}\left[\langle \boldsymbol{g}, \boldsymbol{y} \rangle \geq \frac{1}{2\sqrt{d}}\right] \geq \frac{1}{8}$.

Let $\boldsymbol{y} = \nabla f(\boldsymbol{x}_t)$. Then, with constant probability, we choose a step that is vaguely aligned with $-\frac{\nabla f(\boldsymbol{x}_t)}{\|\nabla f(\boldsymbol{x}_t)\|_2} = \frac{\boldsymbol{x}^\star - \boldsymbol{x}_t}{\|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2}$.

How do we make progress?

Inner product with a random vector

Let $\boldsymbol{g} \sim \mathrm{Unif}(\mathbb{S}_2^{d-1})$ and let $\boldsymbol{y} \in \mathbb{S}_2^{d-1}$ be fixed. Then $\Pr_{\boldsymbol{g}}\left[\langle \boldsymbol{g}, \boldsymbol{y} \rangle \geq \frac{1}{2\sqrt{d}}\right] \geq \frac{1}{8}$.

Let $\boldsymbol{y} = \nabla f(\boldsymbol{x}_t)$. Then, with constant probability, we choose a step that is vaguely aligned with $-\frac{\nabla f(\boldsymbol{x}_t)}{\|\nabla f(\boldsymbol{x}_t)\|_2} = \frac{\boldsymbol{x}^\star - \boldsymbol{x}_t}{\|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2}$.

1. Using a step size of $\sim \frac{1}{\sqrt{d}}$, with constant probability, we get

$$\|\boldsymbol{x}^\star - \boldsymbol{x}_{t+1}\|_2^2 \leq \left(1 - \frac{C}{d}\right) \|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2^2 .$$

How do we make progress?

Inner product with a random vector

Let $\boldsymbol{g} \sim \mathrm{Unif}(\mathbb{S}_2^{d-1})$ and let $\boldsymbol{y} \in \mathbb{S}_2^{d-1}$ be fixed. Then $\Pr_{\boldsymbol{g}}\left[\langle \boldsymbol{g}, \boldsymbol{y} \rangle \geq \frac{1}{2\sqrt{d}}\right] \geq \frac{1}{8}$.

Let $\boldsymbol{y} = \nabla f(\boldsymbol{x}_t)$. Then, with constant probability, we choose a step that is vaguely aligned with $-\frac{\nabla f(\boldsymbol{x}_t)}{\|\nabla f(\boldsymbol{x}_t)\|_2} = \frac{\boldsymbol{x}^\star - \boldsymbol{x}_t}{\|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2}$.

1. Using a step size of $\sim \frac{1}{\sqrt{d}}$, with constant probability, we get
$$\|\boldsymbol{x}^\star - \boldsymbol{x}_{t+1}\|_2^2 \leq \left(1 - \frac{C}{d}\right) \|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2^2.$$
2. After $\sim d$ such "successful steps," we have decreased our cost by a constant factor.

How do we make progress?

Inner product with a random vector

Let $\boldsymbol{g} \sim \mathrm{Unif}(\mathbb{S}_2^{d-1})$ and let $\boldsymbol{y} \in \mathbb{S}_2^{d-1}$ be fixed. Then $\Pr_{\boldsymbol{g}}\left[\langle \boldsymbol{g}, \boldsymbol{y}\rangle \geq \frac{1}{2\sqrt{d}}\right] \geq \frac{1}{8}$.

Let $\boldsymbol{y} = \nabla f(\boldsymbol{x}_t)$. Then, with constant probability, we choose a step that is vaguely aligned with $-\frac{\nabla f(\boldsymbol{x}_t)}{\|\nabla f(\boldsymbol{x}_t)\|_2} = \frac{\boldsymbol{x}^\star - \boldsymbol{x}_t}{\|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2}$.

1. Using a step size of $\sim \frac{1}{\sqrt{d}}$, with constant probability, we get
$$\|\boldsymbol{x}^\star - \boldsymbol{x}_{t+1}\|_2^2 \leq \left(1 - \frac{C}{d}\right) \|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2^2 .$$
2. After $\sim d$ such "successful steps," we have decreased our cost by a constant factor.
3. Once we are confident we have run enough "successful steps", decay ε_t and continue ad infinitum.

How do we make progress?

Inner product with a random vector

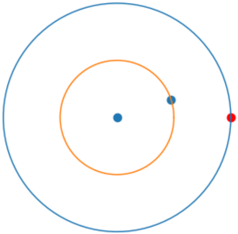
Let $\boldsymbol{g} \sim \mathrm{Unif}(\mathbb{S}_2^{d-1})$ and let $\boldsymbol{y} \in \mathbb{S}_2^{d-1}$ be fixed. Then $\Pr_{\boldsymbol{g}}\left[\langle \boldsymbol{g}, \boldsymbol{y} \rangle \geq \frac{1}{2\sqrt{d}}\right] \geq \frac{1}{8}$.

Let $\boldsymbol{y} = \nabla f(\boldsymbol{x}_t)$. Then, with constant probability, we choose a step that is vaguely aligned with $-\frac{\nabla f(\boldsymbol{x}_t)}{\|\nabla f(\boldsymbol{x}_t)\|_2} = \frac{\boldsymbol{x}^\star - \boldsymbol{x}_t}{\|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2}$.

1. Using a step size of $\sim \frac{1}{\sqrt{d}}$, with constant probability, we get

$$\|\boldsymbol{x}^\star - \boldsymbol{x}_{t+1}\|_2^2 \leq \left(1 - \frac{C}{d}\right) \|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2^2 .$$

2. After $\sim d$ such "successful steps," we have decreased our cost by a constant factor.
3. Once we are confident we have run enough "successful steps", decay ε_t and continue ad infinitum.



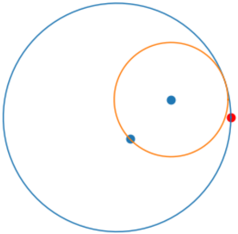
How do we make progress?

Inner product with a random vector

Let $\boldsymbol{g} \sim \mathrm{Unif}(\mathbb{S}_2^{d-1})$ and let $\boldsymbol{y} \in \mathbb{S}_2^{d-1}$ be fixed. Then $\Pr_{\boldsymbol{g}}\left[\langle \boldsymbol{g}, \boldsymbol{y} \rangle \geq \frac{1}{2\sqrt{d}}\right] \geq \frac{1}{8}$.

Let $\boldsymbol{y} = \nabla f(\boldsymbol{x}_t)$. Then, with constant probability, we choose a step that is vaguely aligned with $-\frac{\nabla f(\boldsymbol{x}_t)}{\|\nabla f(\boldsymbol{x}_t)\|_2} = \frac{\boldsymbol{x}^\star - \boldsymbol{x}_t}{\|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2}$.

1. Using a step size of $\sim \frac{1}{\sqrt{d}}$, with constant probability, we get
$$\|\boldsymbol{x}^\star - \boldsymbol{x}_{t+1}\|_2^2 \leq \left(1 - \frac{C}{d}\right) \|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2^2 .$$
2. After $\sim d$ such "successful steps," we have decreased our cost by a constant factor.
3. Once we are confident we have run enough "successful steps", decay ε_t and continue ad infinitum.



How do we make progress?

Inner product with a random vector

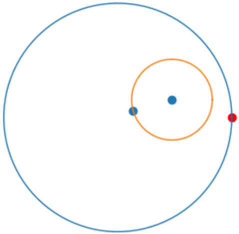
Let $\boldsymbol{g} \sim \mathrm{Unif}(\mathbb{S}_2^{d-1})$ and let $\boldsymbol{y} \in \mathbb{S}_2^{d-1}$ be fixed. Then $\Pr_{\boldsymbol{g}}\left[\langle \boldsymbol{g}, \boldsymbol{y} \rangle \geq \frac{1}{2\sqrt{d}}\right] \geq \frac{1}{8}$.

Let $\boldsymbol{y} = \nabla f(\boldsymbol{x}_t)$. Then, with constant probability, we choose a step that is vaguely aligned with $-\frac{\nabla f(\boldsymbol{x}_t)}{\|\nabla f(\boldsymbol{x}_t)\|_2} = \frac{\boldsymbol{x}^\star - \boldsymbol{x}_t}{\|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2}$.

1. Using a step size of $\sim \frac{1}{\sqrt{d}}$, with constant probability, we get

$$\|\boldsymbol{x}^\star - \boldsymbol{x}_{t+1}\|_2^2 \leq \left(1 - \frac{C}{d}\right) \|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2^2 .$$

2. After $\sim d$ such "successful steps," we have decreased our cost by a constant factor.
3. Once we are confident we have run enough "successful steps", decay ε_t and continue ad infinitum.



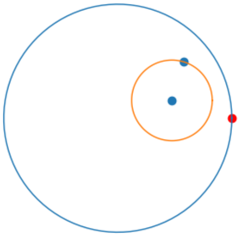
How do we make progress?

Inner product with a random vector

Let $\boldsymbol{g} \sim \mathrm{Unif}(\mathbb{S}_2^{d-1})$ and let $\boldsymbol{y} \in \mathbb{S}_2^{d-1}$ be fixed. Then $\Pr_{\boldsymbol{g}}\left[\langle \boldsymbol{g}, \boldsymbol{y} \rangle \geq \frac{1}{2\sqrt{d}}\right] \geq \frac{1}{8}$.

Let $\boldsymbol{y} = \nabla f(\boldsymbol{x}_t)$. Then, with constant probability, we choose a step that is vaguely aligned with $-\frac{\nabla f(\boldsymbol{x}_t)}{\|\nabla f(\boldsymbol{x}_t)\|_2} = \frac{\boldsymbol{x}^\star - \boldsymbol{x}_t}{\|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2}$.

1. Using a step size of $\sim \frac{1}{\sqrt{d}}$, with constant probability, we get
$$\|\boldsymbol{x}^\star - \boldsymbol{x}_{t+1}\|_2^2 \leq \left(1 - \frac{C}{d}\right) \|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2^2 .$$
2. After $\sim d$ such "successful steps," we have decreased our cost by a constant factor.
3. Once we are confident we have run enough "successful steps", decay ε_t and continue ad infinitum.



How do we make progress?

Inner product with a random vector

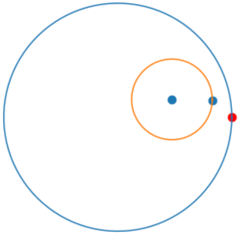
Let $\boldsymbol{g} \sim \mathrm{Unif}(\mathbb{S}_2^{d-1})$ and let $\boldsymbol{y} \in \mathbb{S}_2^{d-1}$ be fixed. Then $\Pr_{\boldsymbol{g}}\left[\langle \boldsymbol{g}, \boldsymbol{y}\rangle \geq \frac{1}{2\sqrt{d}}\right] \geq \frac{1}{8}$.

Let $\boldsymbol{y} = \nabla f(\boldsymbol{x}_t)$. Then, with constant probability, we choose a step that is vaguely aligned with $-\frac{\nabla f(\boldsymbol{x}_t)}{\|\nabla f(\boldsymbol{x}_t)\|_2} = \frac{\boldsymbol{x}^\star - \boldsymbol{x}_t}{\|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2}$.

1. Using a step size of $\sim \frac{1}{\sqrt{d}}$, with constant probability, we get

$$\|\boldsymbol{x}^\star - \boldsymbol{x}_{t+1}\|_2^2 \leq \left(1 - \frac{C}{d}\right)\|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2^2 .$$

2. After $\sim d$ such "successful steps," we have decreased our cost by a constant factor.
3. Once we are confident we have run enough "successful steps", decay ε_t and continue ad infinitum.



How do we make progress?

Inner product with a random vector

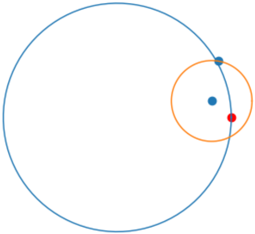
Let $\boldsymbol{g} \sim \text{Unif}(\mathbb{S}_2^{d-1})$ and let $\boldsymbol{y} \in \mathbb{S}_2^{d-1}$ be fixed. Then $\Pr_{\boldsymbol{g}}\left[\langle \boldsymbol{g}, \boldsymbol{y} \rangle \geq \frac{1}{2\sqrt{d}}\right] \geq \frac{1}{8}$.

Let $\boldsymbol{y} = \nabla f(\boldsymbol{x}_t)$. Then, with constant probability, we choose a step that is vaguely aligned with $-\frac{\nabla f(\boldsymbol{x}_t)}{\|\nabla f(\boldsymbol{x}_t)\|_2} = \frac{\boldsymbol{x}^\star - \boldsymbol{x}_t}{\|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2}$.

1. Using a step size of $\sim \frac{1}{\sqrt{d}}$, with constant probability, we get

$$\|\boldsymbol{x}^\star - \boldsymbol{x}_{t+1}\|_2^2 \leq \left(1 - \frac{C}{d}\right) \|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2^2 .$$

2. After $\sim d$ such "successful steps," we have decreased our cost by a constant factor.
3. Once we are confident we have run enough "successful steps", decay ε_t and continue ad infinitum.



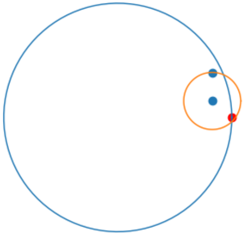
How do we make progress?

Inner product with a random vector

Let $\boldsymbol{g} \sim \mathrm{Unif}(\mathbb{S}_2^{d-1})$ and let $\boldsymbol{y} \in \mathbb{S}_2^{d-1}$ be fixed. Then $\Pr_{\boldsymbol{g}}\left[\langle \boldsymbol{g}, \boldsymbol{y}\rangle \geq \frac{1}{2\sqrt{d}}\right] \geq \frac{1}{8}$.

Let $\boldsymbol{y} = \nabla f(\boldsymbol{x}_t)$. Then, with constant probability, we choose a step that is vaguely aligned with $-\frac{\nabla f(\boldsymbol{x}_t)}{\|\nabla f(\boldsymbol{x}_t)\|_2} = \frac{\boldsymbol{x}^\star - \boldsymbol{x}_t}{\|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2}$.

1. Using a step size of $\sim \frac{1}{\sqrt{d}}$, with constant probability, we get
$$\|\boldsymbol{x}^\star - \boldsymbol{x}_{t+1}\|_2^2 \leq \left(1 - \frac{C}{d}\right) \|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2^2 .$$
2. After $\sim d$ such "successful steps," we have decreased our cost by a constant factor.
3. Once we are confident we have run enough "successful steps", decay ε_t and continue ad infinitum.



How do we make progress?

Inner product with a random vector

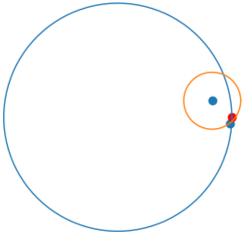
Let $\boldsymbol{g} \sim \mathrm{Unif}(\mathbb{S}_2^{d-1})$ and let $\boldsymbol{y} \in \mathbb{S}_2^{d-1}$ be fixed. Then $\Pr_{\boldsymbol{g}}\left[\langle \boldsymbol{g}, \boldsymbol{y}\rangle \geq \frac{1}{2\sqrt{d}}\right] \geq \frac{1}{8}$.

Let $\boldsymbol{y} = \nabla f(\boldsymbol{x}_t)$. Then, with constant probability, we choose a step that is vaguely aligned with $-\frac{\nabla f(\boldsymbol{x}_t)}{\|\nabla f(\boldsymbol{x}_t)\|_2} = \frac{\boldsymbol{x}^\star - \boldsymbol{x}_t}{\|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2}$.

1. Using a step size of $\sim \frac{1}{\sqrt{d}}$, with constant probability, we get

$$\|\boldsymbol{x}^\star - \boldsymbol{x}_{t+1}\|_2^2 \leq \left(1 - \frac{C}{d}\right)\|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2^2 .$$

2. After $\sim d$ such "successful steps," we have decreased our cost by a constant factor.
3. Once we are confident we have run enough "successful steps", decay ε_t and continue ad infinitum.



How do we make progress?

Inner product with a random vector

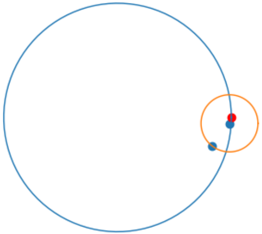
Let $\boldsymbol{g} \sim \mathrm{Unif}(\mathbb{S}_2^{d-1})$ and let $\boldsymbol{y} \in \mathbb{S}_2^{d-1}$ be fixed. Then $\Pr_{\boldsymbol{g}}\left[\langle \boldsymbol{g}, \boldsymbol{y}\rangle \geq \frac{1}{2\sqrt{d}}\right] \geq \frac{1}{8}$.

Let $\boldsymbol{y} = \nabla f(\boldsymbol{x}_t)$. Then, with constant probability, we choose a step that is vaguely aligned with $-\frac{\nabla f(\boldsymbol{x}_t)}{\|\nabla f(\boldsymbol{x}_t)\|_2} = \frac{\boldsymbol{x}^\star - \boldsymbol{x}_t}{\|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2}$.

1. Using a step size of $\sim \frac{1}{\sqrt{d}}$, with constant probability, we get

$$\|\boldsymbol{x}^\star - \boldsymbol{x}_{t+1}\|_2^2 \leq \left(1 - \frac{C}{d}\right)\|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2^2 .$$

2. After $\sim d$ such "successful steps," we have decreased our cost by a constant factor.
3. Once we are confident we have run enough "successful steps", decay ε_t and continue ad infinitum.



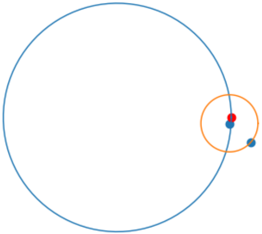
How do we make progress?

Inner product with a random vector

Let $\boldsymbol{g} \sim \mathrm{Unif}(\mathbb{S}_2^{d-1})$ and let $\boldsymbol{y} \in \mathbb{S}_2^{d-1}$ be fixed. Then $\Pr_{\boldsymbol{g}}\left[\langle \boldsymbol{g}, \boldsymbol{y} \rangle \geq \frac{1}{2\sqrt{d}}\right] \geq \frac{1}{8}$.

Let $\boldsymbol{y} = \nabla f(\boldsymbol{x}_t)$. Then, with constant probability, we choose a step that is vaguely aligned with $-\frac{\nabla f(\boldsymbol{x}_t)}{\|\nabla f(\boldsymbol{x}_t)\|_2} = \frac{\boldsymbol{x}^\star - \boldsymbol{x}_t}{\|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2}$.

1. Using a step size of $\sim \frac{1}{\sqrt{d}}$, with constant probability, we get
$$\|\boldsymbol{x}^\star - \boldsymbol{x}_{t+1}\|_2^2 \leq \left(1 - \frac{C}{d}\right) \|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2^2 .$$
2. After $\sim d$ such "successful steps," we have decreased our cost by a constant factor.
3. Once we are confident we have run enough "successful steps", decay ε_t and continue ad infinitum.



How do we make progress?

Inner product with a random vector

Let $\boldsymbol{g} \sim \mathrm{Unif}(\mathbb{S}_2^{d-1})$ and let $\boldsymbol{y} \in \mathbb{S}_2^{d-1}$ be fixed. Then $\Pr_{\boldsymbol{g}}\left[\langle \boldsymbol{g}, \boldsymbol{y} \rangle \geq \frac{1}{2\sqrt{d}}\right] \geq \frac{1}{8}$.

Let $\boldsymbol{y} = \nabla f(\boldsymbol{x}_t)$. Then, with constant probability, we choose a step that is vaguely aligned with $-\frac{\nabla f(\boldsymbol{x}_t)}{\|\nabla f(\boldsymbol{x}_t)\|_2} = \frac{\boldsymbol{x}^\star - \boldsymbol{x}_t}{\|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2}$.

1. Using a step size of $\sim \frac{1}{\sqrt{d}}$, with constant probability, we get

$$\|\boldsymbol{x}^\star - \boldsymbol{x}_{t+1}\|_2^2 \leq \left(1 - \frac{C}{d}\right) \|\boldsymbol{x}^\star - \boldsymbol{x}_t\|_2^2 .$$

2. After $\sim d$ such "successful steps," we have decreased our cost by a constant factor.
3. Once we are confident we have run enough "successful steps", decay ε_t and continue ad infinitum.

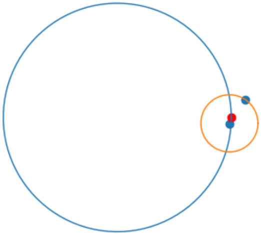


Table of contents

Does guessing more points in each round help?

A more realistic scenario is when the recommender is allowed to suggest up to m items. Can this help us decrease the cost?

Does guessing more points in each round help?

A more realistic scenario is when the recommender is allowed to suggest up to m items. Can this help us decrease the cost?

Lower bound

If $f(x) = \|x - x^\star\|_2^2$, then any randomized algorithm for m-ary convex optimization that is allowed to suggest up to m points in each round must take at least $\sim \frac{d}{\log m}$ rounds to identify a 0.1^2-optimal solution.

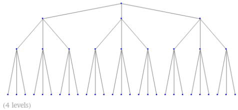
Does guessing more points in each round help?

A more realistic scenario is when the recommender is allowed to suggest up to m items. Can this help us decrease the cost?

Lower bound

If $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2$, then any randomized algorithm for m-ary convex optimization that is allowed to suggest up to m points in each round must take at least $\sim \frac{d}{\log m}$ rounds to identify a 0.1^2-optimal solution.

1. There are m possible responses that the adversary can return, so after R rounds, the algorithm can be in at most m^R possible states.



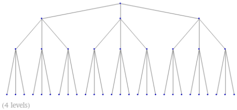
Does guessing more points in each round help?

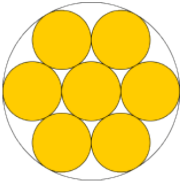
A more realistic scenario is when the recommender is allowed to suggest up to m items. Can this help us decrease the cost?

Lower bound

If $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2$, then any randomized algorithm for m-ary convex optimization that is allowed to suggest up to m points in each round must take at least $\sim \frac{d}{\log m}$ rounds to identify a 0.1^2-optimal solution.

1. There are m possible responses that the adversary can return, so after R rounds, the algorithm can be in at most m^R possible states.
2. There exists a subset $S \subset B_2^d$ of size $2^{\Omega(d)}$ such that every point in S is at least 0.2-far from every other point in S. Suppose $\boldsymbol{x}^\star \in S$.





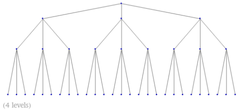
Does guessing more points in each round help?

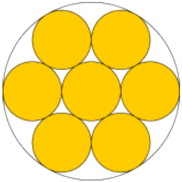
A more realistic scenario is when the recommender is allowed to suggest up to m items. Can this help us decrease the cost?

Lower bound

If $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2$, then any randomized algorithm for m-ary convex optimization that is allowed to suggest up to m points in each round must take at least $\sim \frac{d}{\log m}$ rounds to identify a 0.1^2-optimal solution.

1. There are m possible responses that the adversary can return, so after R rounds, the algorithm can be in at most m^R possible states.
2. There exists a subset $S \subset B_2^d$ of size $2^{\Omega(d)}$ such that every point in S is at least 0.2-far from every other point in S. Suppose $\boldsymbol{x}^\star \in S$.
3. Finding a 0.1^2-optimal point is at least as hard as identifying the member of S that $\boldsymbol{x}^\star$ belonged to.





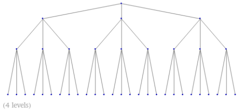
Does guessing more points in each round help?

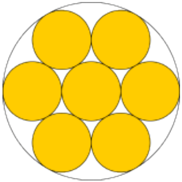
A more realistic scenario is when the recommender is allowed to suggest up to m items. Can this help us decrease the cost?

Lower bound

If $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2$, then any randomized algorithm for m-ary convex optimization that is allowed to suggest up to m points in each round must take at least $\sim \frac{d}{\log m}$ rounds to identify a 0.1^2-optimal solution.

1. There are m possible responses that the adversary can return, so after R rounds, the algorithm can be in at most m^R possible states.
2. There exists a subset $S \subset B_2^d$ of size $2^{\Omega(d)}$ such that every point in S is at least 0.2-far from every other point in S. Suppose $\boldsymbol{x}^\star \in S$.
3. Finding a 0.1^2-optimal point is at least as hard as identifying the member of S that $\boldsymbol{x}^\star$ belonged to.
4. One of the m^R states must have decided the closest member of S. Hence, $m^R \geq 2^{\Omega(d)}$.





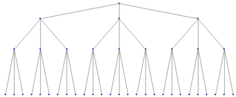
Does guessing more points in each round help?

A more realistic scenario is when the recommender is allowed to suggest up to m items. Can this help us decrease the cost?

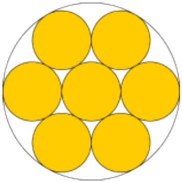
Lower bound

If $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2$, then any randomized algorithm for m-ary convex optimization that is allowed to suggest up to m points in each round must take at least $\sim \frac{d}{\log m}$ rounds to identify a 0.1^2-optimal solution.

1. There are m possible responses that the adversary can return, so after R rounds, the algorithm can be in at most m^R possible states.
2. There exists a subset $S \subset B_2^d$ of size $2^{\Omega(d)}$ such that every point in S is at least 0.2-far from every other point in S. Suppose $\boldsymbol{x}^\star \in S$.
3. Finding a 0.1^2-optimal point is at least as hard as identifying the member of S that $\boldsymbol{x}^\star$ belonged to.
4. One of the m^R states must have decided the closest member of S. Hence, $m^R \geq 2^{\Omega(d)}$.



(4 levels)



Upshot – our algorithm is **optimal**.

Conclusion – Dueling convex optimization with a monotone adversary

Conclusion – Dueling convex optimization with a monotone adversary

- We gave the first algorithm for dueling convex optimization with a monotone adversary when the underlying function is linear or smooth/PŁ.

Conclusion – Dueling convex optimization with a monotone adversary

- We gave the first algorithm for dueling convex optimization with a monotone adversary when the underlying function is linear or smooth/PŁ.
- Our algorithm's total cost and iteration complexities are optimal in the dimension d.

Conclusion – Dueling convex optimization with a monotone adversary

- We gave the first algorithm for dueling convex optimization with a monotone adversary when the underlying function is linear or smooth/PŁ.
- Our algorithm's total cost and iteration complexities are optimal in the dimension d.
- arXiv forthcoming! (or email me for the manuscript)

Modern input concerns for algorithmic data science

My goal – design and analyze algorithms for more realistic input scenarios.

Unwieldy input

- Approximating convex polytopes in a stream (Makarychev, Manoj, and Ovsiankin [MMO22; MMO23], COLT 2022/under submission).
- Approximating large convex objective functions (Manoj and Ovsiankin [MO23], under submission).
- Explaining classifier predictions on large inputs (Gupta and Manoj [GM23], SOSA 2023).

Unexpected input

- Robust machine learning under backdoor poisoning attacks (Manoj and Blum [MB21], NeurIPS 2021).
- Learning from out-of-list feedback (Blum, Gupta, Li, Manoj, Saha, and Yang [BGLMSY23], under submission).
- Generalization of short-program interpolators (Manoj and Srebro [MS23], COLT 2023).

- Research statement – https://narenmanoj.github.io/nsm_statement.pdf
- Publication list – https://narenmanoj.github.io/nsm_publist.pdf

Questions?

Thank you!!

Avrim Blum, Meghal Gupta, Gene Li, Naren Sarayu Manoj, Aadirupa Saha, and Yuanyuan Yang. Dueling optimization with a monotone adversary. *under review*, September 2023 (cited on pages 9–11, 190).

Michael B. Cohen, Ben Cousins, Yin Tat Lee, and Xin Yang. A near-optimal algorithm for approximating the John ellipsoid. In *Proceedings of the Conference on Learning Theory*, volume 99, pages 849–873, 2019 (cited on pages 23–28).

Meghal Gupta and Naren Sarayu Manoj. *An optimal algorithm for certifying monotone functions*. In *Symposium on Simplicity in Algorithms (SOSA)*. January 2023, pages 207–212 (cited on pages 2–11, 190).

Kevin G Jamieson, Robert Nowak, and Ben Recht. Query complexity of derivative-free optimization. *Advances in Neural Information Processing Systems*, 25, 2012 (cited on pages 112–120).

References II

Fritz John. Extremum problems with inequalities as subsidiary conditions. In *Studies and Essays Presented to R. Courant on his 60th Birthday*, pages 187–204. Interscience Publishers, Inc, 1948 (cited on pages 23–28).

Yingkai Li, Edmund Y. Lou, and Liren Shan. Stochastic linear optimization with adversarial corruption. 2019. arXiv: 1909.02109 [cs.LG] (cited on pages 14–22).

Yury Makarychev, Naren Sarayu Manoj, and Max Ovsiankin. Streaming algorithms for ellipsoidal approximation of convex polytopes. In *Proceedings of Thirty Fifth Conference on Learning Theory (COLT)*, pages 3070–3093, July 2022 (cited on pages 4–11, 190).

Yury Makarychev, Naren Sarayu Manoj, and Max Ovsiankin. Near-optimal streaming ellipsoidal rounding for general convex polytopes. *working manuscript*, October 2023 (cited on pages 4–11, 190).

Naren Sarayu Manoj and Avrim Blum. Excess capacity and backdoor poisoning. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021 (cited on pages 2–11, 190).

References III

Naren Sarayu Manoj and Max Ovsiankin. The change-of-measure method, block lewis weights, and approximating matrix block norms. *working manuscript*, October 2023 (cited on pages 2–11, 190).

Naren Sarayu Manoj and Nathan Srebro. Shortest program interpolation learning. In *Proceedings of Thirty Sixth Conference on Learning Theory (COLT)*, pages 4881–4901, July 2023 (cited on pages 2–11, 190).

Elon Rimon and Stephen P. Boyd. Obstacle collision detection using best ellipsoid fit. *J. Intell. Robotics Syst.*, 18(2):105–126, 1997 (cited on pages 14–22).

Aadirupa Saha, Tomer Koren, and Yishay Mansour. Dueling convex optimization with general preferences. *arXiv preprint arXiv:2210.02562*, 2022 (cited on pages 112–120).