Advanced Algorithms (Fall 2025)

Semi-Definite Programming

Lecturers: 尹一通，刘景铖，栗师

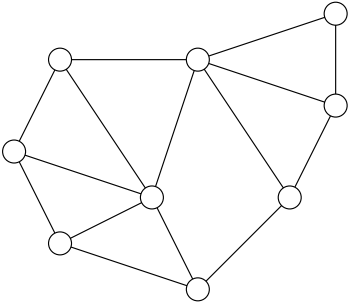
Nanjing University

Outline

Maximum Cut Problem

Input: $G = (V, E)$,

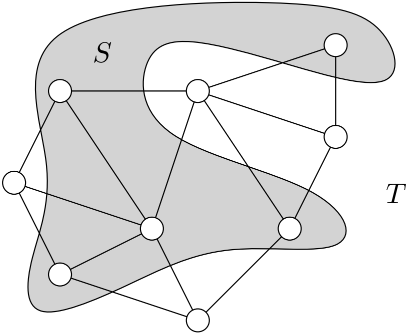
Output: a partition $(S \subseteq V, T := V \setminus S)$ of V so as to maximize $|E(S,T)|$,
where $E(S,T) = \{uv \in E : |\{u,v\} \cap S| = 1\}$



Maximum Cut Problem

Input: $G = (V, E)$,

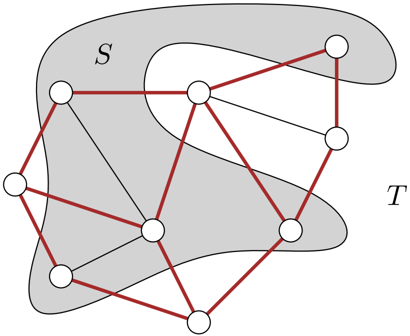
Output: a partition $(S \subseteq V, T := V \setminus S)$ of V so as to maximize $|E(S,T)|$,
where $E(S,T) = \{uv \in E : |\{u,v\} \cap S| = 1\}$



Maximum Cut Problem

Input: $G = (V, E)$,

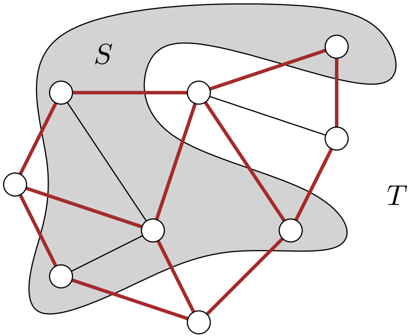
Output: a partition $(S \subseteq V, T := V \setminus S)$ of V so as to maximize $|E(S,T)|$,
where $E(S,T) = \{uv \in E : |\{u,v\} \cap S| = 1\}$



Maximum Cut Problem

Input: $G = (V, E)$,

Output: a partition $(S \subseteq V, T := V \setminus S)$ of V so as to maximize $|E(S,T)|$,
where $E(S,T) = \{uv \in E : |\{u,v\} \cap S| = 1\}$

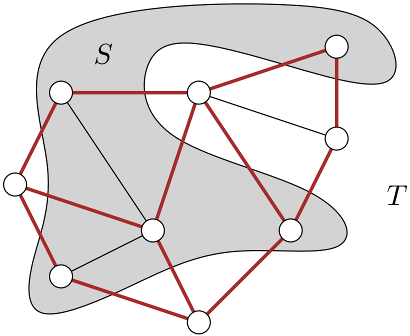


- Min-Uncut: remove minimum number of edges to make graph bipartite

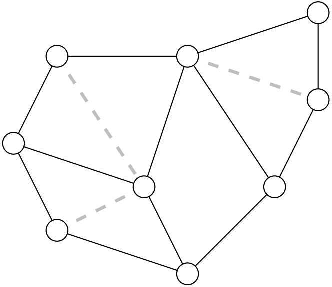
Maximum Cut Problem

Input: $G = (V, E)$,

Output: a partition $(S \subseteq V, T := V \setminus S)$ of V so as to maximize $|E(S,T)|$,
where $E(S,T) = \{uv \in E : |\{u,v\} \cap S| = 1\}$



- Min-Uncut: remove minimum number of edges to make graph bipartite



- Max-Cut = Min-Uncut for exact algorithms, but not the same for approximation algorithms

- Recap: $1/2$-approximation algorithms for Max-Cut:

Randomized Algorithm

1: $S \leftarrow \emptyset$
2: **for** every $u \in V$ **do**
3: with probability $1/2$, add u to S
4: **return** $(S, V \setminus S)$

- Max-Cut = Min-Uncut for exact algorithms, but not the same for approximation algorithms
- Recap: $1/2$-approximation algorithms for Max-Cut:

Randomized Algorithm

```
S ← ∅
for every u ∈ V do
    with probability 1/2, add u to S
return (S, V \ S)
```

Greedy Algorithms

```
S ← ∅, T ← ∅
for every u ∈ V do
    if |E(u, S)| > |E(u, T)| then
        T ← T ∪ {u}
    else
        S ← S ∪ {u}
return (S, T)
```

- Max-Cut = Min-Uncut for exact algorithms, but not the same for approximation algorithms

- Recap: $1/2$-approximation algorithms for Max-Cut:

Randomized Algorithm

```
S ← ∅
for every u ∈ V do
    with probability 1/2, add u to S
return (S, V \ S)
```

Greedy Algorithms

```
S ← ∅, T ← ∅
for every u ∈ V do
    if |E(u, S)| > |E(u, T)| then
        T ← T ∪ {u}
    else
        S ← S ∪ {u}
return (S, T)
```

- Local Search: while we can improve the solution by switching the side of one vertex, perform the operation, stop if no swapping can improve the solution

Linear Programming Relaxation

First Attempt

- $y_v, v \in V$: if $v \in S$
- $x_{uv}, uv \in E$: if uv is cut

$$\max \quad \sum_{uv \in E} x_{uv}$$

$$x_{uv} \leq |y_u - y_v| \quad \forall uv \in E$$
$$y_v \in [0,1] \quad \forall v \in V$$

Linear Programming Relaxation

First Attempt

- $y_v, v \in V$: if $v \in S$
- $x_{uv}, uv \in E$: if uv is cut

$$\max \quad \sum_{uv \in E} x_{uv}$$

$$x_{uv} \leq |y_u - y_v| \qquad \forall uv \in E$$
$$y_v \in [0,1] \qquad \forall v \in V$$

- $x_{uv} \leq |y_u - y_v|$ is not linear

Linear Programming Relaxation

First Attempt

- $y_v, v \in V$: if $v \in S$
- $x_{uv}, uv \in E$: if uv is cut

$$\max \quad \sum_{uv \in E} x_{uv}$$

$$x_{uv} \leq |y_u - y_v| \qquad \forall uv \in E$$
$$y_v \in [0, 1] \qquad \forall v \in V$$

- $x_{uv} \leq |y_u - y_v|$ is not linear
- feasible region is not convex:

y_u	y_v	x_{uv}	Y/N
1	0	0.5	Y
0	1	0.5	Y
0.5	0.5	0.5	N

Linear Programming Relaxation

First Attempt

- $y_v, v \in V$: if $v \in S$
- $x_{uv}, uv \in E$: if uv is cut

$$\max \quad \sum_{uv \in E} x_{uv}$$

$$x_{uv} \le |y_u - y_v| \qquad \forall uv \in E$$
$$y_v \in [0, 1] \qquad \forall v \in V$$

- $x_{uv} \le |y_u - y_v|$ is not linear
- feasible region is not convex:

y_u	y_v	x_{uv}	Y/N
1	0	0.5	Y
0	1	0.5	Y
0.5	0.5	0.5	N

- $x_{uv} \ge |y_u - y_v|$ can be replaced by $x_{uv} \ge y_u - y_v$ and $x_{uv} \ge y_v - y_u$

Second Attempt

- $x_{uv}, uv \in \binom{V}{2}$: whether uv is cut

$$\min \sum_{u,v \in V, u<v} x_{uv}$$

$$\begin{aligned} x_{uv} + x_{vw} + x_{uw} \leq 2 & \qquad \forall u, v, w \in V \\ x_{uv} \in [0, 1] & \qquad \forall u, v \in V \end{aligned}$$

Second Attempt

- $x_{uv}, uv \in \binom{V}{2}$: whether uv is cut

$$\min \sum_{u,v \in V, u<v} x_{uv}$$

$$x_{uv} + x_{vw} + x_{uw} \leq 2 \qquad \forall u, v, w \in V$$

$$x_{uv} \in [0, 1] \qquad \forall u, v \in V$$

- The integrality gap of the LP is $2 - \epsilon$: there is an instance, where opt $\approx |E|/2$ and lp $\approx |E|$

Quadratic Program

- $y_v = \begin{cases} 1 & \text{if } v \in S \\ -1 & \text{if } v \notin S \end{cases}$

$$\max \quad \frac{1}{2} \sum_{uv \in E} (1 - y_u y_v)$$

$$y_v \in \{\pm 1\} \quad \forall v \in V$$

Quadratic Program

- $y_v = \begin{cases} 1 & \text{if } v \in S \\ -1 & \text{if } v \notin S \end{cases}$

$$\max \quad \frac{1}{2}\sum_{uv \in E}(1 - y_u y_v)$$

$$y_v \in \{\pm 1\} \quad \forall v \in V$$

Semi-Definite Program

- $y_v \in \mathbb{R}^n, \forall v \in V$

$$\max \quad \frac{1}{2}\sum_{uv \in E}(1 - \langle y_u, y_v \rangle)$$

$$|y_v| = 1 \quad \forall v \in V$$

- $\langle y_u, y_v \rangle = y_u^{\mathrm{T}} y_v = \sum_{i=1}^{n} y_{u,i} \cdot y_{v,i}$: inner product of y_u and y_v

Quadratic Program

- $y_v = \begin{cases} 1 & \text{if } v \in S \\ -1 & \text{if } v \notin S \end{cases}$

$$\max \quad \frac{1}{2} \sum_{uv \in E} (1 - y_u y_v)$$

$$y_v \in \{\pm 1\} \quad \forall v \in V$$

Semi-Definite Program

- $y_v \in \mathbb{R}^n, \forall v \in V$

$$\max \quad \frac{1}{2} \sum_{uv \in E} (1 - \langle y_u, y_v \rangle)$$

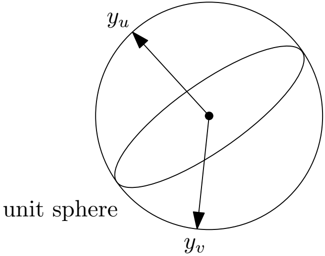
$$|y_v| = 1 \quad \forall v \in V$$

- $\langle y_u, y_v \rangle = y_u^{\mathrm{T}} y_v = \sum_{i=1}^{n} y_{u,i} \cdot y_{v,i}$: inner product of y_u and y_v
- requiring $y_v \in \mathbb{R}^n$ is the same as requiring $y_v \in \mathbb{R}^{n'}$ for any $n' \geq n$

SDP for Max-Cut

$$\max \quad \frac{1}{2}\sum_{uv\in E}(1-\langle y_u, y_v\rangle)$$

$$|y_v| = 1 \quad \forall v \in V$$



SDP for Max-Cut

$$\max \qquad \frac{1}{2}\sum_{uv\in E}(1-\langle y_u, y_v\rangle)$$

$$|y_v| = 1 \quad \forall v \in V$$

SDP for Max-Cut

$$\max \qquad \frac{1}{2}\sum_{uv\in E}(1-\langle y_u, y_v\rangle)$$

$$|y_v| = 1 \quad \forall v \in V$$

- SDP is a relaxation:

$$y_v = \begin{cases} (1,0,0,0,\cdots,0) & \text{if } v \in S \\ (-1,0,0,0,\cdots,0) & \text{if } v \in T \end{cases}$$

- sdp: the value of the SDP, sdp $\geq$ opt

SDP for Max-Cut

$$\max \qquad \frac{1}{2}\sum_{uv\in E}(1-\langle y_u, y_v\rangle)$$

$$|y_v| = 1 \quad \forall v \in V$$

- SDP is a relaxation:

$$y_v = \begin{cases} (1,0,0,0,\cdots,0) & \text{if } v \in S \\ (-1,0,0,0,\cdots,0) & \text{if } v \in T \end{cases}$$

- sdp: the value of the SDP, $\quad$ sdp $\geq$ opt

Q: Can we solve the SDP?

SDP for Max-Cut

$$\max \qquad \frac{1}{2}\sum_{uv\in E}(1-\langle y_u, y_v\rangle)$$

$$|y_v| = 1 \quad \forall v \in V$$

- SDP is a relaxation:

$$y_v = \begin{cases} (1,0,0,0,\cdots,0) & \text{if } v \in S \\ (-1,0,0,0,\cdots,0) & \text{if } v \in T \end{cases}$$

- sdp: the value of the SDP, $\quad$ sdp $\geq$ opt

Q: Can we solve the SDP?

A: Yes

Outline

Def. A symmetric matrix $X \in \mathbb{R}^{n\times n}$ is Positive Semi-Definite (PSD) if $\forall y \in \mathbb{R}^n$, we have $y^{\mathrm{T}} X y \geq 0$. Use $X \succeq 0$ to denote X is PSD.

- $X \succeq X'$ means $X - X' \succeq 0$.

Def. A symmetric matrix $X \in \mathbb{R}^{n\times n}$ is Positive Semi-Definite (PSD) if $\forall y \in \mathbb{R}^n$, we have $y^{\mathrm{T}} X y \geq 0$. Use $X \succeq 0$ to denote X is PSD.

- $X \succeq X'$ means $X - X' \succeq 0$.

Lemma The following statements are equivalent for a symmetric matrix $X \in \mathbb{R}^{n\times n}$:

- $X \succeq 0$
- All the n eigenvalues of X are non-negative
- $X = V^{\mathrm{T}} V$ for some $V \in \mathbb{R}^{m\times n}, m \leq n$
- $X = \sum_{u=1}^{n} \lambda_u w_u w_u^{\mathrm{T}}$ for some reals $\lambda_1, \lambda_2, \cdots, \lambda_n \geq 0$ and orthnormal basis $\{w_u\}_{u\in[n]}$

Semi-definite Programming (SDP)

- matrices of size $n \times n \equiv$ flattened vectors of length n^2:
  - Use $\cdot$ as multiplication for flattened matrices,
  - $X \succeq 0$: view X as a matrix.

Semi-definite Programming (SDP)

- matrices of size $n \times n \equiv$ flattened vectors of length n^2:
  - Use $\cdot$ as multiplication for flattened matrices,
  - $X \succeq 0$: view X as a matrix.
- $A \in \mathbb{R}^{m \times n^2}, b \in \mathbb{R}^m, c \in \mathbb{R}^{n^2}$
- Assume A_k's and c are symmetric matrices of size $n \times n$

Semi-definite Programming (SDP)

- matrices of size $n \times n \equiv$ flattened vectors of length n^2:
  - Use $\cdot$ as multiplication for flattened matrices,
  - $X \succeq 0$: view X as a matrix.
- $A \in \mathbb{R}^{m \times n^2}, b \in \mathbb{R}^m, c \in \mathbb{R}^{n^2}$
- Assume A_k's and c are symmetric matrices of size $n \times n$

Semi-Definite Program

$$\min \quad c^{\mathrm{T}} \cdot X$$

$$A \cdot X \geq b$$

$$X \succeq 0$$

Semi-definite Programming (SDP)

- matrices of size $n \times n \equiv$ flattened vectors of length n^2:
  - Use $\cdot$ as multiplication for flattened matrices,
  - $X \succeq 0$: view X as a matrix.
- $A \in \mathbb{R}^{m \times n^2}, b \in \mathbb{R}^m, c \in \mathbb{R}^{n^2}$
- Assume A_k's and c are symmetric matrices of size $n \times n$

Semi-Definite Program

$$\min \quad c^{\mathrm{T}} \cdot X$$
$$A \cdot X \geq b$$
$$X \succeq 0$$

An equivalent formulation

$$\min \quad \sum_{u,v \in [n]} c_{u,v} \cdot \langle y_u, y_v \rangle$$
$$\sum_{u,v} a_{k,u,v} \langle y_u, y_v \rangle \geq b_k \quad \forall k \in [m]$$
$$y_v \in \mathbb{R}^n \quad \forall v \in [n]$$

Semi-definite Programming (SDP)

- matrices of size $n \times n \equiv$ flattened vectors of length n^2:
  - Use $\cdot$ as multiplication for flattened matrices,
  - $X \succeq 0$: view X as a matrix.
- $A \in \mathbb{R}^{m \times n^2}, b \in \mathbb{R}^m, c \in \mathbb{R}^{n^2}$
- Assume A_k's and c are symmetric matrices of size $n \times n$

Semi-Definite Program

$$\begin{aligned} \min \quad & c^{\mathrm{T}} \cdot X \\ & A \cdot X \geq b \\ & X \succeq 0 \end{aligned}$$

An equivalent formulation

$$\begin{aligned} \min \quad & \sum_{u,v \in [n]} c_{u,v} \cdot \langle y_u, y_v \rangle \\ & \sum_{u,v} a_{k,u,v} \langle y_u, y_v \rangle \geq b_k \quad \forall k \in [m] \\ & y_v \in \mathbb{R}^n \quad \forall v \in [n] \end{aligned}$$

- requiring $y_v \in \mathbb{R}^n$ is the same as requiring $y_v \in \mathbb{R}^{n'}$ for any $n' \geq n$

Semi-Definite Program

$$\min \quad c^{\mathrm{T}} \cdot X$$

$$A \cdot X \geq b$$

$$X \succeq 0$$

Semi-Definite Program

$$\min \quad c^{\mathrm{T}} \cdot X$$

$$A \cdot X \geq b$$
$$X \succeq 0$$

Example

$$\min \quad 5y_1 + 6y_2 + 7y_3$$

$$y_1 + 3y_2 + 4y_3 \geq 5$$
$$2y_1 + 3y_2 + y_3 \geq 10$$
$$3y_1 + 2y_2 + 2y_3 \geq 7$$
$$\begin{pmatrix} y_1 & y_2 \\ y_2 & y_3 \end{pmatrix} \succeq 0$$

Semi-Definite Program

$$\min \quad c^{\mathrm{T}} \cdot X$$

$$A \cdot X \geq b$$
$$X \succeq 0$$

Example

$$\min \quad 5y_1 + 6y_2 + 7y_3$$

$$y_1 + 3y_2 + 4y_3 \geq 5$$
$$2y_1 + 3y_2 + y_3 \geq 10$$
$$3y_1 + 2y_2 + 2y_3 \geq 7$$
$$\begin{pmatrix} y_1 & y_2 \\ y_2 & y_3 \end{pmatrix} \succeq 0$$

- $X \succeq 0 \Longleftrightarrow X_{u,v} = X_{v,u}, \forall u, v \in [n]; (yy^{\mathrm{T}}) \cdot X \geq 0, \forall y \in \mathbb{R}^n$.
- SDP $\equiv$ LP with infinite number of linear constraints

Seperation Oracle $\mathcal{O}$

- Given a symmetric $X \in \mathbb{R}^{n\times n}$, we need to either claim $X \succeq 0$, or return a $y \in \mathbb{R}^n$ such that $y^{\mathrm{T}} X y < 0$.

Seperation Oracle $\mathcal{O}$

- Given a symmetric $X \in \mathbb{R}^{n\times n}$, we need to either claim $X \succeq 0$, or return a $y \in \mathbb{R}^n$ such that $y^{\mathrm{T}} X y < 0$.
- QR decomposition finds eigenvalues and eigenvectors of X.

Seperation Oracle $\mathcal{O}$

- Given a symmetric $X \in \mathbb{R}^{n \times n}$, we need to either claim $X \succeq 0$, or return a $y \in \mathbb{R}^n$ such that $y^{\mathrm{T}} X y < 0$.
- QR decomposition finds eigenvalues and eigenvectors of X.

Recall: Ellipsoid Method

Seperation Oracle $\mathcal{O}$

- Given a symmetric $X \in \mathbb{R}^{n\times n}$, we need to either claim $X \succeq 0$, or return a $y \in \mathbb{R}^n$ such that $y^{\mathrm{T}} X y < 0$.
- QR decomposition finds eigenvalues and eigenvectors of X.

Recall: Ellipsoid Method

- maintain an ellipsoid that contains the feasible region

Seperation Oracle $\mathcal{O}$

- Given a symmetric $X \in \mathbb{R}^{n\times n}$, we need to either claim $X \succeq 0$, or return a $y \in \mathbb{R}^n$ such that $y^{\mathrm{T}} X y < 0$.
- QR decomposition finds eigenvalues and eigenvectors of X.

Recall: Ellipsoid Method

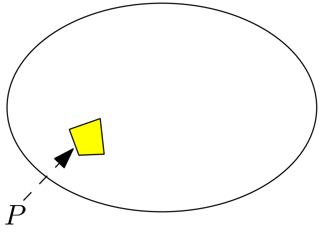
- maintain an ellipsoid that contains the feasible region
- query $\mathcal{O}$ if the center of ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat

Seperation Oracle $\mathcal{O}$

- Given a symmetric $X \in \mathbb{R}^{n\times n}$, we need to either claim $X \succeq 0$, or return a $y \in \mathbb{R}^n$ such that $y^{\mathrm{T}} X y < 0$.
- QR decomposition finds eigenvalues and eigenvectors of X.

Recall: Ellipsoid Method

- maintain an ellipsoid that contains the feasible region
- query $\mathcal{O}$ if the center of ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat

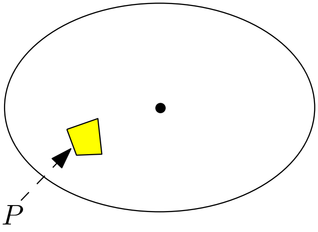


Seperation Oracle $\mathcal{O}$

- Given a symmetric $X \in \mathbb{R}^{n\times n}$, we need to either claim $X \succeq 0$, or return a $y \in \mathbb{R}^n$ such that $y^{\mathrm{T}} X y < 0$.
- QR decomposition finds eigenvalues and eigenvectors of X.

Recall: Ellipsoid Method

- maintain an ellipsoid that contains the feasible region
- query $\mathcal{O}$ if the center of ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat

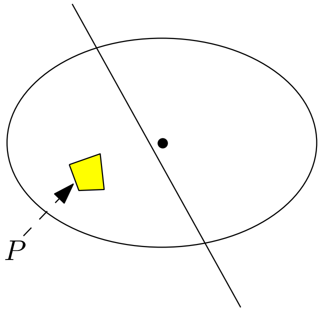


Seperation Oracle $\mathcal{O}$

- Given a symmetric $X \in \mathbb{R}^{n\times n}$, we need to either claim $X \succeq 0$, or return a $y \in \mathbb{R}^n$ such that $y^{\mathrm{T}}Xy < 0$.
- QR decomposition finds eigenvalues and eigenvectors of X.

Recall: Ellipsoid Method

- maintain an ellipsoid that contains the feasible region
- query $\mathcal{O}$ if the center of ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat

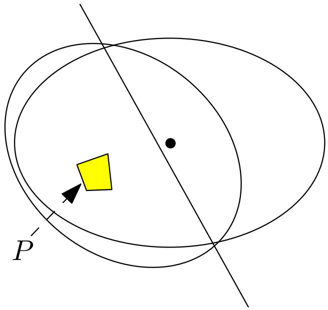


Seperation Oracle $\mathcal{O}$

- Given a symmetric $X \in \mathbb{R}^{n\times n}$, we need to either claim $X \succeq 0$, or return a $y \in \mathbb{R}^n$ such that $y^{\mathrm{T}}Xy < 0$.
- QR decomposition finds eigenvalues and eigenvectors of X.

Recall: Ellipsoid Method

- maintain an ellipsoid that contains the feasible region
- query $\mathcal{O}$ if the center of ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat

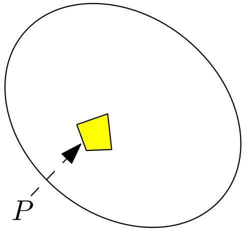


Seperation Oracle $\mathcal{O}$

- Given a symmetric $X \in \mathbb{R}^{n\times n}$, we need to either claim $X \succeq 0$, or return a $y \in \mathbb{R}^n$ such that $y^{\mathrm{T}} X y < 0$.
- QR decomposition finds eigenvalues and eigenvectors of X.

Recall: Ellipsoid Method

- maintain an ellipsoid that contains the feasible region
- query $\mathcal{O}$ if the center of ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat

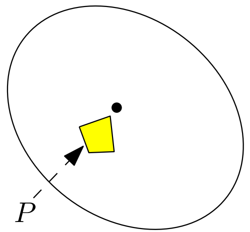


Seperation Oracle $\mathcal{O}$

- Given a symmetric $X \in \mathbb{R}^{n\times n}$, we need to either claim $X \succeq 0$, or return a $y \in \mathbb{R}^n$ such that $y^{\mathrm{T}} X y < 0$.
- QR decomposition finds eigenvalues and eigenvectors of X.

Recall: Ellipsoid Method

- maintain an ellipsoid that contains the feasible region
- query $\mathcal{O}$ if the center of ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat

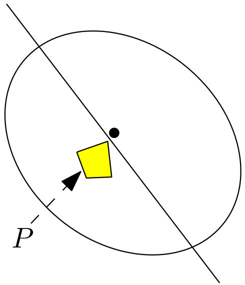


Seperation Oracle $\mathcal{O}$

- Given a symmetric $X \in \mathbb{R}^{n \times n}$, we need to either claim $X \succeq 0$, or return a $y \in \mathbb{R}^n$ such that $y^{\mathrm{T}} X y < 0$.
- QR decomposition finds eigenvalues and eigenvectors of X.

Recall: Ellipsoid Method

- maintain an ellipsoid that contains the feasible region
- query $\mathcal{O}$ if the center of ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat

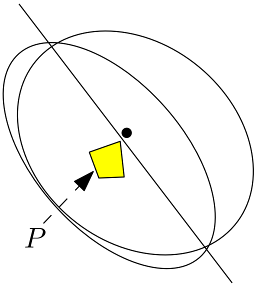


Seperation Oracle $\mathcal{O}$

- Given a symmetric $X \in \mathbb{R}^{n\times n}$, we need to either claim $X \succeq 0$, or return a $y \in \mathbb{R}^n$ such that $y^{\mathrm{T}} X y < 0$.
- QR decomposition finds eigenvalues and eigenvectors of X.

Recall: Ellipsoid Method

- maintain an ellipsoid that contains the feasible region
- query $\mathcal{O}$ if the center of ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat

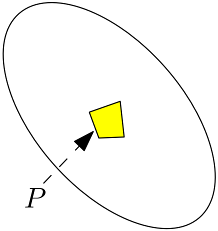


Seperation Oracle $\mathcal{O}$

- Given a symmetric $X \in \mathbb{R}^{n \times n}$, we need to either claim $X \succeq 0$, or return a $y \in \mathbb{R}^n$ such that $y^{\mathrm{T}} X y < 0$.
- QR decomposition finds eigenvalues and eigenvectors of X.

Recall: Ellipsoid Method

- maintain an ellipsoid that contains the feasible region
- query $\mathcal{O}$ if the center of ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat

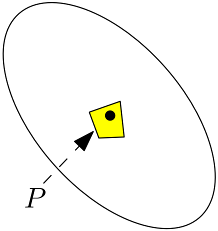


Seperation Oracle $\mathcal{O}$

- Given a symmetric $X \in \mathbb{R}^{n\times n}$, we need to either claim $X \succeq 0$, or return a $y \in \mathbb{R}^n$ such that $y^{\mathrm{T}}Xy < 0$.
- QR decomposition finds eigenvalues and eigenvectors of X.

Recall: Ellipsoid Method

- maintain an ellipsoid that contains the feasible region
- query $\mathcal{O}$ if the center of ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat

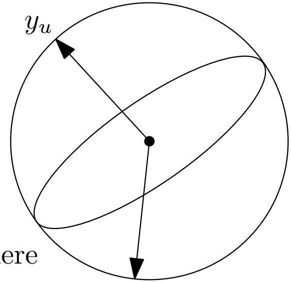


Outline

SDP for Max-Cut

$$\max \qquad \frac{1}{2}\sum_{uv\in E}(1-\langle y_u, y_v\rangle)$$

$$|y_v| = 1 \quad \forall v \in V$$

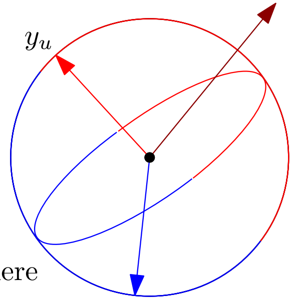


- let $(y_v)_{v\in V}$ be the vectors obtained from solving SDP
- sdp $= \frac{1}{2}\sum_{uv\in E}(1 - y_u^{\mathrm{T}} y_v) \geq$ opt

SDP for Max-Cut

$$\max \quad \frac{1}{2}\sum_{uv\in E}(1-\langle y_u, y_v\rangle)$$

$$|y_v| = 1 \quad \forall v \in V$$

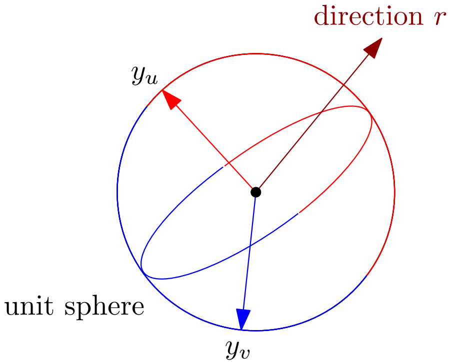


- let $(y_v)_{v\in V}$ be the vectors obtained from solving SDP
- sdp $= \frac{1}{2}\sum_{uv\in E}(1-y_u^{\mathrm{T}}y_v) \geq$ opt

SDP for Max-Cut

$$\max \quad \frac{1}{2} \sum_{uv \in E} (1 - \langle y_u, y_v \rangle)$$

$$|y_v| = 1 \quad \forall v \in V$$

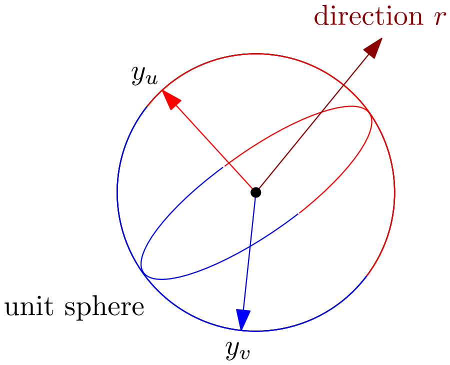


- let $(y_v)_{v \in V}$ be the vectors obtained from solving SDP
- sdp $= \frac{1}{2} \sum_{uv \in E} (1 - y_u^{\mathrm{T}} y_v) \geq$ opt

SDP for Max-Cut

$$\max \quad \frac{1}{2}\sum_{uv\in E}(1-\langle y_u, y_v\rangle)$$

$$|y_v| = 1 \quad \forall v \in V$$



- let $(y_v)_{v\in V}$ be the vectors obtained from solving SDP
- sdp $= \frac{1}{2}\sum_{uv\in E}(1 - y_u^{\mathrm{T}} y_v) \geq$ opt

[Goemans-Williamson'95] Rounding Algorithm

1: randomly choose a direction $r \in \mathbb{R}^n$:
- choose each $r_u \sim N(0,1)$ i.i.d

$N(0,1)$: standard normal distribution

2: $\bar{y}_v = \mathrm{sgn}(\langle y_v, r\rangle)$, $S = \{v \in V : \bar{y}_v > 0\}$, **return** $(S, V\setminus S)$

$$\Pr[uv \text{ is cut}] = \frac{\text{radian angle between } y_u \text{ and } y_v}{\pi} = \frac{\arccos\langle y_u, y_v\rangle}{\pi}$$

$$\Pr[uv \text{ is cut}] = \frac{\text{radian angle between } y_u \text{ and } y_v}{\pi} = \frac{\arccos\langle y_u, y_v\rangle}{\pi}$$

$$\frac{\Pr[uv \text{ is cut}]}{\frac{1}{2}(1 - \langle y_u, y_v\rangle)} = \frac{\frac{1}{\pi}\arccos\langle y_u, y_v\rangle}{\frac{1}{2}(1 - \langle y_u, y_v\rangle)}$$

$$= \frac{\frac{1}{\pi}\arccos(x)}{\frac{1}{2}(1 - x)}$$

$$x := \langle y_u, y_v\rangle \in [-1, 1]$$

$$\Pr[uv \text{ is cut}] = \frac{\text{radian angle between } y_u \text{ and } y_v}{\pi} = \frac{\arccos\langle y_u, y_v\rangle}{\pi}$$

$$\frac{\Pr[uv \text{ is cut}]}{\frac{1}{2}(1 - \langle y_u, y_v\rangle)} = \frac{\frac{1}{\pi}\arccos\langle y_u, y_v\rangle}{\frac{1}{2}(1 - \langle y_u, y_v\rangle)}$$

$$= \frac{\frac{1}{\pi}\arccos(x)}{\frac{1}{2}(1 - x)}$$

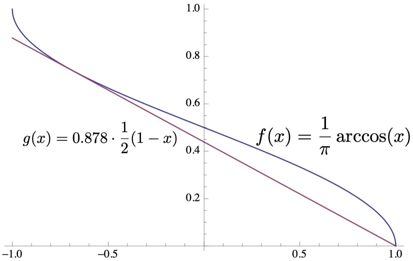
$$x := \langle y_u, y_v\rangle \in [-1, 1]$$

- $\alpha_{\text{GW}} := \inf_{x\in[-1,1]} \frac{2}{\pi} \cdot \frac{\arccos(x)}{(1-x)} \geq 0.878$

$$\Pr[uv \text{ is cut}] = \frac{\text{radian angle between } y_u \text{ and } y_v}{\pi} = \frac{\arccos\langle y_u, y_v\rangle}{\pi}$$

$$\frac{\Pr[uv \text{ is cut}]}{\frac{1}{2}(1 - \langle y_u, y_v\rangle)} = \frac{\frac{1}{\pi}\arccos\langle y_u, y_v\rangle}{\frac{1}{2}(1 - \langle y_u, y_v\rangle)} = \frac{\frac{1}{\pi}\arccos(x)}{\frac{1}{2}(1-x)}$$

$$x := \langle y_u, y_v\rangle \in [-1, 1]$$



- $\alpha_{\mathsf{GW}} := \inf_{x\in[-1,1]} \frac{2}{\pi} \cdot \frac{\arccos(x)}{(1-x)} \geq 0.878$

$$\mathbb{E}[|E(S,T)|] = \sum_{uv \in E} \Pr[uv \text{ is cut}] \geq \alpha_{\mathsf{GW}} \sum_{uv \in E} \frac{1}{2}(1 - \langle y_u, y_v \rangle)$$
$$= \alpha_{\mathsf{GW}} \cdot \mathsf{sdp} \geq \alpha_{\mathsf{GW}} \cdot \mathsf{opt} \geq 0.878 \cdot \mathsf{opt}.$$

$$\mathbb{E}[|E(S,T)|] = \sum_{uv \in E} \Pr[uv \text{ is cut}] \geq \alpha_{\mathsf{GW}} \sum_{uv \in E} \frac{1}{2}(1 - \langle y_u, y_v \rangle)$$
$$= \alpha_{\mathsf{GW}} \cdot \mathsf{sdp} \geq \alpha_{\mathsf{GW}} \cdot \mathsf{opt} \geq 0.878 \cdot \mathsf{opt}.$$

- Assuming Unique Game Conjecture (UGC), no polynomial-time algorithm can give an approximation ratio of $\alpha_{\mathsf{GW}} + \epsilon$ for any constant $\epsilon > 0$.

Outline

1. Max-Cut Problem
2. Semi-Definite Programming
3. 0.878-Approximation for Max-Cut Using SDP
4. Duality for Semi-Definite Programming
5. Ellipsoid Method runs In Polynomial Time

Duality for Semi-Definite Programming

Semi-Definite Program

$$\min \quad c^{\mathrm{T}} \cdot X$$

$$A \cdot X \geq b$$

$$X \succeq 0$$

Duality for Semi-Definite Programming

Semi-Definite Program

$$\min \quad c^{\mathrm{T}} \cdot X$$
$$A \cdot X \geq b$$
$$X \succeq 0$$

Semi-Definite Program

$$\min \quad c^{\mathrm{T}} \cdot X$$
$$\sum_{u,v \in [n]} a_{k,u,v} X_{u,v} \geq b \qquad \forall k \in [m]$$
$$\sum_{u,v \in [n]} r_u r_v X_{u,v} \geq 0 \qquad \forall r \in \mathbb{R}^n$$

- replace $X \succeq 0$ with infinite number of linear constraints: $(r^{\mathrm{T}} r) \cdot X \geq 0, \forall r \in \mathbb{R}^n$.
- no symmetry constraint as A_k's and c are symmetric

Duality for Semi-Definite Programming

Semi-Definite Program

$$\min \quad c^{\mathrm{T}} \cdot X$$

$$A \cdot X \geq b$$

$$X \succeq 0$$

Semi-Definite Program

$$\min \quad c^{\mathrm{T}} \cdot X$$

$$\sum_{u,v \in [n]} a_{k,u,v} X_{u,v} \geq b \qquad \forall k \in [m]$$

$$\sum_{u,v \in [n]} r_u r_v X_{u,v} \geq 0 \qquad \forall r \in \mathbb{R}^n$$

Dual : $\max \quad \sum_{k=1}^{m} b_k \cdot y_k$

$$\sum_{k=1}^{m} a_{k,u,v} \cdot y_k + \sum_{r \in \mathbb{R}^n} r_u r_v \cdot z_r = c_{u,v} \qquad \forall u, v \in [n]$$

$$y_k \geq 0 \qquad \forall k \in [m]$$

$$z_r \geq 0 \qquad \forall r \in \mathbb{R}^n$$

Duality for Semi-Definite Programming

Dual : $\max \quad \sum_{k=1}^{m} b_k \cdot y_k$

$$\sum_{k=1}^{m} a_{k,u,v} \cdot y_k + \sum_{r \in \mathbb{R}^n} r_u r_v \cdot z_r = c_{u,v} \qquad \forall u, v \in [n]$$

$$y_k \geq 0 \qquad \forall k \in [m]$$

$$z_r \geq 0 \qquad \forall r \in \mathbb{R}^n$$

Duality for Semi-Definite Programming

Dual : $\max \quad \sum_{k=1}^{m} b_k \cdot y_k$

$$\sum_{k=1}^{m} a_{k,u,v} \cdot y_k + \sum_{r \in \mathbb{R}^n} r_u r_v \cdot z_r = c_{u,v} \qquad \forall u, v \in [n]$$

$$y_k \geq 0 \qquad \forall k \in [m]$$

$$z_r \geq 0 \qquad \forall r \in \mathbb{R}^n$$

- $\mathbb{R}^n$ is infinite. So the notion $\sum_{r \in \mathbb{R}^n}$ is bad. Informal.
- first red constraint $\Leftrightarrow A^{\mathrm{T}} y + \sum_{r \in \mathbb{R}^n} z_r \cdot r r^{\mathrm{T}} = c$
- $\sum_{r \in \mathbb{R}^n} z_r \cdot r r^{\mathrm{T}}$ is PSD

Duality for Semi-Definite Programming

Dual : $\max \quad \sum_{k=1}^{m} b_k \cdot y_k$

$$\sum_{k=1}^{m} a_{k,u,v} \cdot y_k + \sum_{r \in \mathbb{R}^n} r_u r_v \cdot z_r = c_{u,v} \qquad \forall u, v \in [n]$$

$$y_k \geq 0 \qquad \forall k \in [m]$$

$$z_r \geq 0 \qquad \forall r \in \mathbb{R}^n$$

- $\mathbb{R}^n$ is infinite. So the notion $\sum_{r \in \mathbb{R}^n}$ is bad. Informal.
- first red constraint $\Leftrightarrow A^{\mathrm{T}} y + \sum_{r \in \mathbb{R}^n} z_r \cdot r r^{\mathrm{T}} = c$
- $\sum_{r \in \mathbb{R}^n} z_r \cdot r r^{\mathrm{T}}$ is PSD
- moreover, any PSD matrix can be written is of this form

Duality for Semi-Definite Programming

Dual : $\max \quad \sum_{k=1}^{m} b_k \cdot y_k$

$$\sum_{k=1}^{m} a_{k,u,v} \cdot y_k + \sum_{r \in \mathbb{R}^n} r_u r_v \cdot z_r = c_{u,v} \qquad \forall u, v \in [n]$$

$$y_k \geq 0 \qquad \forall k \in [m]$$

$$z_r \geq 0 \qquad \forall r \in \mathbb{R}^n$$

- $\mathbb{R}^n$ is infinite. So the notion $\sum_{r \in \mathbb{R}^n}$ is bad. Informal.
- first red constraint $\Leftrightarrow A^{\mathrm{T}} y + \sum_{r \in \mathbb{R}^n} z_r \cdot r r^{\mathrm{T}} = c$
- $\sum_{r \in \mathbb{R}^n} z_r \cdot r r^{\mathrm{T}}$ is PSD
- moreover, any PSD matrix can be written is of this form
- $\Longrightarrow$ red constraints can be replaced by $A^{\mathrm{T}} y \preceq c$

Duality for Semi-Definite Programming

Semi-Definite Program

$$\min \quad c^{\mathrm{T}} \cdot X$$

$$A \cdot X \geq b$$

$$X \succeq 0$$

Dual for SDP

$$\max \quad b^{\mathrm{T}} y$$

$$A^{\mathrm{T}} y \preceq c$$

$$y \geq 0$$

Duality for Semi-Definite Programming

Semi-Definite Program

$$\min \quad c^{\mathrm{T}} \cdot X$$
$$A \cdot X \geq b$$
$$X \succeq 0$$

Dual for SDP

$$\max \quad b^{\mathrm{T}} y$$
$$A^{\mathrm{T}} y \preceq c$$
$$y \geq 0$$

- Linear Program: $X \geq 0$
- In Dual of LP: $A^{\mathrm{T}} y \leq c$

Outline

Focus on $\mathbb{R}^n$:

- axis-aligned ellipsoid centered at c with axis lengths $\mathcal{Q}_{c,a} := a \in \mathbb{R}^n_{>0}$: $\left\{x \in \mathbb{R}^n : \sum_{i\in[n]} \frac{(x_i - c_i)^2}{a_i^2} \leq 1\right\}$

Focus on $\mathbb{R}^n$:

- axis-aligned ellipsoid centered at c with axis lengths $\mathcal{Q}_{c,a} := a \in \mathbb{R}^n_{>0}$: $\left\{x \in \mathbb{R}^n : \sum_{i\in[n]} \frac{(x_i - c_i)^2}{a_i^2} \leq 1\right\}$
- axis-aligned half-ellipsoid: $\mathcal{R}_{c,a,w} := \left\{x \in \mathcal{Q}_{c,a} : w^{\mathrm{T}}(x - c) \geq 0\right\}$, $w \in \mathbb{R}^n$

Focus on $\mathbb{R}^n$:

- axis-aligned ellipsoid centered at c with axis lengths $\mathcal{Q}_{c,a} := a \in \mathbb{R}^n_{>0}$: $\left\{x \in \mathbb{R}^n : \sum_{i\in[n]} \frac{(x_i - c_i)^2}{a_i^2} \leq 1\right\}$
- axis-aligned half-ellipsoid: $\mathcal{R}_{c,a,w} := \left\{x \in \mathcal{Q}_{c,a} : w^{\mathrm{T}}(x-c) \geq 0\right\}$, $w \in \mathbb{R}^n$

Lemma For any axis-aligned axis-aligned half-ellipsoid $\mathcal{R}_{c,a,w}$, we can efficiently find an axis-aligned ellipsoid $\mathcal{Q}_{c',a'}$ such that

- $\mathcal{R}_{c,a,w} \subseteq \mathcal{Q}_{c',a'}$
- $\frac{\mathrm{vol}(\mathcal{Q}_{c',a'})}{\mathrm{vol}(\mathcal{Q}_{c,a})} \leq e^{-\frac{1}{2(n+1)}} = 1 - \Omega\left(\frac{1}{n}\right)$

Focus on $\mathbb{R}^n$:

- axis-aligned ellipsoid centered at c with axis lengths $\mathcal{Q}_{c,a} := a \in \mathbb{R}^n_{>0}$: $\left\{x \in \mathbb{R}^n : \sum_{i\in[n]} \frac{(x_i - c_i)^2}{a_i^2} \leq 1\right\}$
- axis-aligned half-ellipsoid: $\mathcal{R}_{c,a,w} := \left\{x \in \mathcal{Q}_{c,a} : w^{\mathrm{T}}(x-c) \geq 0\right\}$, $w \in \mathbb{R}^n$

Lemma For any axis-aligned axis-aligned half-ellipsoid $\mathcal{R}_{c,a,w}$, we can efficiently find an axis-aligned ellipsoid $\mathcal{Q}_{c',a'}$ such that

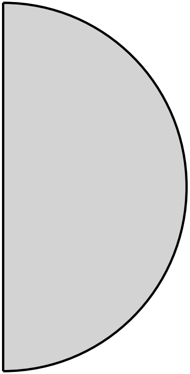
- $\mathcal{R}_{c,a,w} \subseteq \mathcal{Q}_{c',a'}$
- $\frac{\mathrm{vol}(\mathcal{Q}_{c',a'})}{\mathrm{vol}(\mathcal{Q}_{c,a})} \leq e^{-\frac{1}{2(n+1)}} = 1 - \Omega\left(\frac{1}{n}\right)$

Proof.

- we can assume $c = 0, a = 1$ and $w = (1, 0, 0, 0, \cdots, 0)^{\mathrm{T}}$.
- half-ellipsoid becomes half ball: $\{x \in \mathbb{R}^n : |x|_2 \leq 1, x_1 \geq 0\}$

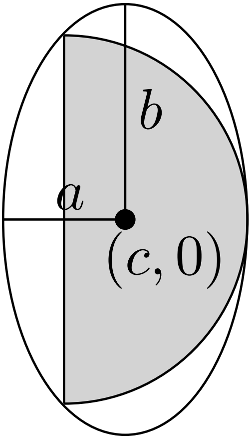
Proof.

- center of new ellipsoid : $(c, 0, \cdots, 0)$, $c \in [0, 1]$



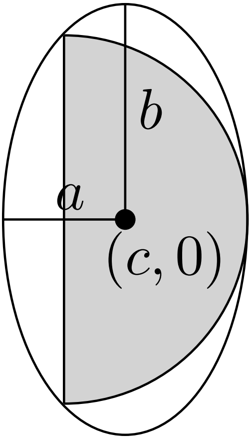
Proof.

- center of new ellipsoid : $(c, 0, \cdots, 0)$, $c \in [0, 1]$
- axis lengths: $(a, b, b, \cdots, b), a < b$



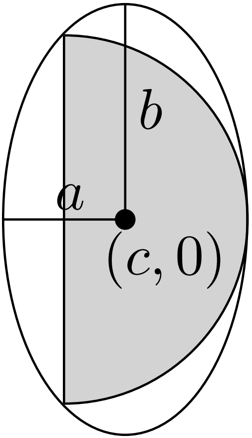
Proof.

- center of new ellipsoid : $(c, 0, \cdots, 0)$, $c \in [0, 1]$
- axis lengths: $(a, b, b, \cdots, b), a < b$
- the ellipsoid: $\frac{(x_1 - c)^2}{a^2} + \sum_{i=2}^{n} \frac{x_i^2}{b^2} \leq 1$



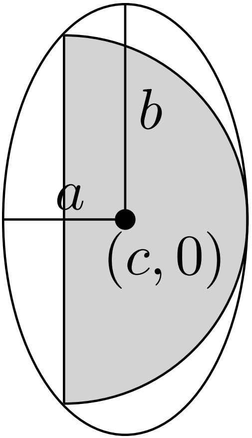
Proof.

- center of new ellipsoid : $(c, 0, \cdots, 0)$, $c \in [0, 1]$
- axis lengths: $(a, b, b, \cdots, b), a < b$
- the ellipsoid: $\frac{(x_1-c)^2}{a^2} + \sum_{i=2}^{n} \frac{x_i^2}{b^2} \leq 1$
- $(1, 0, 0, \cdots, 0)$ in ellipsoid: $\frac{(1-c)^2}{a^2} \leq 1$



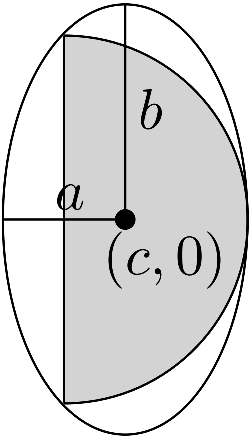
Proof.

- center of new ellipsoid : $(c, 0, \cdots, 0)$, $c \in [0, 1]$
- axis lengths: $(a, b, b, \cdots, b), a < b$
- the ellipsoid: $\frac{(x_1-c)^2}{a^2} + \sum_{i=2}^{n} \frac{x_i^2}{b^2} \leq 1$
- $(1, 0, 0, \cdots, 0)$ in ellipsoid: $\frac{(1-c)^2}{a^2} \leq 1$
- $(0, 1, 0, \cdots, 0)$ in ellipsoid: $\frac{c^2}{a^2} + \frac{1}{b^2} \leq 1$



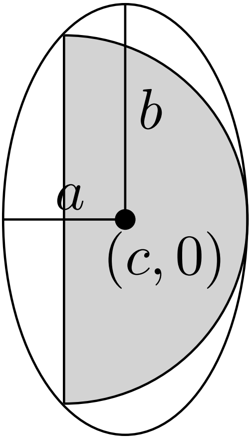
Proof.

- center of new ellipsoid : $(c, 0, \cdots, 0)$, $c \in [0, 1]$
- axis lengths: $(a, b, b, \cdots, b), a < b$
- the ellipsoid: $\frac{(x_1-c)^2}{a^2} + \sum_{i=2}^n \frac{x_i^2}{b^2} \le 1$
- $(1, 0, 0, \cdots, 0)$ in ellipsoid: $\frac{(1-c)^2}{a^2} \le 1$
- $(0, 1, 0, \cdots, 0)$ in ellipsoid: $\frac{c^2}{a^2} + \frac{1}{b^2} \le 1$
- set a, b and c so that both constraints are tight: $a = 1 - c, b = \sqrt{\frac{(1-c)^2}{(1-c)^2-c^2}} = \frac{1-c}{\sqrt{1-2c}}$



Proof.

- center of new ellipsoid : $(c, 0, \cdots, 0)$, $c \in [0, 1]$
- axis lengths: $(a, b, b, \cdots, b), a < b$
- the ellipsoid: $\frac{(x_1 - c)^2}{a^2} + \sum_{i=2}^{n} \frac{x_i^2}{b^2} \leq 1$
- $(1, 0, 0, \cdots, 0)$ in ellipsoid: $\frac{(1-c)^2}{a^2} \leq 1$
- $(0, 1, 0, \cdots, 0)$ in ellipsoid: $\frac{c^2}{a^2} + \frac{1}{b^2} \leq 1$
- set a, b and c so that both constraints are tight: $a = 1 - c, b = \sqrt{\frac{(1-c)^2}{(1-c)^2 - c^2}} = \frac{1-c}{\sqrt{1-2c}}$

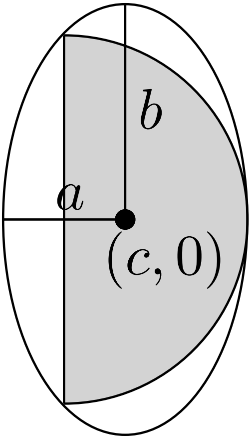


Proof.

- we won't prove that the ellipsoid contains all points in half ball.

Proof.

- center of new ellipsoid : $(c, 0, \cdots, 0)$, $c \in [0, 1]$
- axis lengths: $(a, b, b, \cdots, b), a < b$
- the ellipsoid: $\frac{(x_1-c)^2}{a^2} + \sum_{i=2}^{n} \frac{x_i^2}{b^2} \leq 1$
- $(1, 0, 0, \cdots, 0)$ in ellipsoid: $\frac{(1-c)^2}{a^2} \leq 1$
- $(0, 1, 0, \cdots, 0)$ in ellipsoid: $\frac{c^2}{a^2} + \frac{1}{b^2} \leq 1$
- set a, b and c so that both constraints are tight: $a = 1 - c, b = \sqrt{\frac{(1-c)^2}{(1-c)^2-c^2}} = \frac{1-c}{\sqrt{1-2c}}$

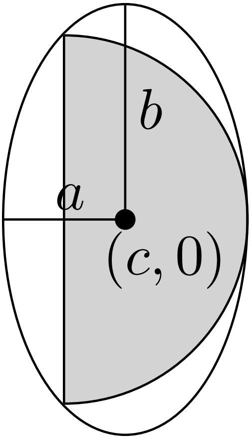


Proof.

- we won't prove that the ellipsoid contains all points in half ball.
- volume of ellipsoid is minimized when $c = \frac{1}{n+1}$

Proof.

- center of new ellipsoid : $(c, 0, \cdots, 0)$, $c \in [0, 1]$
- axis lengths: $(a, b, b, \cdots, b), a < b$
- the ellipsoid: $\frac{(x_1-c)^2}{a^2} + \sum_{i=2}^{n} \frac{x_i^2}{b^2} \leq 1$
- $(1, 0, 0, \cdots, 0)$ in ellipsoid: $\frac{(1-c)^2}{a^2} \leq 1$
- $(0, 1, 0, \cdots, 0)$ in ellipsoid: $\frac{c^2}{a^2} + \frac{1}{b^2} \leq 1$
- set a, b and c so that both constraints are tight: $a = 1 - c, b = \sqrt{\frac{(1-c)^2}{(1-c)^2-c^2}} = \frac{1-c}{\sqrt{1-2c}}$



Proof.

- we won't prove that the ellipsoid contains all points in half ball.
- volume of ellipsoid is minimized when $c = \frac{1}{n+1}$
- $a = \frac{n}{n+1}, b = \frac{n/(n+1)}{\sqrt{(n-1)/(n+1)}}$

Proof.

- $a = \frac{n}{n+1}, b = \frac{n/(n+1)}{\sqrt{(n-1)/(n+1)}}$

Proof.

- $a = \frac{n}{n+1}, b = \frac{n/(n+1)}{\sqrt{(n-1)/(n+1)}}$

$$\frac{\text{vol(ellipsoid)}}{\text{vol(unit ball)}} = ab^{n-1} = \left(\frac{n}{n+1}\right)^n \cdot \left(\frac{n+1}{n-1}\right)^{\frac{n-1}{2}}$$
$$= \left(\frac{n^2}{n^2-1}\right)^{\frac{n-1}{2}} \cdot \frac{n}{n+1}$$
$$\ln \frac{\text{vol(ellipsoid)}}{\text{vol(unit ball)}} = \frac{n-1}{2} \ln \frac{n^2}{n^2-1} + \ln \frac{n}{n+1}$$
$$\leq \frac{n-1}{2} \cdot \frac{1}{n^2-1} - \frac{1}{n+1} = -\frac{1}{2(n+1)}$$

- we used $\ln(1+x) \leq x, \forall x > -1$

Proof.

- $a = \frac{n}{n+1}, b = \frac{n/(n+1)}{\sqrt{(n-1)/(n+1)}}$

$$\frac{\text{vol(ellipsoid)}}{\text{vol(unit ball)}} = ab^{n-1} = \left(\frac{n}{n+1}\right)^n \cdot \left(\frac{n+1}{n-1}\right)^{\frac{n-1}{2}}$$
$$= \left(\frac{n^2}{n^2-1}\right)^{\frac{n-1}{2}} \cdot \frac{n}{n+1}$$
$$\ln \frac{\text{vol(ellipsoid)}}{\text{vol(unit ball)}} = \frac{n-1}{2} \ln \frac{n^2}{n^2-1} + \ln \frac{n}{n+1}$$
$$\leq \frac{n-1}{2} \cdot \frac{1}{n^2-1} - \frac{1}{n+1} = -\frac{1}{2(n+1)}$$

- we used $\ln(1+x) \leq x, \forall x > -1$
- $\frac{\text{vol(ellipsoid)}}{\text{vol(unit ball)}} \leq e^{-\frac{1}{2(n+1)}}$ □

Lemma For any axis-aligned axis-aligned half-ellipsoid $\mathcal{R}_{c,a,w}$, we can efficiently find an axis-aligned ellipsoid $\mathcal{Q}_{c',a'}$ such that

- $\mathcal{R}_{c,a,w} \subseteq \mathcal{Q}_{c',a'}$
- $\frac{\mathsf{vol}(\mathcal{Q}_{c',a'})}{\mathsf{vol}(\mathcal{Q}_{c,a})} \leq e^{-\frac{1}{2(n+1)}} = 1 - \Omega\left(\frac{1}{n}\right)$

Lemma For any axis-aligned axis-aligned half-ellipsoid $\mathcal{R}_{c,a,w}$, we can efficiently find an axis-aligned ellipsoid $\mathcal{Q}_{c',a'}$ such that

- $\mathcal{R}_{c,a,w} \subseteq \mathcal{Q}_{c',a'}$
- $\frac{\mathrm{vol}(\mathcal{Q}_{c',a'})}{\mathrm{vol}(\mathcal{Q}_{c,a})} \leq e^{-\frac{1}{2(n+1)}} = 1 - \Omega\left(\frac{1}{n}\right)$

Assumption

- The initial polytope is contained in a ball of radius R, where $R \leq 2^{\text{poly(input size)}}$.
- When the polytope is not empty, it contains a ball of radius at least r, where $r \geq 1/2^{\text{poly(input size)}}$.

Lemma For any axis-aligned axis-aligned half-ellipsoid $\mathcal{R}_{c,a,w}$, we can efficiently find an axis-aligned ellipsoid $\mathcal{Q}_{c',a'}$ such that

- $\mathcal{R}_{c,a,w} \subseteq \mathcal{Q}_{c',a'}$
- $\frac{\mathrm{vol}(\mathcal{Q}_{c',a'})}{\mathrm{vol}(\mathcal{Q}_{c,a})} \leq e^{-\frac{1}{2(n+1)}} = 1 - \Omega\left(\frac{1}{n}\right)$

Assumption

- The initial polytope is contained in a ball of radius R, where $R \leq 2^{\text{poly(input size)}}$.
- When the polytope is not empty, it contains a ball of radius at least r, where $r \geq 1/2^{\text{poly(input size)}}$.

- The first assumption is easy to guarantee; the second assumption needs some twisting.

Lemma For any axis-aligned axis-aligned half-ellipsoid $\mathcal{R}_{c,a,w}$, we can efficiently find an axis-aligned ellipsoid $\mathcal{Q}_{c',a'}$ such that

- $\mathcal{R}_{c,a,w} \subseteq \mathcal{Q}_{c',a'}$
- $\frac{\mathrm{vol}(\mathcal{Q}_{c',a'})}{\mathrm{vol}(\mathcal{Q}_{c,a})} \leq e^{-\frac{1}{2(n+1)}} = 1 - \Omega\left(\frac{1}{n}\right)$

Assumption

- The initial polytope is contained in a ball of radius R, where $R \leq 2^{\text{poly(input size)}}$.
- When the polytope is not empty, it contains a ball of radius at least r, where $r \geq 1/2^{\text{poly(input size)}}$.

- The first assumption is easy to guarantee; the second assumption needs some twisting.
- we can stop the ellipsoid algorithm when volume is less than the volume of a ball of radius r

Lemma For any axis-aligned axis-aligned half-ellipsoid $\mathcal{R}_{c,a,w}$, we can efficiently find an axis-aligned ellipsoid $\mathcal{Q}_{c',a'}$ such that

- $\mathcal{R}_{c,a,w} \subseteq \mathcal{Q}_{c',a'}$
- $\frac{\text{vol}(\mathcal{Q}_{c',a'})}{\text{vol}(\mathcal{Q}_{c,a})} \leq e^{-\frac{1}{2(n+1)}} = 1 - \Omega\left(\frac{1}{n}\right)$

Assumption

- The initial polytope is contained in a ball of radius R, where $R \leq 2^{\text{poly(input size)}}$.
- When the polytope is not empty, it contains a ball of radius at least r, where $r \geq 1/2^{\text{poly(input size)}}$.

- The first assumption is easy to guarantee; the second assumption needs some twisting.
- we can stop the ellipsoid algorithm when volume is less than the volume of a ball of radius r

- $R \leq 2^{\mathsf{poly(input\ size)}}, \qquad r \geq 1/2^{\mathsf{poly(input\ size)}}$

- $R \leq 2^{\text{poly(input size)}}, \qquad r \geq 1/2^{\text{poly(input size)}}$
- number of iterations for ellipsoid method is at most

$$\ln_{e^{\frac{1}{2(n+1)}}} \left(\frac{R}{r}\right)^n = n \cdot \frac{\ln(R/r)}{1/(2(n+1))} = O(n^2) \cdot \ln \frac{R}{r}$$
$$\leq O(n^2) \cdot \text{poly(input size)}$$