Lecture 15: Semidefinite Programming, Duality & SDP Relaxations

Rafael Oliveira

University of Waterloo
Cheriton School of Computer Science

rafael.oliveira.teaching@gmail.com

November 2, 2021

Overview

- Duality Theory
- Why Relax & Round?
- Conclusion
- Acknowledgements

Working with Symmetric Matrices

Definition (Frobenius Inner Product)

$A, B \in \mathcal{S}^m$, define the *Frobenius inner product* as

symmetric m x m matrices

$$\langle A, B \rangle := \mathrm{tr}[AB] = \sum_{i,j} A_{ij} B_{ij}$$

- This is the "usual inner product" if you think of the matrices as vectors
- Thus, have the norm

$$\|A\|_F = \sqrt{\langle A, A \rangle} = \sqrt{\sum_{i,j} A_{ij}^2}$$

- With this norm, can talk about the *polar dual* to a given spectrahedron $S \subseteq \mathcal{S}^m$:

$$S^\circ = \{ Y \in \mathcal{S}^m \mid \langle Y, X \rangle \leq 1, \ \forall X \in S \}$$

Standard Primal Form

Just like in Linear Programming, we can represent SDPs in standard form:

$C, A_i \in S^{m \times m}$

$$
\begin{aligned}
\text{minimize} \quad & \langle C, X \rangle \\
\text{subject to} \quad & \langle A_i, X \rangle = b_i \\
& X \succeq 0
\end{aligned}
$$

← linear function of entries of X

← linear constraints

← PSD constraint

Where now:

for LP standard form

$$
\begin{aligned}
\text{minimize} \quad & c^T y \\
\text{s.t.} \quad & Ay = b \iff \langle A_i, y \rangle = b_i \\
& y \geq 0
\end{aligned}
$$

$$A = \begin{pmatrix} - A_1 - \\ - A_2 - \\ \vdots \\ - A_m - \end{pmatrix}$$

Standard Primal Form

Just like in Linear Programming, we can represent SDPs in standard form:

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{aligned}$$

Where now:

- the variables are encoded in a positive semidefinite matrix X,

Standard Primal Form

Just like in Linear Programming, we can represent SDPs in standard form:

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{aligned}$$

Where now:

- the variables are encoded in a positive semidefinite matrix X,
- each constraint is given by an inner product $\langle A_i, X \rangle = b_i$

Standard Primal Form

Just like in Linear Programming, we can represent SDPs in standard form:

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{aligned}$$

Where now:

- the variables are encoded in a positive semidefinite matrix X,
- each constraint is given by an inner product $\langle A_i, X \rangle = b_i$
- Note the similarity with LP standard primal. Can obtain LP standard form by making X and A_i's to be diagonal

Standard Primal Form

Just like in Linear Programming, we can represent SDPs in standard form:

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{aligned}$$

should be a spectrahedron (∴ encoded by LMIs)

Where now:

- the variables are encoded in a positive semidefinite matrix X,
- each constraint is given by an inner product $\langle A_i, X \rangle = b_i$
- Note the similarity with LP standard primal. Can obtain LP standard form by making X and A_i's to be diagonal
- How is that an LMI though?

Standard Primal Form as LMI

$$
\begin{aligned}
\text{minimize} \quad & \langle C, X \rangle \\
\text{subject to} \quad & \langle A_i, X \rangle = b_i \\
& X \succeq 0
\end{aligned}
$$

t constraints of this form

LMI: $M_0 + \sum_{i=1}^{n} M_i x_i \succeq 0$

$$\sum_{j,k=1}^{m} (A_i)_{jk} X_{jk} = b_i$$

$$B_{jk} = \begin{pmatrix} (A_1)_{jk} & & & \\ & (A_2)_{jk} & & \\ & & \ddots & \\ & & & (A_t)_{jk} \end{pmatrix}$$

$$M_{jk} = \left(\begin{array}{cc|c} B_{jk} & 0 & 0 \\ 0 & -B_{jk} & 0 \\ \hline 0 & 0 & E_{jk} \end{array}\right)$$

E_{jk}: 1 in entry jk and 0 all other entries

$$M_0 = \left(\begin{array}{c|c} \begin{matrix} b_1 & & & & & \\ & \ddots & & & 0 & \\ & & b_t & & & \\ & & & -b_1 & & \\ & 0 & & & \ddots & \\ & & & & & -b_t \end{matrix} & 0 \\ \hline 0 & 0 \end{array}\right)$$

$$b_i = \langle A_i, X \rangle$$

$$M_0 + \sum_{jk=1}^{m} M_{jk} \cdot X_{jk} \succeq 0 \iff \boxed{\begin{aligned} b_i &\geq \langle A_i, X \rangle \\ -b_i &\geq -\langle A_i, X \rangle \end{aligned}}$$

$$\underbrace{0 + \sum x_{jk} \cdot E_{jk} \succeq 0}_{X \succeq 0}$$

Example

$$\begin{aligned} \text{minimize} \quad & 2x_{11} + 2x_{12} \\ \text{subject to} \quad & x_{11} + x_{22} = 1 \\ & \begin{pmatrix} x_{11} & x_{12} \\ x_{12} & x_{22} \end{pmatrix} \succeq 0 \end{aligned}$$

$\langle c, X \rangle$

in standard primal form

$$\Big\langle \underbrace{\begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix}}_{C}, \begin{pmatrix} x_{11} & x_{12} \\ x_{12} & x_{22} \end{pmatrix} \Big\rangle = 2x_{11} + 2x_{12}$$

$$\Big\langle \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{A_1}, \begin{pmatrix} x_{11} & x_{12} \\ x_{12} & x_{22} \end{pmatrix} \Big\rangle = 1$$

$$X \succeq 0$$

Semidefinite Programming Duality

Consider our SDP:

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{aligned}$$

Semidefinite Programming Duality

Consider our SDP:

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{aligned}$$

- If we look at what happens when we multiply i^{th} equality by a variable y_i:

$$\sum_{i=1}^{t} y_i \cdot \langle A_i, X \rangle = \sum_{i=1}^{t} y_i \cdot b_i \quad \Rightarrow \quad \left\langle \sum_{i=1}^{t} y_i A_i \ , \ X \right\rangle = y^T b$$

Semidefinite Programming Duality

Consider our SDP:

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{aligned}$$

- If we look at what happens when we multiply i^{th} equality by a variable y_i:

$$\sum_{i=1}^{t} y_i \cdot \langle A_i, X \rangle = \sum_{i=1}^{t} y_i \cdot b_i \quad \Rightarrow \quad \left\langle \sum_{i=1}^{t} y_i A_i \ , \ X \right\rangle = y^T b$$

- Thus, if $\sum_{i=1}^{t} y_i A_i \preceq C$, then we have:

$$y^T b = \left\langle \sum_{i=1}^{t} y_i A_i \ , \ X \right\rangle \leq \langle C, X \rangle$$

Property of inner product: if $X \succeq 0$ and $A \succeq B$ then $\langle A, X \rangle \geq \langle B, X \rangle$

Semidefinite Programming Duality

Consider our SDP:

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{aligned}$$

- If we look at what happens when we multiply i^{th} equality by a variable y_i:

$$\sum_{i=1}^{t} y_i \cdot \langle A_i, X \rangle = \sum_{i=1}^{t} y_i \cdot b_i \quad \Rightarrow \quad \left\langle \sum_{i=1}^{t} y_i A_i \; , \; X \right\rangle = y^T b$$

- Thus, if $\sum_{i=1}^{t} y_i A_i \preceq C$, then we have:

$$y^T b = \left\langle \sum_{i=1}^{t} y_i A_i \; , \; X \right\rangle \leq \langle C, X \rangle$$

- $y^T b$ is a *lower bound* on the solution to our SDP!

Semidefinite Programming Duality

Consider the following SDPs:

Primal SDP

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{aligned}$$

Dual SDP

$$\begin{aligned} \text{maximize} \quad & y^T b \\ \text{subject to} \quad & \sum_{i=1}^{t} y_i A_i \preceq C \end{aligned}$$

any solution of dual is lower bound on the primal!

Semidefinite Programming Duality

Consider the following SDPs:

Primal SDP

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{aligned}$$

Dual SDP

$$\begin{aligned} \text{maximize} \quad & y^T b \\ \text{subject to} \quad & \sum_{i=1}^{t} y_i A_i \preceq C \end{aligned}$$

- From previous slide

$$\sum_{i=1}^{t} y_i A_i \preceq C \Rightarrow y^T b \text{ is a lower bound on value of Primal}$$

Semidefinite Programming Duality

Consider the following SDPs:

Primal SDP

$$\begin{array}{ll} \text{minimize} & \langle C, X\rangle \\ \text{subject to} & \langle A_i, X\rangle = b_i \\ & X \succeq 0 \end{array}$$

Dual SDP

$$\begin{array}{ll} \text{maximize} & y^T b \\ \text{subject to} & \sum_{i=1}^{t} y_i A_i \preceq C \end{array}$$

- From previous slide

$$\sum_{i=1}^{t} y_i A_i \preceq C \Rightarrow y^T b \text{ is a lower bound on value of Primal}$$

- Thus, the optimal (maximum) value of *dual LP* lower bounds the optimal (minimum) value of the *Primal LP*!

Semidefinite Programming Duality

Consider the following SDPs:

Primal SDP

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{aligned}$$

Dual SDP

$$\begin{aligned} \text{maximize} \quad & y^T b \\ \text{subject to} \quad & \sum_{i=1}^{t} y_i A_i \preceq C \end{aligned}$$

- From previous slide

$$\sum_{i=1}^{t} y_i A_i \preceq C \Rightarrow y^T b \text{ is a lower bound on value of Primal}$$

- Thus, the optimal (maximum) value of *dual LP* lower bounds the optimal (minimum) value of the *Primal LP*!

Theorem (Weak Duality)

Let X be a feasible solution of the primal SDP and y be a feasible solution of the dual SDP. Then

$$y^T b \leq \langle C, X \rangle.$$

Remarks on Duality

Primal SDP		*Dual SDP*	
minimize	$\langle C, X \rangle$	maximize	$y^T b$
subject to	$\langle A_i, X \rangle = b_i$	subject to	$\sum_{i=1}^{t} y_i A_i \preceq C$
	$X \succeq 0$		

Remarks on Duality

Primal SDP

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{aligned}$$

Dual SDP

$$\begin{aligned} \text{maximize} \quad & y^T b \\ \text{subject to} \quad & \sum_{i=1}^{t} y_i A_i \preceq C \end{aligned}$$

Theorem (Complementary Slackness)

Let X be a feasible solution of the primal SDP and y be a feasible solution of the dual SDP. If (X, y) satisfy the complementary slackness condition

$$\left(C - \sum_{i=1}^{t} y_i A_i \right) X = 0$$

Then (X, y) are primal and dual optimum solutions of the SDP problem.

Remarks on Duality

Primal SDP

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{aligned}$$

Dual SDP

$$\begin{aligned} \text{maximize} \quad & y^T b \\ \text{subject to} \quad & \sum_{i=1}^{t} y_i A_i \preceq C \end{aligned}$$

Theorem (Complementary Slackness)

Let X be a feasible solution of the primal SDP and y be a feasible solution of the dual SDP. If (X, y) satisfy the complementary slackness condition

$$\left(C - \sum_{i=1}^{t} y_i A_i \right) X = 0$$

Then (X, y) are primal and dual optimum solutions of the SDP problem.

Complementary slackness gives us *sufficient* conditions to check optimality of our solutions.

Strong Duality

Primal SDP

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{aligned}$$

Dual SDP

$$\begin{aligned} \text{maximize} \quad & y^T b \\ \text{subject to} \quad & \sum_{i=1}^{t} y_i A_i \preceq C \end{aligned}$$

Strong Duality

Primal SDP

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{aligned}$$

Dual SDP

$$\begin{aligned} \text{maximize} \quad & y^T b \\ \text{subject to} \quad & \sum_{i=1}^{t} y_i A_i \preceq C \end{aligned}$$

- Strong duality in SDPs is a bit more complex than in LPs.

Strong Duality

Primal SDP		*Dual SDP*	
minimize	$\langle C, X \rangle$	maximize	$y^T b$
subject to	$\langle A_i, X \rangle = b_i$	subject to	$\sum_{i=1}^{t} y_i A_i \preceq C$
	$X \succeq 0$		

- Strong duality in SDPs is a bit more complex than in LPs.
- Both primal and dual may be feasible, and yet strong duality may not hold!

Strong Duality

Primal SDP	*Dual SDP*
$$\begin{array}{ll} \text{minimize} & \langle C, X\rangle \\ \text{subject to} & \langle A_i, X\rangle = b_i \\ & X \succeq 0 \end{array}$$	$$\begin{array}{ll} \text{maximize} & y^T b \\ \text{subject to} & \sum_{i=1}^{t} y_i A_i \preceq C \end{array}$$

- Strong duality in SDPs is a bit more complex than in LPs.
- Both primal and dual may be feasible, and yet strong duality may not hold!
- But under mild conditions, strong duality holds!

Strong Duality

Primal SDP

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{aligned}$$

Dual SDP

$$\begin{aligned} \text{maximize} \quad & y^T b \\ \text{subject to} \quad & \sum_{i=1}^{t} y_i A_i \preceq C \end{aligned}$$

- Strong duality in SDPs is a bit more complex than in LPs.
- Both primal and dual may be feasible, and yet strong duality may not hold!
- But under mild conditions, strong duality holds!
- Primal SDP is *strictly feasible* if there is feasible solution $X \succ 0$.
- Dual SDP is *strictly feasible* if there is feasible $\sum_{i=1}^{t} y_i A_i \prec C$.

positive definite

strict inequality

Strong Duality

Primal SDP

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{aligned}$$

Dual SDP

$$\begin{aligned} \text{maximize} \quad & y^T b \\ \text{subject to} \quad & \sum_{i=1}^{t} y_i A_i \preceq C \end{aligned}$$

- Strong duality in SDPs is a bit more complex than in LPs.
- Both primal and dual may be feasible, and yet strong duality may not hold!
- But under mild conditions, strong duality holds!
- Primal SDP is *strictly feasible* if there is feasible solution $X \succ 0$.
- Dual SDP is *strictly feasible* if there is feasible $\sum_{i=1}^{t} y_i A_i \prec C$.

Theorem (Strong Duality under Slater Conditions)

If primal SDP and dual SDP are both strictly feasible, then

↪ Slater conditions

$$\textit{max dual} = \textit{min of primal}.$$

- Duality Theory
- Why Relax & Round?
- Conclusion
- Acknowledgements

Motivation - NP-hard problems

- Many important problems are NP-hard to solve.
- What do we do when we see one?

Motivation - NP-hard problems

- Many important problems are NP-hard to solve.
- What do we do when we see one?
  1. Find approximate solutions in polynomial time!
  2. Sometimes we even do that for problems in P (but we want much much faster solutions)

Motivation - NP-hard problems

- Many important problems are NP-hard to solve.
- What do we do when we see one?
  1. Find approximate solutions in polynomial time!
  2. Sometimes we even do that for problems in P (but we want much much faster solutions)
- **Integer Linear Program (ILP):**

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & \boxed{x \in \mathbb{N}^n} \end{aligned}$$

Motivation - NP-hard problems

- Many important problems are NP-hard to solve.
- What do we do when we see one?
  1. Find approximate solutions in polynomial time!
  2. Sometimes we even do that for problems in P (but we want much much faster solutions)
- **Integer Linear Program (ILP):**

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax \le b \\ & x \in \mathbb{N}^n \end{aligned}$$

- Advantage of ILPs: very expressive language to formulate optimization problems (capture many combinatorial optimization problems)
- Disadvantage of ILPs: capture even NP-hard problems (thus NP-hard)
- But we know how to solve LPs. Can we get partial credit in life?

Motivation - NP-hard problems

- **Quadratic Program (QP):**

$$\begin{aligned} \text{minimize} \quad & g(x) \\ \text{subject to} \quad & q_i(x) \geq 0 \end{aligned}$$

where each $q_i(x)$ and $g(x)$ are quadratic functions on x.

Motivation - NP-hard problems

- **Quadratic Program (QP):**

$$\begin{aligned} \text{minimize} \quad & g(x) \\ \text{subject to} \quad & q_i(x) \geq 0 \end{aligned}$$

  where each $q_i(x)$ and $g(x)$ are quadratic functions on x.
- Advantage of QPs: very expressive language to formulate optimization problems

Motivation - NP-hard problems

- **Quadratic Program (QP):**

$$\begin{aligned} \text{minimize} \quad & g(x) \\ \text{subject to} \quad & q_i(x) \geq 0 \end{aligned}$$

  where each $q_i(x)$ and $g(x)$ are quadratic functions on x.
- Advantage of QPs: very expressive language to formulate optimization problems
- Disadvantage of QPs: capture even NP-hard problems (ILPs for instance)

0-1 valued

$x \in \{0,1\} \iff x(x-1) = 0$ quadratic constraint

Motivation - NP-hard problems

- **Quadratic Program (QP):**

$$\begin{aligned} \text{minimize} \quad & g(x) \\ \text{subject to} \quad & q_i(x) \geq 0 \end{aligned}$$

  where each $q_i(x)$ and $g(x)$ are quadratic functions on x.
- Advantage of QPs: very expressive language to formulate optimization problems
- Disadvantage of QPs: capture even NP-hard problems (ILPs for instance)
- Can relax quadratic programs with SDPs

Motivation - NP-hard problems

- **Quadratic Program (QP):**

$$\begin{aligned} \text{minimize} \quad & g(x) \\ \text{subject to} \quad & q_i(x) \geq 0 \end{aligned}$$

  where each $q_i(x)$ and $g(x)$ are quadratic functions on x.
- Advantage of QPs: very expressive language to formulate optimization problems
- Disadvantage of QPs: capture even NP-hard problems (ILPs for instance)
- Can relax quadratic programs with SDPs
    - Can we get **better** approximations using SDPs instead of ILPs?

Motivation - NP-hard problems

- **Quadratic Program (QP):**

$$\begin{aligned} \text{minimize} \quad & g(x) \\ \text{subject to} \quad & q_i(x) \geq 0 \end{aligned}$$

  where each $q_i(x)$ and $g(x)$ are quadratic functions on x.
- Advantage of QPs: very expressive language to formulate optimization problems
- Disadvantage of QPs: capture even NP-hard problems (ILPs for instance)
- Can relax quadratic programs with SDPs
  - Can we get **better** approximations using SDPs instead of ILPs?
- Yes. Today and next lecture we will see Max-Cut (more generally constraint satisfaction relaxations)

Motivation - NP-hard problems

- **Quadratic Program (QP):**

$$\begin{aligned} \text{minimize} \quad & g(x) \\ \text{subject to} \quad & q_i(x) \geq 0 \end{aligned}$$

  where each $q_i(x)$ and $g(x)$ are quadratic functions on x.
- Advantage of QPs: very expressive language to formulate optimization problems
- Disadvantage of QPs: capture even NP-hard problems (ILPs for instance)
- Can relax quadratic programs with SDPs
  - Can we get **better** approximations using SDPs instead of ILPs?
- Yes. Today and next lecture we will see Max-Cut (more generally constraint satisfaction relaxations)
- Very impressive recent theoretical developments! Unique Games Conjecture, Sum-of-Squares, and more!

Example

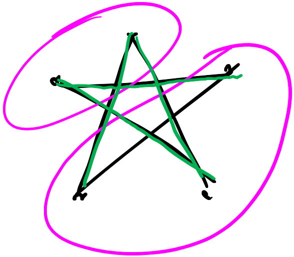
Maximum Cut (Max-Cut):

$$G(V, E) \text{ graph.}$$

$$\text{Cut } S \subseteq V \text{ and size of cut is}$$

$$|E(S, \overline{S})| = |\{(u, v) \in E \mid u \in S, v \notin S\}|.$$

Goal: find cut of maximum size.



size of cut is 4

Example

Maximum Cut (Max-Cut):

$u \in S$, $v \in \bar{S}$ $\Rightarrow$ $x_u + x_v = 1$, $2 - x_u - x_v = 1$ $\Rightarrow$ $z_e = 1$

$G(V, E)$ graph.

Cut $S \subseteq V$ and size of cut is

$$|E(S, \overline{S})| = |\{(u, v) \in E \mid u \in S, v \notin S\}|.$$

Goal: find cut of maximum size.

Integer Linear Program:

want to say that $z_e = \begin{cases} 1 & \text{if } e \in E(S, \bar{S}) \\ 0 & \text{otherwise} \end{cases}$

$$\text{maximize} \quad \sum_{e \in E} z_e$$

$$\text{subject to} \quad x_u + x_v \geq z_e \quad \text{for } e = \{u, v\} \in E$$

(if $u, v \in \bar{S}$ don't count e)

$$2 - x_u - x_v \geq z_e \quad \text{for } e = \{u, v\} \in E$$

(if $u, v \in S$ don't count e)

$$x_v \in \{0, 1\} \quad \text{for } v \in V$$

$x = \begin{cases} 1 & \text{if } v \in S \\ 0 & \text{otherwise} \end{cases}$

Example - Weighted Variant

Maximum Cut (Max-Cut):

$G(V, E, w)$ weighted graph. $\sum_{e \in E} w_e = 1$

Cut $S \subseteq V$ and weight of cut is the sum of weights of edges crossing cut.
Goal: find cut of maximum weight.

Integer Linear Program:

1 if include edge
0 otherwise

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} z_e \cdot w_e \\ \text{subject to} \quad & x_u + x_v \geq z_e \quad \text{for } e = \{u, v\} \in E \\ & 2 - x_u - x_v \geq z_e \quad \text{for } e = \{u, v\} \in E \\ & x_v \in \{0, 1\} \quad \text{for } v \in V \end{aligned}$$

Relax... & Round!

In our quest to get efficient (exact or approximate) algorithms for problems of interest, the following strategy is very useful:

[1] Even more general mathematical program, so long as derive SDP from it.

Relax... & Round!

In our quest to get efficient (exact or approximate) algorithms for problems of interest, the following strategy is very useful:

1. Formulate optimization problem as QP[1]

(instead of ILP)

[1] Even more general mathematical program, so long as derive SDP from it.

Relax... & Round!

In our quest to get efficient (exact or approximate) algorithms for problems of interest, the following strategy is very useful:

1. Formulate optimization problem as QP[1]
2. Derive SDP from the QP by going to higher dimensions and imposing PSD constraint

This is called an *SDP relaxation*.

(instead of dropping integrality constraint)

[1] Even more general mathematical program, so long as derive SDP from it.

Relax... & Round!

In our quest to get efficient (exact or approximate) algorithms for problems of interest, the following strategy is very useful:

1. Formulate optimization problem as QP[1]
2. Derive SDP from the QP by going to higher dimensions and imposing PSD constraint

   This is called an *SDP relaxation*.
3. We are still maximizing the same objective function, but over a (potentially) larger set of solutions.

$$OPT(SDP) \geq OPT(QP)$$

(analogous to OPT(LP) ≥ OPT(ILP))

[1] Even more general mathematical program, so long as derive SDP from it.

Relax... & Round!

In our quest to get efficient (exact or approximate) algorithms for problems of interest, the following strategy is very useful:

1. Formulate optimization problem as QP[1]
2. Derive SDP from the QP by going to higher dimensions and imposing PSD constraint

   This is called an *SDP relaxation*.
3. We are still maximizing the same objective function, but over a (potentially) larger set of solutions.

   $$OPT(SDP) \geq OPT(QP)$$
4. Solve SDP (approximately) optimally using efficient algorithm.

[1] Even more general mathematical program, so long as derive SDP from it.

Relax... & Round!

In our quest to get efficient (exact or approximate) algorithms for problems of interest, the following strategy is very useful:

1. Formulate optimization problem as QP[1]
2. Derive SDP from the QP by going to higher dimensions and imposing PSD constraint

   This is called an *SDP relaxation*.
3. We are still maximizing the same objective function, but over a (potentially) larger set of solutions.

   $$OPT(SDP) \geq OPT(QP)$$
4. Solve SDP (approximately) optimally using efficient algorithm.
   1. If solution to SDP is *integral* and *one-dimensional*, then it is a solution to QP and we are done

[1] Even more general mathematical program, so long as derive SDP from it.

Relax... & Round!

In our quest to get efficient (exact or approximate) algorithms for problems of interest, the following strategy is very useful:

1. Formulate optimization problem as QP[1]
2. Derive SDP from the QP by going to higher dimensions and imposing PSD constraint

   This is called an *SDP relaxation*.
3. We are still maximizing the same objective function, but over a (potentially) larger set of solutions.

   $$OPT(SDP) \geq OPT(QP)$$
4. Solve SDP (approximately) optimally using efficient algorithm.
   1. If solution to SDP is *integral* and *one-dimensional*, then it is a solution to QP and we are done
   2. If solution has *higher dimension*, then we have to devise *rounding procedure* that transforms

      high dimensional solutions $\rightarrow$ integral & 1D solutions

      rounded SDP solution value $\geq c \cdot OPT(QP)$

[1] Even more general mathematical program, so long as derive SDP from it.

Analyzing ILP for Max-Cut

$G(V, E, w)$ weighted graph. $\sum_{e \in E} w_e = 1$

Integer Linear Program:

$$
\begin{aligned}
\text{maximize} \quad & \sum_{e \in E} z_e \cdot w_e \\
\text{subject to} \quad x_u + x_v &\geq z_e \quad \text{for } e = \{u, v\} \in E \\
2 - x_u - x_v &\geq z_e \quad \text{for } e = \{u, v\} \in E \\
x_v &\in \{0, 1\} \quad \text{for } v \in V
\end{aligned}
$$

Analyzing ILP for Max-Cut

$G(V, E, w)$ weighted graph. $\sum_{e \in E} w_e = 1$

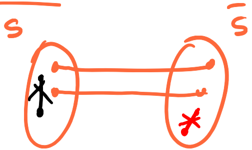
$w_e > 0$

Integer Linear Program:

$$\text{maximize} \quad \sum_{e \in E} z_e \cdot w_e$$

$$\begin{aligned} \text{subject to} \quad x_u + x_v &\geq z_e \quad \text{for } e = \{u, v\} \in E \\ 2 - x_u - x_v &\geq z_e \quad \text{for } e = \{u, v\} \in E \\ x_v &\in \{0, 1\} \quad \text{for } v \in V \end{aligned}$$

- $OPT(ILP) = 1 \Leftrightarrow G$ is bipartite



if an edge $E(S,S)$ or $E(\bar{S},\bar{S})$ then an cut has value < 1

Analyzing ILP for Max-Cut

$G(V, E, w)$ weighted graph. $\sum_{e \in E} w_e = 1$

Integer Linear Program:

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} z_e \cdot w_e \\ \text{subject to} \quad & x_u + x_v \geq z_e \quad \text{for } e = \{u, v\} \in E \\ & 2 - x_u - x_v \geq z_e \quad \text{for } e = \{u, v\} \in E \\ & x_v \in \{0, 1\} \quad \text{for } v \in V \end{aligned}$$

- $OPT(ILP) = 1 \Leftrightarrow G$ is bipartite
- $OPT(ILP) \geq 1/2$ (next slide)

Analyzing ILP for Max-Cut

$$G(V,E,w) \text{ weighted graph. } \sum_{e\in E} w_e = 1$$

Integer Linear Program:

$$\begin{aligned} \text{maximize} \quad & \sum_{e\in E} z_e \cdot w_e \\ \text{subject to} \quad & x_u + x_v \geq z_e \quad \text{for } e=\{u,v\}\in E \\ & 2 - x_u - x_v \geq z_e \quad \text{for } e=\{u,v\}\in E \\ & x_v \in \{0,1\} \quad \text{for } v\in V \end{aligned}$$

- $OPT(ILP) = 1 \Leftrightarrow G$ is bipartite
- $OPT(ILP) \geq 1/2$
- G complete graph $\Rightarrow OPT = \frac{1}{2} + \frac{1}{2(n-1)}$

Analyzing ILP for Max-Cut

$G(V, E, w)$ weighted graph. $\sum_{e \in E} w_e = 1$

Integer Linear Program:

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} z_e \cdot w_e \\ \text{subject to} \quad x_u + x_v &\geq z_e \quad \text{for } e = \{u, v\} \in E \\ 2 - x_u - x_v &\geq z_e \quad \text{for } e = \{u, v\} \in E \\ x_v &\in \{0, 1\} \quad \text{for } v \in V \end{aligned}$$

- $OPT(ILP) = 1 \Leftrightarrow G$ is bipartite
- $OPT(ILP) \geq 1/2$
- G complete graph $\Rightarrow OPT = \frac{1}{2} + \frac{1}{2(n-1)}$
- Max-Cut NP-hard

Proof that $OPT(ILP) \geq 1/2$

Averaging / Probabilistic method

pick a cut uniformly at random : for each vertex $v \in V$ make $x_v = \begin{cases} 1 & \text{w.p. } 1/2 \\ 0 & \text{w.p. } 1/2 \end{cases}$

$$\mathbb{E}[\text{value of cut}] = \mathbb{E}\left[\sum_{e \in E} w_e z_e\right] = \sum_{e \in E} w_e \cdot \mathbb{E}[z_e]$$

$$z_e = \begin{cases} 1 & \text{if } x_u \neq x_v \quad \text{w.p. } 1/2 \\ 0 & \text{if } x_u = x_v \quad \text{w.p. } 1/2 \end{cases} \qquad \mathbb{E}[z_e] = \frac{1}{2}$$

$e = \{u, v\}$

$$\therefore \mathbb{E}[\text{value of cut}] = \frac{1}{2} \cdot \sum w_e = 1/2$$

$$\| \quad \frac{1}{2^n} \cdot \sum_{S, \bar{S}} |E(S, \bar{S})|$$

$\Rightarrow \exists$ cut $(S, \bar{S})$ s.t. $|E(S, \bar{S})| \geq 1/2$

Rounding Max-Cut ILP

$G(V, E, w)$ weighted graph. $\sum_{e \in E} w_e = 1$

Linear Program Relaxation:

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} z_e \cdot w_e \\ \text{subject to} \quad & x_u + x_v \geq z_e \quad \text{for } e = \{u, v\} \in E \\ & 2 - x_u - x_v \geq z_e \quad \text{for } e = \{u, v\} \in E \\ & 0 \leq x_v \leq 1 \quad \text{for } v \in V \\ & 0 \leq z_e \leq 1 \quad \text{for } e \in E \end{aligned}$$

ILP
$x_v \in \{0,1\}$
$z_e \in \{0,1\}$

Rounding Max-Cut ILP

$G(V, E, w)$ weighted graph. $\sum_{e \in E} w_e = 1$

Linear Program Relaxation:

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} z_e \cdot w_e \quad \le \sum w_e = 1 \\ \text{subject to} \quad & x_u + x_v \ge z_e \quad \text{for } e = \{u, v\} \in E \\ & 2 - x_u - x_v \ge z_e \quad \text{for } e = \{u, v\} \in E \\ & 0 \le x_v \le 1 \quad \text{for } v \in V \\ & 0 \le z_e \le 1 \quad \text{for } e \in E \end{aligned}$$

- Setting $x_v = 1/2, \; z_e = 1$ we get $OPT(LP)$ always $= 1$

feasible solution!

Rounding Max-Cut ILP

$G(V, E, w)$ weighted graph. $\sum_{e \in E} w_e = 1$

Linear Program Relaxation:

$$
\begin{aligned}
\text{maximize} \quad & \sum_{e \in E} z_e \cdot w_e \\
\text{subject to} \quad & x_u + x_v \geq z_e \quad \text{for } e = \{u, v\} \in E \\
& 2 - x_u - x_v \geq z_e \quad \text{for } e = \{u, v\} \in E \\
& 0 \leq x_v \leq 1 \quad \text{for } v \in V \\
& 0 \leq z_e \leq 1 \quad \text{for } e \in E
\end{aligned}
$$

- Setting $x_v = 1/2$, $z_e = 1$ we get $OPT(LP)$ always $= 1$
- This relaxation is not helpful! :(

Max-Cut

$G(V, E, w)$ weighted graph. $\sum_{e \in E} w_e = 1$

Quadratic Program:

$$\text{maximize} \quad \sum_{\{u,v\} \in E} \frac{1}{2} \cdot w_{u,v} \cdot (1 - x_u x_v)$$

$$\text{subject to} \quad x_v^2 = 1 \quad \text{for } v \in V$$

$$x_v = \begin{cases} 1 & \text{if } v \in S \\ -1 & \text{if } v \in \bar{S} \end{cases} \iff x_v^2 = 1$$

$$\frac{1}{2}(1 - x_u x_v) = \begin{cases} 0 & \text{if } x_u = x_v \\ 1 & \text{if } x_u \neq x_v \text{ and } x_u \in \{-1,1\} \end{cases} \iff x_u x_v = -1$$

SDP Relaxation [Delorme, Poljak 1993]

$G(V, E, w)$ weighted graph, $|V| = n$ and $\sum_{e \in E} w_e = 1$

Semidefinite Program:

$$\text{maximize} \quad \sum_{\{u,v\} \in E} \frac{1}{2} \cdot w_{u,v} \cdot \left(1 - y_u^T y_v\right)$$

$$\text{subject to} \quad \|y_v\|_2^2 = 1 \quad \text{for } v \in V$$

$$y_v \in \mathbb{R}^d \quad \text{for } v \in V$$

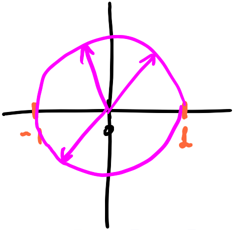
some dimension $d \geq 1$

replace $x_v \in \{-1, 1\}$ by a vector $y_v \in \mathbb{R}^d$

s.t. $\|y_v\|_2^2 = 1$

(if $d = 1$ then y_v becomes x_v)

$x_u x_v$ become $\langle y_u, y_v \rangle = y_u^T y_v$



SDP Relaxation [Delorme, Poljak 1993]

$G(V, E, w)$ weighted graph, $|V| = n$ and $\sum_{e \in E} w_e = 1$

$V = \{1, 2, \cdots, n\}$

Semidefinite Program:

$$\begin{aligned} \text{maximize} \quad & \sum_{\{u,v\} \in E} \frac{1}{2} \cdot w_{u,v} \cdot \left(1 - y_u^T y_v\right) \\ \text{subject to} \quad & \|y_v\|_2^2 = 1 \quad \text{for } v \in V \\ & y_v \in \mathbb{R}^d \quad \text{for } v \in V \end{aligned}$$

- How is that an SDP?

$X = Y^T Y$ $n \times n$ matrix

$Y = \begin{pmatrix} | & | & & | \\ y_1 & y_2 & \cdots & y_n \\ | & | & & | \end{pmatrix}$

we know that $X \succeq 0$

$X_{ii} = y_i^T y_i = \|y_i\|^2$

$X_{ii} = 1 \quad \forall i \in V$

Showing it is SDP

$y_u^T y_v$

$$X_{uv} = (Y^T Y)_{uv} = y_u^T y_v$$

thus, the optimization problem above can be written as

$$\text{maximize} \sum_{\{i,j\} \in E} \frac{1}{2} w_{ij} (1 - x_{ij})$$

$$\text{s.t.} \quad x_{ii} = 1 \quad \forall i \in [n]$$

$$X \succcurlyeq 0$$

← an SDP

Showing it is SDP

Conclusion

- Mathematical programming - very general, and pervasive in (combinatorial) algorithmic life
- Mathematical Programming hard in general
- Sometimes can get SDP rounding!

  Next lecture Max-Cut SDP rounding.
- Solve SDP and round the solution
  - Deterministic rounding when solutions are nice
  - Randomized rounding when things a bit more complicated

Acknowledgement

- Lecture based largely on:
  - Lecture 14 of Anupam Gupta and Ryan O'Donnell's Optimization class

    `https://www.cs.cmu.edu/~anupamg/adv-approx/`
- See their notes at

  `https://www.cs.cmu.edu/~anupamg/adv-approx/lecture14.pdf`

References I

Delorme, Charles, and Svatopluk Poljak (1993)
Laplacian eigenvalues and the maximum cut problem.
Mathematical Programming 62.1-3 (1993): 557-574.

Goemans, Michel and Williamson, David 1994
0.879-approximation algorithms for Max Cut and Max 2SAT.
Proceedings of the twenty-sixth annual ACM symposium on Theory of computing. 1994