CSE 421

LP Duality

Shayan Oveis Gharan
Intro to Duality

\[
\begin{align*}
\text{max} & \quad x_1 + 2x_2 \\
\text{s.t.,} & \quad x_1 + 3x_2 \leq 2 \\
& \quad 2x_1 + 2x_2 \leq 3 \\
& \quad x_1, x_2 \geq 0
\end{align*}
\]

Optimum solution: \( x_1 = \frac{5}{4} \) and \( x_2 = \frac{1}{4} \) with value \( x_1 + 2x_2 = \frac{7}{4} \)

How can you prove an upper-bound on the optimum?

**First attempt:** Since \( x_1, x_2 \geq 0 \)
\[
x_1 + 2x_2 \leq x_1 + 3x_2 \leq 2
\]

**Second attempt:**
\[
x_1 + 2x_2 \leq \frac{2}{3} (x_1 + 3x_2) + \frac{1}{3} (2x_1 + 2x_2) \leq \frac{2}{3} (2) + \frac{1}{3} (3) = \frac{7}{3}
\]

**Third attempt:**
\[
x_1 + 2x_2 \leq \frac{1}{2} (x_1 + 3x_2) + \frac{1}{4} (2x_1 + 2x_2) \leq \frac{1}{2} (2) + \frac{1}{4} (3) = \frac{7}{4}
\]
Dual Certificate

\[
\begin{align*}
\text{max} & \quad x_1 + 2x_2 \\
\text{s.t.,} & \quad x_1 + 3x_2 \leq 2 \quad y_1 \\
& \quad 2x_1 + 2x_2 \leq 3 \quad y_2 \\
& \quad x_1, x_2 \geq 0 
\end{align*}
\]

**Goal:** Minimize \(2y_1 + 3y_2\)

But, we must make sure the sum of the LHS is least the objective, i.e.,
\[
x_1 + 2x_2 \leq y_1 (x_1 + 3x_2) + y_2 (2x_1 + 2x_2)
\]

In other words,
\[
1 \leq 1 \cdot y_1 + 2 \cdot y_2 \\
2 \leq 3 \cdot y_1 + 2 \cdot y_2
\]

Finally, \(y_1, y_2 \geq 0\) (else the direction of inequalities change)
Dual Program

\[
\begin{align*}
\text{max} & \quad x_1 + 2x_2 \\
\text{s.t.} & \quad x_1 + 3x_2 \leq 2 \\
& \quad 2x_1 + 2x_2 \leq 3 \\
& \quad x_1, x_2 \geq 0
\end{align*}
\]

OPT: \( x_1 = 5/4 \) and \( x_2 = 1/4 \)
Value 7/4

\[
\begin{align*}
\text{min} & \quad 2y_1 + 3y_2 \\
\text{s.t.} & \quad y_1 + 2y_2 \geq 1 \\
& \quad 3y_1 + 2y_2 \geq 2 \\
& \quad y_1, y_2 \geq 0
\end{align*}
\]

OPT: \( y_1 = 1/2 \) and \( y_2 = 1/4 \)
Value 7/4
**Dual of Standard LP**

\[
\begin{align*}
\text{max} & \quad \langle c, x \rangle \\
\text{s.t.} & \quad \langle a_1, x \rangle \leq b_1 \quad y_1 \\
& \quad \langle a_2, x \rangle \leq b_2 \quad y_2 \\
& \quad \vdots \quad \vdots \\
& \quad \langle a_m, x \rangle \leq b_m \quad y_m \\
x_1, \ldots, x_n \geq 0
\end{align*}
\]

\[
\begin{align*}
\text{min} & \quad \langle b, y \rangle \\
\text{s.t.} & \quad a_{1,1}y_1 + \cdots + a_{m,1}y_m \geq c_1 \\
& \quad a_{1,2}y_1 + \cdots + a_{m,2}y_m \geq c_2 \\
& \quad \vdots \\
& \quad a_{1,n}y_1 + \cdots + a_{m,n}y_m \geq c_n \\
y_1, \ldots, y_m \geq 0
\end{align*}
\]

**Primal**

\[
\begin{align*}
\text{max} & \quad \langle c, x \rangle \\
\text{s.t.} & \quad Ax \leq b \\
x & \geq 0
\end{align*}
\]

**Dual**

\[
\begin{align*}
\text{min} & \quad \langle b, y \rangle \\
\text{s.t.} & \quad A^Ty \geq c \\
y & \geq 0
\end{align*}
\]
Facts About Linear Programs

**Lem**: Dual of Dual = Primal

**Thm** (weak duality): Every solution to the primal is at most every solution to the dual
\[ \langle c, x \rangle \leq \langle b, y \rangle \]

**Thm** (strong duality): If primal has a solution and dual has a solution then optimum of primal is equal to optimum of dual
Dual of Max-Flow

\[
\begin{align*}
\text{max} & \quad \sum_{e \text{ out of } s} x_e \\
\text{s.t.} & \quad \sum_{e \text{ out of } v} x_e = \sum_{e \text{ in to } v} x_e \quad \forall v \neq s, t \\
& \quad x_e \leq c(e) \quad \forall e \\
& \quad x_e \geq 0 \quad \forall e
\end{align*}
\]

\[
\begin{align*}
\text{min} & \quad \langle c, a \rangle \\
\text{s.t.} & \quad a_e + b_v \geq 1 \quad e = (s, v) \\
& \quad a_e - b_v \geq 0 \quad e = (v, t) \\
& \quad a_e + b_u - b_v \geq 0 \quad \text{other } e = (u, v) \\
& \quad a_e \geq 0 \quad \forall e
\end{align*}
\]
\[ \begin{align*}
\text{min} & \quad \langle c, a \rangle \\
\text{s. t., } & \quad a_e + b_v \geq 1 \quad e = (s, v) \\
& \quad a_e - b_v \geq 0 \quad e = (v, t) \\
& \quad a_e + b_u - b_v \geq 0 \quad \text{other } e = (u, v) \\
& \quad a_e \geq 0 \\
& \quad \forall e
\end{align*} \]

\[ \begin{align*}
\text{min} & \quad \langle c, a \rangle \\
\text{s. t., } & \quad a_e = \max(0, 1 - b_v) \quad e = (s, v) \\
& \quad a_e = \max(0, b_v) \quad e = (v, t) \\
& \quad a_e = \max(0, b_v - b_u) \quad \text{other } e = (u, v)
\end{align*} \]
\[ \min \quad \langle c, a \rangle \]
\[ \text{s. t.}, \quad a_e = \max(0, 1 - b_v) \quad e = (s, v) \]
\[ a_e = \max(0, b_v) \quad e = (v, t) \]
\[ a_e = \max(0, b_v - b_u) \quad \text{other } e = (u, v) \]

**Lem:** In OPT \( 0 \leq b_v \leq 1 \) for all \( v \)

**Pf:** If not, move up/down the value only decreases
\[
\min \quad \langle c, a \rangle \\
\text{s.t.,} \quad b_s = 1, b_t = 0 \\
0 \leq b_v \leq 1 \\
a_e = \max(0, b_v - b_u) \quad e = (u, v)
\]

**Lem:** In \( \text{OPT} \) \( 0 \leq b_v \leq 1 \) for all \( v \) 

**Pf:** If not, move up/down the value only decreases

**Lem:** In \( \text{OPT} \) \( b_v \in \{0,1\} \) for all \( v \) 
**Pf:** If not, choose a u.r. \( 0 \leq t \leq 1 \) 
If \( b_v \geq t \) set \( b_v = 1 \) else set \( b_v = 0 \). 
Then, the expected value of resulting solution sames as \( \text{OPT} \).
\[
\begin{align*}
\min & \quad \langle c, a \rangle \\
\text{s.t.,} & \quad b_s = 1, b_t = 0 \\
& \quad 0 \leq b_v \leq 1 \\
& \quad a_e = \max(0, b_v - b_u) \quad \text{other } e = (u, v)
\end{align*}
\]

**Lem:** In OPT \(0 \leq b_v \leq 1\) for all \(v\)

**Pf:** If not, move up/down the value only decreases

**Lem:** In OPT \(b_v \in \{0,1\}\) for all \(v\)

**Pf:** If not, choose a u.r. \(0 \leq t \leq 1\)

If \(b_v \geq t\) set \(b_v = 1\) else set \(b_v = 0\).

Then, the expected value of resulting solution same as OPT.
\[
\begin{align*}
\min & \quad \langle c, a \rangle \\
\text{s.t.,} & \quad b_s = 1, b_t = 0 \\
& \quad b_v \in \{0,1\} \\
& \quad a_e = \max(0, b_v - b_u) \quad \text{other } e = (u, v)
\end{align*}
\]

Min Cut!
Beyond LP: Convex Programming

A function $f: \mathbb{R} \rightarrow \mathbb{R}$ is convex if $f'' \geq 0$.

e.g., $f(x) = x^2$.

A function $f: \mathbb{R}^d \rightarrow \mathbb{R}$ is convex if $\nabla^2 f \succeq 0$

Convex Program

$$\begin{align*}
\min & \quad f(x) \\
\text{s.t.,} & \quad g_1(x) \leq b_1 \\
& \quad g_2(x) \leq b_2 \\
& \quad \quad \quad \vdots \\
& \quad g_m(x) \leq b_m
\end{align*}$$

$f$ and $g_1, \ldots, g_m$ must be convex. $\geq$ and $=$ are not allows!
Example

\[ \text{max } c_1 x_1 + c_2 x_2 \]
\[ \text{s. t. , } x_1^2 + x_2^2 \leq 1 \]
Summary (Linear Programming)

• Linear programming is one of the biggest advances in 20th century

• It is being used in many areas of science: Mechanics, Physics, Operations Research, and in CS: AI, Machine Learning, Theory, ...

• Almost all problems that we talked can be solved with LPs, Why not use LPs?
  • Combinatorial algorithms are typically faster
  • They exhibit a better understanding of worst case instances of a problem
  • They give certain structural properties, e.g., Integrality of Max-flow when capacities are integral

• There is rich theory of LP-duality which generalizes max-flow min-cut theorem
Reductions & NP-Completeness
**Goal**: Classify problems according to the amount of computational resources used by the best algorithms that solve them

Here we focus on time complexity

**Recall**: worst-case running time of an algorithm

- $\text{max}$ # steps algorithm takes on any input of size $n$
Computational Complexity and Reduction

In most cases, we cannot characterize the true hardness of a computational problem

So?

We only reduce the number of problems

Want to be able to make statements of the form

• "If we could solve problem B in polynomial time then we can solve problem A in polynomial time”
• “Problem B is at least as hard as problem A”
Polynomial Time Reduction

Def $A \leq_p B$: if there is an algorithm for problem $A$ using a ‘black box’ (subroutine) that solve problem $B$ s.t.,
- Algorithm uses only a polynomial number of steps
- Makes only a polynomial number of calls to a subroutine for $B$

So,

- B is Polynomial time solvable $\implies$ A is Polynomial time solvable

Conversely,

- No efficient Algorithm for $A$ $\implies$ No efficient Algorithm for $B$

In words, $B$ is as hard as $A$ (it can be even harder)
\( \leq_{p}^{1} \) Reductions

In this lecture we see a restricted form of polynomial-time reduction often called Karp or many-to-one reduction

\( A \leq_{p}^{1} B \): if and only if there is an algorithm for \( A \) given a black box solving \( B \) that on input \( x \)

- Runs for polynomial time computing an input \( f(x) \) of \( B \)
- Makes one call to the black box for \( B \) for input \( f(x) \)
- Returns the answer that the black box gave

We say that the function \( f(.) \) is the reduction
Decision Problems

A decision problem is a computational problem where the answer is just yes/no

Here, we study computational complexity of decision Problems.

Why?
- much simpler to deal with
- Decision version is not harder than Search version, so it is easier to lower bound Decision version
- Less important, usually, you can use decider multiple times to find an answer.
Indep Set: Given $G=(V,E)$ and an integer $k$, is there $S \subseteq V$ s.t. $|S| \geq k$ and no two vertices in $S$ are joined by an edge?

Clique: Given a graph $G=(V,E)$ and an integer $k$, is there $S \subseteq V$, $|U| \geq k$ s.t., every pair of vertices in $S$ is joined by an edge?

Claim: Indep Set $\leq_p$ Clique

Pf: Given $G = (V, E)$ and instance of indep Set. Construct a new graph $G' = (V, E')$ where $\{u, v\} \in E'$ if and only if $\{u, v\} \notin E$. 

Example 1: Indep Set $\leq_p$ Clique

S is an independent set in $G$

S is an Clique in $G'$
Example 2: Vertex Cover $\leq_p$ Indep Set

**Vertex Cover**: Given a graph $G=(V,E)$ and an integer $k$, is there a vertex cover of size at most $k$?

**Claim**: For any graph $G = (V, E)$, $S$ is an independent set iff $V - S$ is a vertex cover.

**Pf**:

$\Rightarrow$ Let $S$ be a independent set of $G$
Then, $S$ has **at most one** endpoint of every edge of $G$
So, $V - S$ has at least one endpoint of every edge of $G$
So, $V - S$ is a vertex cover.

$\Leftarrow$ Suppose $V - S$ is a vertex cover.
Then, there is no edge between vertices of $S$ (otherwise, $V - S$ is not a vertex cover)
So, $S$ is an independent set.
Example 3: Vertex Cover $\leq_p$ Set Cover

**Set Cover**: Given a set $U$, collection of subsets $S_1, \ldots, S_m$ of $U$ and an integer $k$, is there a collection of $k$ sets that contain all elements of $U$?

**Claim**: Vertex Cover $\leq_p$ Set Cover

**Pf:**
Given $(G = (V, E), k)$ of vertex cover we construct a set cover input $f(G, k)$
- $U = E$
- For each $v \in V$ we create a set $S_v$ of all edges connected to $v$

This clearly is a polynomial-time reduction

So, we need to prove it gives the right answer
**Example 3: Vertex Cover \( \leq_p \) Set Cover**

**Claim:** Vertex Cover \( \leq_p \) Set Cover

**Pf:** Given \( (G = (V, E), k) \) of vertex cover we construct a set cover input \( f(G, k) \)
- \( U = E \)
- For each \( v \in V \) we create a set \( S_v \) of all edges connected to \( v \)

Vertex-Cover \( (G, k) \) is yes \( \Rightarrow \) Set-Cover \( f(G, k) \) is yes
  - If a set \( W \subseteq V \) covers all edges, just choose \( S_v \) for all \( v \in W \), it covers all \( U \).

Set-Cover \( f(G, k) \) is yes \( \Rightarrow \) Vertex-Cover \( (G, k) \) is yes
  - If \( (S_{v_1}, ..., S_{v_k}) \) covers all \( U \), the set \( \{v_1, ..., v_k\} \) covers all edges of \( G \).