Duality.

\[ R = \max \min (x^t Ay) \]
\[ C = \min \max (x^t Ay) \]

Weak Duality: \( R \leq C \).

Proof: Better to go second.

Note:
- In some situation, \( R = 0 \).
- In some situation, \( C = 0 \).

Equilibrium =⇒ \( R = C \)!

Strong Duality: There is an equilibrium point! and \( R = C \)!

Doesn’t matter who plays first!

Summary and...

Zero sum game: \( m \times n \) matrix \( A \)

row minimizes. strategy: \( m \)-dimensional vector \( x \)

... probability distribution over rows.

column maximizes. strategy: \( m \)-dimensional vector \( x \)

... probability distribution over columns.

Payoff \( (x, y) = x^t Ay \).

Nash equilibrium \( (x^*, y^*) \):
neither player has better response against others.

If there is an equilibrium: no disadvantage in announcing strategy!

All equilibrium points all have same payoff.

Why? Equilibriums: \( x^* = y^* \).

Best row is worse under \( y^* \).

Best row is not equilibrium.

Example.

Row solution: \( Pr[p_1] = 1/2, Pr[p_2] = 1/3, Pr[p_3] = 1/6 \).

Edge solution: \( Pr[e_1] = 1/2, Pr[e_2] = 1/2 \).

Offense (Best Response):
- Catch me: route along shortest path.
- (Knows catcher’s distribution.)
- Catcher: raise toll on most congested edge.
- (Knows catch me’s distribution.)

Defense:
- Where should “catcher” play to catch any path? a cut.
- Minimum cut allows the maximum toll on any edge!
- What should “catch me” do when “catcher” plays?
- Max-Flow Problem.

Note: exponentially many strategies for “catch me”!

An “asymptotic” game.

“Catch me.”

Given: \( G = (V, E) \).

Row (“Catch me”): choose path from \( a \) to \( b \).

Column (“Catcher”): choose edge.

Row pays if column chooses edge on path.

Matrix:
row for each path: \( p \)

column for each edge: \( e \)

\[ A[p,e] = 1 \text{ if } e \in p. \]

Toll/Congestion

Given: \( G = (V, E) \).

Row: choose routing of all paths.

Column: choose edge.

Row pays if column chooses edge on any path.

Matrix:
row for each path: \( p \)

column for each edge: \( e \)

\[ A[p,e] = 1 \text{ if } e \in p. \]

Offense: (Best Response)
- Route: route along shortest paths.
- Toll: charge most loaded edge.

Defense: (Best Response)
- Toll: maximize shortest path under tolls.
- Route: minimize max loaded on any edge.

Again: exponential number of paths for route player.
You should now know about
Games
Nash Equilibrium
Pure Strategies
Zero Sum Two Person Games
Mixed Strategies.
Checking Equilibrium.
Best Response.
Statement of Duality Theorem.

Today
Maximum Weight Matching
Undergraduate: saw maximum matching! (hopefully.) Will review.

Matching.
Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to \mathbb{R}$, find a maximum weight matching.
A matching is a set of edges where no two share an endpoint.

Applications
Jobs to workers.
Teachers to classes.
Classes to classrooms.
“The assignment problem”
Min Weight Matching.
Negate values and find maximum weight matching.

Vertex Cover
Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to \mathbb{R}$, find an vertex cover function of minimum total value.
A function $p : V \to \mathbb{R}$, where for all edges, $e = (u, v)$,
$p(u) + p(v) \geq w(e)$.
Minimize $\sum_{u \in U} p(u)$.

Cover is upper bound.
Feasible $p(\cdot)$, for edge $e = (u, v)$, $p(u) + p(v) \geq w(e)$.

Holds with equality if for $e \in M$, $w(e) = p(u) + p(v)$ (Defn: tight edge.) and
perfect matching.
Simple example.

```
1  a  x  0
2  b  y  0
0  a  x  1
1  b  y  0
```

Matching and cover are optimal, edges in matching have\( w(e) = p(u) + p(v) \). Tight edge.

Proof of optimality.

```
1  a  x  0
2  b  y  0
0  a  x  1
1  b  y  0
```

Back to Maximum Weight Matching.

```
Simple example.

```
1  a  x  0
2  b  y  0
0  a  x  1
1  b  y  0
```

Maximum Matching

Given a bipartite graph, \( G = (U, V, E) \), find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

Example:

```
1  a  x
2  b  y
```

Start at unmatched node(s), follow unmatched edge(s), follow matched.
Repeat until an unmatched node.

```
1  a  x
2  b  y
```

No perfect matching

```
1  a  x
2  b  y
```

Can’t increase matching size.
No alternating path from (a) to (y).
Cut!
```
1  a  x
2  b  y
```

Cut!
```
1  a  x
2  b  y
```

Still Cut?
```
1  a  x
2  b  y
```

Use directed graph!
```
1  a  x
2  b  y
```

Cut in this graph.

Algorithm:
```
1  a  x
2  b  y
```

Given matching.
Direct unmatched edges \( U \) to \( V \), matched \( V \) to \( U \).
Find path between unmatched nodes on left to right. (BFS, DFS).
Until everything matched ... or output a cut.

Some details

```
1  a  x
2  b  y
```

Add 0 value edges, so that optimal solution contains perfect matching.
Beginning “Matcher” Solution: \( M = \{ \} \).
Feasible! Value = 0.
Beginning “Coverer” Solution:
```
1  a  x
2  b  y
```

\( p(u) = \) maximum incident edge for \( u \in U \), 0 otherwise.
Main Work:
```
1  a  x
2  b  y
```

Breadth first search from unmatched nodes finds cut.
Update prices (find minimum delta.)
Simple Implementation:
```
1  a  x
2  b  y
```

Each bfs either augments or adds node to \( S \) in next cut.
\( O(n) \) iterations per augmentation.
\( O(n) \) augmentations.
```
1  a  x
2  b  y
```

\( O(n^2 m) \) time.

Maximum Weight Matching

Goal: perfect matching on tight edges.

```
1  a  x
2  b  y
```

Algorithm
```
1  a  x
2  b  y
```

Init: empty matching, feasible cover function \( p(\cdot) \)
```
1  a  x
2  b  y
```

Add tight edges to matching.
```
1  a  x
2  b  y
```

Maximum matching algorithm.*
```
1  a  x
2  b  y
```

No augmenting path.
```
1  a  x
2  b  y
```

Cut, \( (S, T) \), in directed graph of tight edges!
```
1  a  x
2  b  y
```

All edges across cut are not tight. (loose?)
```
1  a  x
2  b  y
```

Non.tight edges leaving cut, go from \( S_U, T_V \).
```
1  a  x
2  b  y
```

Lower prices in \( S_U \), raise prices in \( S_V \), all explored edges still tight
```
1  a  x
2  b  y
```

... and get new tight edge!
```
1  a  x
2  b  y
```

What’s delta? \( w(e) < p(u) + p(v) \) →
```
1  a  x
2  b  y
```

\( \delta = \min_{e \in \{(S_U - T_V)|p(u) + p(v) - w(e)\}} \)
Example

Weight legend:
- Black 1, green 2, blue 3
- Tight edges for initial prices.
- Max matching in tight edges.
- Dashed means matched.
- No augmenting path.
- Terminate with perfect matching.
- Blue edge on right soon to be tight.
- Adjust prices.
- Adjacent edges.
- Rescale δ.
- Some more tight edges.
- Adjust prices.
- Some more tight edges.
- And another augmentation.
- ...and finally: a perfect matching.

All matched edges tight.
Perfect matching. Feasible price function. Values the same. Optimal!

Notice:
- no weights on the right problem.
- retain previous matching through price changes.
- retains edges in failed search through price changes.

Some thoughts..

Unweighted matching algorithm to weighted.
How?
Use duality.
In this case:
- Dual feasible.
- Primal infeasible.
- Primal only "plays" tight constraints. Best offense.
- Dual only plays tight constraints.
- Terminate when perfect matching.
- → Dual's best offense.
- Equilibrium.

...see you on Tuesday