Solution Techniques for Zero-sum Extensive-form Games

Kevin Waugh
kevin.waugh@gmail.com
Carnegie Mellon University
University of Alberta

AAAI 2017
Matrix Games

Definitions—What is a Nash Equilibrium?
Formulation as Minimax Problem
Formulation as Linear Program
Solutions via Subgradient Methods
Solutions via Smoothing and Gradient Methods
Solutions via Online Learning

Extensive-form Games

Tips and Tricks
What is a zero-sum matrix game?
(Osborne and Rubinstein 1994, Fudenburg and Tirole 1991)

- Defined by $A \in \mathbb{R}^{m \times n}$
- Two players—the row player and the column player

$$A = \begin{bmatrix} 2 & 0 & 0 \\ -1 & 3 & 1 \end{bmatrix}$$
How is it played?

- A matrix game is a **one-shot** game
- The row player selects a row \( i \in [m] \) and his opponent a column \( j \in [n] \)
- We call \((i, j)\) the game’s **outcome**

\[
A = \begin{bmatrix}
2 & 0 & 0 \\
-1 & 3 & 1 \\
\end{bmatrix}
\]

\((i, j) = (2, 1)\)
Why is it played?

- The row player receives utility
  \( u_x(i, j) = -a_{i,j} \)
- The column player gets
  \( u_y(i, j) = a_{i,j} \)
- The game is **zero-sum** since
  \( u_x(i, j) + u_y(i, j) = 0 \)

\[
A = \begin{bmatrix}
2 & 0 & 0 \\
-1 & 3 & 1
\end{bmatrix}
\]

\[
u_x(i, j) = -a_{2,1} = 1
\]
\[
u_y(i, j) = a_{2,1} = -1
\]
**Why is it played?**

- The row player receives utility
  \[ u_x(i, j) = -a_{i,j} \]
- The column player gets
  \[ u_y(i, j) = a_{i,j} \]
- The game is **zero-sum** since
  \[ u_x(i, j) + u_y(i, j) = 0 \]
- Let \( L = \max_{i,j} |a_{i,j}| \)

\[
A = \begin{bmatrix}
2 & 0 & 0 \\
-1 & 3 & 1
\end{bmatrix}
\]

\[
\begin{align*}
u_x(i, j) &= -a_{2,1} = 1 \\
u_y(i, j) &= a_{2,1} = -1
\end{align*}
\]
Strategies and Profiles

- A **strategy** for the row player is a probability distribution over the rows of $A$, $x \in \Delta_m = \{ x \mid \sum_{i=1}^{m} x_i = 1, x \geq 0 \}$

- A **strategy profile** is a pair of strategies, one for each player $(x, y) \in \Delta_m \times \Delta_n$

$$A = \begin{bmatrix} 2 & 0 & 0 \\ -1 & 3 & 1 \end{bmatrix}$$

$$x = \left( \frac{1}{3}, \frac{2}{3} \right)$$

$$y = \left( \frac{1}{2}, \frac{1}{4}, \frac{1}{4} \right)$$
The **expected utility** for the row player under profile \((x, y)\) is

\[
u_x(x, y) = \sum_{i=1}^{m} \sum_{j=1}^{n} -x_i y_j a_{i,j}
= -x' Ay
\]

\[
u_y(x, y) = x' Ay
\]

\[
A = \begin{bmatrix} 2 & 0 & 0 \\ -1 & 3 & 1 \end{bmatrix}
\]

\[
x = \left( \frac{1}{3}, \frac{2}{3} \right)
\]

\[
y = \left( \frac{1}{2}, \frac{1}{4}, \frac{1}{4} \right)
\]

\[
u_x(x, y) = -\frac{2}{3}
\]
Optimal play against a known opponent

A **best response** to the row player’s strategy \( x \) is a **pure strategy** that maximizes the column player’s utility:

\[
\text{brv}_y(x) = \max_{y \in [j]} u_y(x, y) := x' Ay
\]

Note:

\[
\text{brv}_y(x) \geq x'Ay, \forall y \in \Delta_n
\]
Nash equilibrium

A **Nash equilibrium** is a pair of mutual best responses:

\[
\begin{align*}
\text{brv}_x(y) &= u_x(x, y), \\
\text{brv}_y(x) &= u_y(x, y)
\end{align*}
\]
Nash equilibrium

A **Nash equilibrium** is a pair of mutual best responses:

$$\text{brv}_x(y) = u_x(x, y),$$
$$\text{brv}_y(x) = u_y(x, y)$$

equivalently

$$u_x(x, y) \geq u_x(\bar{x}, y), \quad \forall \bar{x} \in \Delta_m$$
$$u_y(x, y) \geq u_y(x, \bar{y}), \quad \forall \bar{y} \in \Delta_n$$
A **Nash equilibrium** is a pair of mutual best responses:

\[
\begin{align*}
\text{brv}_x(y) &= u_x(x, y), \\
\text{brv}_y(x) &= u_y(x, y)
\end{align*}
\]
equivalently

\[
\begin{align*}
u_x(x, y) &\geq u_x(\bar{x}, y), & \forall \bar{x} \in \Delta_m \\
u_y(x, y) &\geq u_y(x, \bar{y}), & \forall \bar{y} \in \Delta_n
\end{align*}
\]

**Theorem (Nash 1950)**

*For any matrix game, a Nash equilibrium exists.*
Example—a strategy for $y$

$$A = \begin{bmatrix} 2 & 0 & 0 \\ -1 & 3 & 1 \end{bmatrix}$$

- If $x = (1, 0)$ then $y$ responds $j = 1$
- If $x = (0, 1)$ then $y$ responds $j = 2$
- If $x = (p, 1 - p)$ when is $y$ indifferent 1 and 2?
Example—a strategy for $y$

$$A = \begin{bmatrix} 2 & 0 & 0 \\ -1 & 3 & 1 \end{bmatrix}$$

- If $x = (1, 0)$ then $y$ responds $j = 1$
- If $x = (0, 1)$ then $y$ responds $j = 2$
- If $x = (p, 1 - p)$ when is $y$ indifferent 1 and 2?

$$u_y(x, 1) = u_y(x, 2)$$

$$2p - 1(1 - p) = 0p + 3(1 - p)$$

$$p = \frac{2}{3}$$
Example—a strategy for $x$

$$A = \begin{bmatrix} 2 & 0 & 0 \\ -1 & 3 & 1 \end{bmatrix}$$

- If $y = (q, 1 - q, 0)$ when is $x$ indifferent $1$ and $2$?
Example—a strategy for $x$

$$A = \begin{bmatrix}
2 & 0 & 0 \\
-1 & 3 & 1 \\
\end{bmatrix}$$

If $y = (q, 1-q, 0)$ when is $x$ indifferent 1 and 2?

$$u_x(1, y) = u_x(2, y)$$

$$-2q + 0(1-q) = q - 3(1-q)$$

$$q = \frac{1}{2}$$
Example—checking our work

\[
A = \begin{bmatrix}
2 & 0 & 0 \\
-1 & 3 & 1
\end{bmatrix}
\]

\[
x = \left( \frac{2}{3}, \frac{1}{3} \right),
\quad y = \left( \frac{1}{2}, \frac{1}{2}, 0 \right)
\]
Example—checking our work

\[
A = \begin{bmatrix}
2 & 0 & 0 \\
-1 & 3 & 1
\end{bmatrix}
\]

\[
x = \left( \frac{2}{3}, \frac{1}{3} \right), \quad y = \left( \frac{1}{2}, \frac{1}{2}, 0 \right)
\]

\[
\text{brv}_x(y) = \max_{i \in [n]} \{-1, -1\}
\]
Example—checking our work

\[ A = \begin{bmatrix} 2 & 0 & 0 \\ -1 & 3 & 1 \end{bmatrix} \]

\[ x = \left( \frac{2}{3}, \frac{1}{3} \right), \quad y = \left( \frac{1}{2}, \frac{1}{2}, 0 \right) \]

\[ \text{brv}_x(y) = \max_{i \in [n]} \{-1, -1\} \]

\[ \text{brv}_y(x) = \max_{j \in [m]} \left\{ 1, 1, \frac{1}{3} \right\} \]
Nash Equilibrium—computational complexity
(Papadimitriou 1994, Daskalakis et. al. 2009)

For general sum games:

- Finding a Nash equilibrium is PPAD-complete
- Simplex-like algorithm (Lemke and Howson 1964)
- Newton-like algorithm (Nisan et. al. 2007)
- Guess and check (Lipton et. al. 2003)
Matrix Games
Definitions—What is a Nash Equilibrium?
Formulation as Minimax Problem
Formulation as Linear Program
Solutions via Subgradient Methods
Solutions via Smoothing and Gradient Methods
Solutions via Online Learning

Extensive-form Games

Tips and Tricks
Minimax Setup

Claim:

\[(x, y) \text{ is Nash equilibrium} \iff (x, y) \text{ is a saddle-point of } \min_{x \in \Delta_m} \max_{y \in \Delta_n} x' Ay\]
Minimax Setup—proof

$(x, y)$ is Nash equilibrium $\Rightarrow$

\[-x'Ay = u_x(x, y) \geq u_x(\bar{x}, y) = -\bar{x}'Ay, \quad \forall \bar{x} \in \Delta_m\]
\[x'Ay = u_y(x, y) \geq u_y(x, \bar{y}) = x'\bar{y}, \quad \forall \bar{y} \in \Delta_n\]
Minimax Setup—proof

(x, y) is Nash equilibrium ⇒

\[-x'Ay = u_x(x, y) \geq u_x(\bar{x}, y) = -\bar{x}'Ay, \quad \forall \bar{x} \in \Delta_m\]

\[x'Ay = u_y(x, y) \geq u_y(x, \bar{y}) = x'A\bar{y} \quad \forall \bar{y} \in \Delta_n\]

therefore \(\forall \bar{x} \in \Delta_m, \bar{y} \in \Delta_n\)

\[x'A\bar{y} \leq x'Ay \leq \bar{x}'Ay\]
Minimax Setup—proof

$(x, y)$ is Nash equilibrium $\Rightarrow$

$$
-x' Ay = u_x(x, y) \geq u_x(\bar{x}, y) = -\bar{x}' Ay, \quad \forall \bar{x} \in \Delta_m
$$

$$
x' Ay = u_y(x, y) \geq u_y(x, \bar{y}) = x' A\bar{y} \quad \forall \bar{y} \in \Delta_n
$$

therefore $\forall \bar{x} \in \Delta_m, \bar{y} \in \Delta_n$

$$
x' A\bar{y} \leq x' Ay \leq \bar{x}' Ay
$$

Reverse direction is just as obvious.
Minimax Theorem

Theorem (von Neumann 1928)

\[
\min_{x \in \Delta_m} \max_{y \in \Delta_n} x' Ay = \max_{y \in \Delta_n} \min_{x \in \Delta_m} x' Ay
\]
Minimax Theorem

Theorem (von Neumann 1928)

\[
\min_{x \in \Delta_m} \max_{y \in \Delta_n} x' Ay = \max_{y \in \Delta_n} \min_{x \in \Delta_m} x' Ay
\]

Let \( v^* = \min_{x \in \Delta_m} \max_{y \in \Delta_n} x' Ay \)
Minimax Theorem

Theorem (von Neumann 1928)

\[ \min_{x \in \Delta_m} \max_{y \in \Delta_n} x' Ay = \max_{y \in \Delta_n} \min_{x \in \Delta_m} x' Ay \]

Let \( v^* = \min_{x \in \Delta_m} \max_{y \in \Delta_n} x' Ay \)

Max-min inequality:

Theorem (Boyd and Vandenberghe 2004)

\[ \max_{y \in \Delta_n} \min_{x \in \Delta_m} x' Ay \leq \min_{x \in \Delta_m} \max_{y \in \Delta_n} x' Ay \]
Minimax Theorem—proof

Let \((x, y)\) be a Nash equilibrium,

\[
\min_{x \in \Delta_m} \max_{y \in \Delta_n} \ x' A y \leq \max_{y \in \Delta_n} \ min_{x \in \Delta_m} \ x' A y
\]
Minimax Theorem—proof

Let \((x, y)\) be a Nash equilibrium,

\[
\min_{\bar{x} \in \Delta_m} \max_{\bar{y} \in \Delta_n} \bar{x}' A \bar{y} \leq \max_{\bar{y} \in \Delta_n} \min_{\bar{x} \in \Delta_m} \bar{x}' A \bar{y} = x' A y
\]
Minimax Theorem—proof

Let \((x, y)\) be a Nash equilibrium,

\[
\min_{\bar{x} \in \Delta_m} \max_{\bar{y} \in \Delta_n} \bar{x}' A \bar{y} \leq \max_{\bar{y} \in \Delta_n} \min_{\bar{x} \in \Delta_m} \bar{x}' A \bar{y} = x' A y = \min_{\bar{x} \in \Delta_m} \bar{x}' A y
\]

Max-min inequality implies inequalities must hold at equality.
Minimax Theorem—proof

Let \((x, y)\) be a Nash equilibrium,

\[
\min_{\bar{x} \in \Delta_m} \max_{\bar{y} \in \Delta_n} \bar{x}' A \bar{y} \leq \max_{\bar{y} \in \Delta_n} \min_{\bar{x} \in \Delta_m} \bar{x}' A \bar{y}
\]

\[
= x' A y
\]

\[
= \min_{\bar{x} \in \Delta_m} \bar{x}' A y
\]

\[
\leq \max_{\bar{y} \in \Delta_n} \min_{\bar{x} \in \Delta_m} \bar{x}' A \bar{y}
\]
Minimax Theorem—proof

Let \((x, y)\) be a Nash equilibrium,

\[
\begin{align*}
\min_{\bar{x} \in \Delta_m} \max_{\bar{y} \in \Delta_n} \bar{x}' A\bar{y} & \leq \max_{\bar{y} \in \Delta_n} x' A\bar{y} \\
& = x' Ay \\
& = \min_{\bar{x} \in \Delta_m} \bar{x}' A y \\
& \leq \max_{\bar{y} \in \Delta_n} \min_{\bar{x} \in \Delta_m} \bar{x}' A\bar{y}
\end{align*}
\]

Max-min inequality implies inequalities must hold at equality.
An **ε-Nash equilibrium** is a pair of mutual $\varepsilon$-best responses:

$$\brv_x(y) \leq u_x(x, y) + \varepsilon$$

$$\brv_y(x) \leq u_y(x, y) + \varepsilon$$
Approximate Nash equilibrium

An $\varepsilon$-Nash equilibrium is a pair of mutual $\varepsilon$-best responses:

$$brv_x(y) \leq u_x(x, y) + \varepsilon$$
$$brv_y(x) \leq u_y(x, y) + \varepsilon$$

The exploitability of a strategy is:

$$\epsilon_x(x) = brv_y(x) - v^*$$
$$\epsilon_y(y) = brv_x(y) + v^*$$
Matrix Games
Definitions—What is a Nash Equilibrium?
Formulation as Minimax Problem
Formulation as Linear Program
Solutions via Subgradient Methods
Solutions via Smoothing and Gradient Methods
Solutions via Online Learning

Extensive-form Games

Tips and Tricks
Linear programming solution

Consider

$$\max_{y \in \Delta} w' y$$

$$= \min_t t \text{ subject to:}$$

$$w \leq te$$

where $e = (1, \ldots, 1)$. 
Letting $w = A'x$:

$$\min_{x \in \Delta_m} \max_{y \in \Delta_n} x' Ay = \min_{x, t} t \text{ subject to:}$$

$$A'x \leq te$$
$$e'x = 1$$
$$x \geq 0$$
Linear programming—polynomial time solution

Can solve linear programming with:

- Simplex method (Dantzig 1987)
- Interior point algorithms (Boyd and Vandenberghe 2004)
- Ellipsoid algorithm (Khachiyan 1979, Lovasz 1988)
Matrix Games

Definitions—What is a Nash Equilibrium?
Formulation as Minimax Problem
Formulation as Linear Program
Solutions via Subgradient Methods
Solutions via Smoothing and Gradient Methods
Solutions via Online Learning

Extensive-form Games

Tips and Tricks
Consider,
\[
\min_{x \in \Delta_m} \text{brv}_y(x) = \min_{x \in \Delta_m} \max_{y \in \Delta_n} x' Ay
\]

This objective is convex, albeit non-smooth, and
\[
\frac{\partial}{\partial x} \text{brv}_y(x) = \{ Ay^* \mid y^* \text{ is a best response to } x \}\]
Projected Subgradient Method

Initialize $x_1 = e/m, \alpha > 0$
For $t = 1, \ldots, T$:

$$y_t \in \arg\max_{y \in \Delta_n} x'_t Ay$$
Projected Subgradient Method

Initialize $x_1 = e/m, \alpha > 0$
For $t = 1, \ldots, T$:

$$y_t \in \arg\max_{y \in \Delta_n} x_t' A y \quad [Ay_t \in \partial \text{brv}_y(x_t)]$$
Projected Subgradient Method

Initialize $x_1 = e/m, \alpha > 0$
For $t = 1, \ldots, T$:

$$y_t \in \arg\max_{y \in \Delta_n} x'_t Ay \quad [Ay_t \in \partial \brv_y(x_t)]$$

$$z = x_t - \alpha Ay_t$$
Projected Subgradient Method

Initialize $x_1 = e/m, \alpha > 0$
For $t = 1, \ldots, T$:

$$y_t \in \arg\max_{y \in \Delta_n} x_t' Ay \quad [Ay_t \in \partial \text{brv}_y(x_t)]$$

$$z = x_t - \alpha Ay_t$$

$$x_{t+1} = \arg\min_{x \in \Delta_n} \|x - z\|^2 = \Pi_{\Delta_m}(z)$$
Projected Subgradient Method

Initialize $x_1 = e/m, \alpha > 0$

For $t = 1, \ldots, T$:

$$y_t \in \text{argmax}_{y \in \Delta_n} x_t'Ay \quad [Ay_t \in \partial \text{brv}_y(x_t)]$$

$$z = x_t - \alpha Ay_t$$

$$x_{t+1} = \text{argmin}_{x \in \Delta_n} \|x - z\|^2 = \Pi_{\Delta_m}(z)$$

Output $x = \frac{1}{T} \sum_{t=1}^T x_t, y = \frac{1}{T} \sum_{t=1}^T y_t$. 
Projecting onto the simplex

\[ x^* = \arg\min_{x \in \Delta} \| x - z \|^2 \]
Projecting onto the simplex

\[ x^* = \arg\min_{x \in \Delta} \|x - z\|^2 = \arg\max_{x \in \Delta} -\|x\|^2 + 2x'z - \|z\|^2 \]
Projecting onto the simplex

\[ x^* = \arg\min_{x \in \Delta} \| x - z \|^2 = \arg\max_{x \in \Delta} -\| x \|^2 + 2x'z - \| z \|^2 \]

\[ = \arg\max_{x \in \Delta} x'z - \frac{1}{2}\| x \|^2 \]
Projecting onto the simplex

\[ x^* = \arg\min_{x \in \Delta} \|x - z\|^2 = \arg\max_{x \in \Delta} -\|x\|^2 + 2x'z - \|z\|^2 \]

\[ = \arg\max_{x \in \Delta} x'z - \frac{1}{2}\|x\|^2 \]

\[ x^* = (z + \lambda)_+ = \max\{0, z + \lambda\} \]

where \( \lambda \) is chosen so that \( x \in \Delta \).
Projecting onto the simplex
see (Duchi et. al. 2008) for $O(n)$ solution

Let $q = \text{sort}(z)$ and $Z_1 = \sum_{i=1}^{n} z_i$

For $i \in [n]$:

Solve $Z + (n - i + 1)\gamma = 1$

if $q_i + \gamma \geq 0$ then $\lambda = \gamma$; break

$Z_{i+1} = Z_i - q_i$

Output $x^* = (z + \lambda)_+$
Subgradient method convergence
extending (Zinkevich 2003)

Without loss of generality $n \geq m$:

Theorem

$$\epsilon(x) + \epsilon(y) \leq \frac{n + nmL^2T\alpha^2}{T\alpha}$$
Subgradient method convergence
extending (Zinkevich 2003)

Without loss of generality \( n \geq m \):

**Theorem**

\[
\epsilon(x) + \epsilon(y) \leq \frac{n + nmL^2T\alpha^2}{T\alpha}
\]

Choosing \( \alpha = 1/L\sqrt{mT} \)

\[
\epsilon_x(x) + \epsilon_y(y) \leq 2nL\sqrt{\frac{m}{T}}
\]
Towards mirror descent
(Nemirovski 2012)

Initialize $x_1 = e/m, \alpha > 0$
For $t = 1, \ldots, T$:

$$y_t \in \arg\max_{y \in \Delta_n} x_t'Ay$$
Towards mirror descent
(Nemirovski 2012)

Initialize $x_1 = e/m, \alpha > 0$
For $t = 1, \ldots, T$:

$y_t \in \arg\max_{y \in \Delta_n} x_t^t Ay$

$x_{t+1} = \arg\min_{x \in \Delta_n} \| x_t - \alpha Ay_t - x \|^2$
Towards mirror descent
(Nemirovski 2012)

Initialize $x_1 = e/m, \alpha > 0$
For $t = 1, \ldots, T$:

\[
y_t \in \arg\max_{y \in \Delta_n} x_t' Ay
\]

\[
x_{t+1} = \arg\min_{x \in \Delta_n} ||x_t - \alpha Ay_t - x||^2
\]

\[
= \arg\min_{x \in \Delta_n} x' (\alpha Ay_t - x_t) + \frac{1}{2} ||x||^2 - \frac{1}{2} ||x_t||^2
\]
Towards mirror descent
(Nemirovski 2012)

Initialize $x_1 = e/m, \alpha > 0$
For $t = 1, \ldots, T$:

$$y_t \in \arg\max_{y \in \Delta_n} x_t' Ay$$

$$x_{t+1} = \arg\min_{x \in \Delta_n} \|x_t - \alpha Ay_t - x\|^2$$

$$= \arg\min_{x \in \Delta_n} x' (\alpha Ay_t - x_t) + \frac{1}{2}\|x\|^2 - \frac{1}{2}\|x_t\|^2$$

$$= \arg\min_{x \in \Delta_n} x' (\alpha Ay_t - \nabla h(x_t)) + h(x) - h(x_t)$$
Towards mirror descent
(Nemirovski 2012)

Initialize $x_1 = e/m, \alpha > 0$
For $t = 1, \ldots, T$:

$$y_t \in \arg \max_{y \in \Delta_n} x_t' Ay$$

$$x_{t+1} = \arg \min_{x \in \Delta_n} \|x_t - \alpha Ay_t - x\|^2$$

$$= \arg \min_{x \in \Delta_n} x' (\alpha Ay_t - x_t) + \frac{1}{2} \|x\|^2 - \frac{1}{2} \|x_t\|^2$$

$$= \arg \min_{x \in \Delta_n} x' (\alpha Ay_t - \nabla h(x_t)) + h(x) - h(x_t)$$

With mirror descent choose an alternative $h(x)$
Bregman divergences and distance generating functions
(Bregman 1967)

A distance generating function is a 1-strongly convex function $h(x)$ on $\Delta$ such that $\forall x \in \Delta$:

- $2h(x) \geq \|x\|^2$, $\forall x \in \Delta$, and
A distance generating function is a 1-strongly convex function $h(x)$ on $\Delta$ such that $\forall x \in \Delta$:

- $2h(x) \geq \|x\|^2$, $\forall x \in \Delta$, and
- $h(x) \geq h(x_0) = 0$. 

(Bregman 1967)
A **distance generating function** is a $1$-strongly convex function $h(x)$ on $\Delta$ such that $\forall x \in \Delta$:

- $2h(x) \geq ||x||^2$, $\forall x \in \Delta$, and
- $h(x) \geq h(x_0) = 0$.

We say $h(x)$ fits $\Delta$ if we can efficiently solve

$$\min_{x \in \Delta} g'x + h(x)$$
A **distance generating function** is a 1-strongly convex function \( h(x) \) on \( \Delta \) such that \( \forall x \in \Delta \):

- \( 2h(x) \geq \|x\|^2 \), \( \forall x \in \Delta \), and
- \( h(x) \geq h(x_0) = 0 \).

We say \( h(x) \) fits \( \Delta \) if we can efficiently solve

\[
\min_{x \in \Delta} \ g'x + h(x)
\]

We define the **Bregman divergence** as

\[
D(x, y) = h(x) - h(y) - \nabla h(y)'(x - y)
\]
Negative entropy distance generating function

Let \( h(x) = x \log(x) + \log(n) - e'x + 1 \)

\[ \nabla h(x) = \log(x) \]
Let \( h(x) = x \log(x) + \log(n) - e'x + 1 \)

\[ \nabla h(x) = \log(x) \]

Let \( Z = \sum_{i=1}^{n} \exp(-g_i) \)
Negative entropy distance generating function

Let \( h(x) = x \log(x) + \log(n) - e'x + 1 \)

\[ \nabla h(x) = \log(x) \]

Let \( Z = \sum_{i=1}^{n} \exp(-g_i) \)

\[ \min_{x \in \Delta} \ g'x + h(x) = \log(Z) - \log(n) \]
Negative entropy distance generating function

Let $h(x) = x \log(x) + \log(n) - e'x + 1$

$$\nabla h(x) = \log(x)$$

Let $Z = \sum_{i=1}^{n} \exp(-g_i)$

$$\min_{x \in \Delta} g'x + h(x) = \log(Z) - \log(n)$$

$x^* = \arg\min_{x \in \Delta} g'x + h(x)$

$$= \exp(-g)/Z$$
Exponentiated Subgradient Method
(Kivinen and Warmuth 1994)

Initialize $x_1 = e/m, \alpha > 0$
For $t = 1, \ldots, T$:

$$y_t \in \arg\max_{y \in \Delta_n} x_t' Ay$$
Exponentiated Subgradient Method
(Kivinen and Warmuth 1994)

Initialize $x_1 = e/m, \alpha > 0$
For $t = 1, \ldots, T$:

$$y_t \in \arg\max_{y \in \Delta_n} x_t' Ay \quad [Ay_t \in \partial \text{brv}_y(x_t)]$$

$$x_{t+1} = x_t \exp(-\alpha Ay_t)/Z$$

Output $x = \frac{1}{T} \sum_{t=1}^T x_t, y = \frac{1}{T} \sum_{t=1}^T y_t$. 
Exponentiated Subgradient Method
(Kivinen and Warmuth 1994)

Initialize $x_1 = e/m, \alpha > 0$
For $t = 1, \ldots, T$:

$$y_t \in \arg\max_{y \in \Delta_n} x_t' Ay \quad [Ay_t \in \partial \text{brv}_y(x_t)]$$

$$x_{t+1} = x_t \exp(-\alpha Ay_t)/Z$$
Exponentiated Subgradient Method
(Kivinen and Warmuth 1994)

Initialize $x_1 = e/m, \alpha > 0$
For $t = 1, \ldots, T$:

$$y_t \in \arg\max_{y \in \Delta_n} x_t' Ay \quad [Ay_t \in \partial \brv_y(x_t)]$$

$$x_{t+1} = x_t \exp(-\alpha Ay_t)/Z$$

Output $x = \frac{1}{T} \sum_{t=1}^{T} x_t$, $y = \frac{1}{T} \sum_{t=1}^{T} y_t$. 
Exponentiated Subgradient method convergence

Without loss of generality $n \geq m$:

**Theorem**

$$\epsilon(x) + \epsilon(y) \leq \frac{\log(n) + nmL^2T\alpha^2}{T\alpha}$$
Without loss of generality $n \geq m$:

**Theorem**

$$\epsilon(x) + \epsilon(y) \leq \frac{\log(n) + nmL^2T\alpha^2}{T\alpha}$$

Choosing $\alpha = \sqrt{\log(n)/nmTL^2}$

$$\epsilon_x(x) + \epsilon_y(y) \leq 2L\sqrt{\frac{nm\log(n)}{T}}$$
Further reading on Subgradient methods

- Primal-dual subgradient methods for convex problems (Nesterov 2009)
- Universal gradient methods for convex optimization problems (Nesterov 2013)
Matrix Games
Definitions—What is a Nash Equilibrium?
Formulation as Minimax Problem
Formulation as Linear Program
Solutions via Subgradient Methods
Solutions via Smoothing and Gradient Methods
Solutions via Online Learning

Extensive-form Games

Tips and Tricks
Smooth vs. non-smooth optimization

For non-smooth $f$, subgradient methods achieve

$$f(x) - f(x^*) \in O\left(\frac{1}{\sqrt{T}}\right)$$

For smooth $f$, accelerated gradient methods achieve (Nesterov 1984)

$$f(x) - f(x^*) \in O\left(\frac{1}{T^{2}}\right)$$

Can we smooth our objective for better asymptotic convergence?
Smooth vs. non-smooth optimization

For non-smooth $f$, subgradient methods achieve

$$f(x) - f(x^*) \in O \left( \frac{1}{\sqrt{T}} \right)$$

For smooth $f$, gradient descent achieves

$$f(x) - f(x^*) \in O \left( \frac{1}{T} \right)$$
Smooth vs. non-smooth optimization

For non-smooth $f$, subgradient methods achieve

$$f(x) - f(x^*) \in O\left(\frac{1}{\sqrt{T}}\right)$$

For smooth $f$, gradient descent achieves

$$f(x) - f(x^*) \in O\left(\frac{1}{T}\right)$$

For smooth $f$, accelerated gradient methods achieve (Nesterov 1984)

$$f(x) - f(x^*) \in O\left(\frac{1}{T^2}\right)$$
Smooth vs. non-smooth optimization

For non-smooth \( f \), subgradient methods achieve

\[
f(x) - f(x^*) \in O \left( \frac{1}{\sqrt{T}} \right)
\]

For smooth \( f \), gradient descent achieves

\[
f(x) - f(x^*) \in O \left( \frac{1}{T} \right)
\]

For smooth \( f \), accelerated gradient methods achieve (Nesterov 1984)

\[
f(x) - f(x^*) \in O \left( \frac{1}{T^2} \right)
\]

Can we smooth our objective for better asymptotic convergence?
An accelerated gradient method  
(Auslender and Teboulle 2006)

Initialize $x_1 = u_1 = \epsilon/m, \alpha > 0$

For $t = 1, \ldots, T$:

$$v_t = \frac{(t - 1)x_t + 2u_t}{t + 1}$$

Output $x_{T+1}$
An accelerated gradient method
(Auslender and Teboulle 2006)

Initialize $x_1 = u_1 = e/m, \alpha > 0$
For $t = 1, \ldots, T$:

$$v_t = \frac{(t - 1)x_t + 2u_t}{t + 1}$$

$$u_{t+1} = \arg\min_{x \in \Delta} \alpha(t + 1)\nabla f(v_t)'x + D(x, u_1)$$

Output $x_{T+1}$
An accelerated gradient method
(Auslender and Teboulle 2006)

Initialize $x_1 = u_1 = e/m, \alpha > 0$
For $t = 1, \ldots, T$:

$$v_t = \frac{(t - 1)x_t + 2u_t}{t + 1}$$

$$u_{t+1} = \arg\min_{x \in \Delta} \alpha(t + 1)\nabla f(v_t)'x + D(x, u_1)$$

$$x_{t+1} = \frac{(t - 1)x_t + 2u_{t+1}}{t + 1}$$

Output $x_{T+1}$
Further reading

- Linear coupling: An ultimate unification of gradient and mirror descent (Allen-Zhu and Orecchia 2014)
- Templates for convex cone problems with applications to sparse signal recovery (Becker et. al. 2010)
- On accelerated proximal gradient methods for convex-concave optimization (Tseng 2008)
Consider the function for d.g.f. \( h(y) \) and \( \mu > 0 \):

\[
\text{brv}_y(x) \approx f_\mu(x) = \max_{y \in \Delta_n} x^tAy - \mu h(y)
\]
Consider the function for d.g.f. $h(y)$ and $\mu > 0$:

$$\text{brv}_y(x) \approx f_\mu(x) = \max_{y \in \Delta_n} x' Ay - \mu h(y)$$

Let $D = \max_{y \in \Delta_n} h(y)$, we have

$$\text{brv}_y(x) - \mu D \leq f_\mu(x) \leq \text{brv}_y(x)$$
Conjugate smoothing
(Nesterov 2005)

Consider the function for d.g.f. $h(y)$ and $\mu > 0$:

$$brv_y(x) \approx f_\mu(x) = \max_{y \in \Delta_n} x' Ay - \mu h(y)$$

Let $D = \max_{y \in \Delta_n} h(y)$, we have

$$brv_y(x) - \mu D \leq f_\mu(x) \leq brv_y(x)$$

$$\nabla f_\mu(x) = Ay_t \quad [y_t = \arg\max_{y \in \Delta_n} x' Ay - \mu h(y)]$$
Conjugate smoothing
(Nesterov 2005)

Consider the function for d.g.f. $h(y)$ and $\mu > 0$:

$$\text{brv}_y(x) \approx f_\mu(x) = \max_{y \in \Delta_n} x' Ay - \mu h(y)$$

Let $D = \max_{y \in \Delta_n} h(y)$, we have

$$\text{brv}_y(x) - \mu D \leq f_\mu(x) \leq \text{brv}_y(x)$$

$$\nabla f_\mu(x) = Ay_t \quad [y_t = \arg\max_{y \in \Delta_n} x' Ay - \mu h(y)]$$

And $f_\mu(x)$ is $\frac{L}{\mu}$-smooth.
Typical accelerated methods have convergence bounds like:

\[ f_\mu(x_{T+1}) - f_\mu(x^*) \leq \frac{LD}{\mu T^2} \]
Conjugate smoothing for matrix games
(Nesterov 2005)

Typical accelerated methods have convergence bounds like:

\[ f_\mu(x_{T+1}) - f_\mu(x^*) \leq \frac{LD}{\mu T^2} \]

Using \( \text{brv}_y(x) - \mu D \leq f_\mu(x) \leq \text{brv}_y(x) \)

\[ \epsilon(x_{T+1}) = f(x_{T+1}) - v^* \leq \frac{LD}{\mu T^2} + \mu D \]
Conjugate smoothing for matrix games
(Nesterov 2005)

Typical accelerated methods have convergence bounds like:

$$ f_\mu(x_{T+1}) - f_\mu(x^*) \leq \frac{LD}{\mu T^2} $$

Using $brv_y(x) - \mu D \leq f_\mu(x) \leq brv_y(x)$

$$ \epsilon(x_{T+1}) = f(x_{T+1}) - v^* \leq \frac{LD}{\mu T^2} + \mu D $$

Choosing $\mu = \frac{\sqrt{L}}{T}$

$$ \epsilon(x_{T+1}) \leq \frac{D\sqrt{L}}{T} $$

An order of magnitude better than subgradient methods!
Excessive Gap Technique
(Nesterov 2005)

What if we don’t know $T$ in advance?
What if we don’t know $T$ in advance? Consider the smoothed pair of problems:

\[
\begin{align*}
&\min_{x \in \Delta_m} \max_{y \in \Delta_n} \ x' Ay - \mu_y h_y(y), \quad \text{and} \\
&\max_{y \in \Delta_n} \min_{x \in \Delta_m} \quad - x' Ay + \mu_x h_x(x)
\end{align*}
\]
Excessive Gap Technique
(Nesterov 2005)

What if we don’t know $T$ in advance? Consider the smoothed pair of problems:

$$\min_{x \in \Delta_m} \max_{y \in \Delta_n} x' Ay - \mu_y h_y(y), \text{ and}$$
$$\max_{y \in \Delta_n} \min_{x \in \Delta_m} -x' Ay + \mu_x h_x(x)$$

As we optimize $x$ is using an accelerated method, we can decrease $\mu_x$. Then, we switch to optimizing $y$ and decreasing $\mu_y$. 
Additional optimization focused methods

- Interior-point methods (Pays 2014)
- Double-oracle methods (Bosansky et. al. 2013, Zinkevich et. al. 2007)
Matrix Games
Definitions—What is a Nash Equilibrium?
Formulation as Minimax Problem
Formulation as Linear Program
Solutions via Subgradient Methods
Solutions via Smoothing and Gradient Methods
Solutions via Online Learning

Extensive-form Games

Tips and Tricks
Adversarial bandits

For $t = 1, \ldots, T$:

Choose $x_t \in \Delta$

Adversary chooses $u_t$, subject to $\|u_t\|_\infty \leq L$

Observe $u_t$, and receive $u_t'x_t$ utility
Adversarial bandits

For $t = 1, \ldots, T$:

Choose $x_t \in \Delta$

Adversary chooses $u_t$, subject to $\|u_t\|_{\infty} \leq L$

Observe $u_t$, and receive $u'_t x_t$ utility

The algorithm's **average overall regret** is the average benefit of having chosen the best single action in hindsight:

$$R_T = \max_{a \in A} \left[ R_T(a) = \frac{1}{T} \sum_{t=1}^{T} u_t(a) - u'_t x_t \right]$$
Adversarial bandits

For $t = 1, \ldots, T$:

Choose $x_t \in \Delta$

Adversary chooses $u_t$, subject to $\|u_t\|_\infty \leq L$

Observe $u_t$, and receive $u'_tx_t$ utility

The algorithm’s **average overall regret** is the average benefit of having chosen the best single action in hindsight:

$$R_T = \max_{a \in A} \left[ R_T(a) = \frac{1}{T} \sum_{t=1}^{T} u_t(a) - u'_tx_t \right]$$

An algorithm is **no-regret** if it’s average overall regret grows sublinearly in $T$. 
Subgradient method and mirror descent are no-regret with

\[ R_T \in O \left( L \sqrt{nT} \right) \]
Why do we care about no-regret algorithms?
(see Waugh 2009 for proof)

Theorem

The average strategies of two no-regret algorithms in self-play with no more than $\varepsilon$ average overall regret form a $2\varepsilon$-equilibrium.
Why do we care about no-regret algorithms?
(see Waugh 2009 for proof)

Theorem

The average strategies of two no-regret algorithms in self-play with no more than $\varepsilon$ average overall regret form a $2\varepsilon$-equilibrium.

Let $A$ be a no-regret algorithm on $\Delta_m$ and $B$ on $\Delta_n$.
For $t = 1, \ldots, T$:

\[
\begin{align*}
x_t &= \text{Strategy}(A) \\
y_t &= \text{Strategy}(B) \\
\text{Update}(A, -Ay_t) \\
\text{Update}(B, A'x_t)
\end{align*}
\]

Output $\bar{x}_T = \frac{1}{T} \sum_{t=1}^{T} x_t$, $\bar{y}_T = \frac{1}{T} \sum_{t=1}^{T} y_t$
No-regret algorithms: Hedge/weighted Majority
(Freund and Schapire 1996)

Choose $x_{t+1}(a) \propto \exp(\alpha R_t(a))$

$$R_T \in O \left( L\sqrt{T \log(n)} \right)$$

Related to Nesterov’s dual averaging (2009)
No-regret algorithms: Follow the perturbed leader
(Kalai and Vempala 2004)

Choose $x_{t+1} = \arg\max R_t - \log(\epsilon)/\lambda$, where $\epsilon \sim (0, 1)$

$$R_T \in O \left( L \sqrt{T \log(n)} \right)$$
Regret matching
(Blackwell 1956, Hart and Mas-Colell 1999)

Choose \( x_{t+1} \propto (R_t)_+ \)

\[ R_T \in O \left( L\sqrt{nT} \right) \]
Pure regret matching
(Tammelin, Gibson 2017, Cesa-Bianchi and Lugosi 2006)

Sample $x_{t+1} \sim (R_t)_+$

$$R_T \in O \left( L \sqrt{2nT} \right)$$

Regrets are integral, and average strategies are counts. Only need to examine one row and one column of $A$ each iteration.
Regret matching-plus
(Tammelin 2014)

Choose $x_{t+1} \propto (R_t^+)_{+}$
Update $R^+_{t+1} = (R^+_t + u_t - u'_tx_t)_{+}$

$$R_T \in O\left(L\sqrt{nT}\right)$$
Regret matching-plus
(Tammelin 2014)

Choose \( x_{t+1} \propto (R_t^+) \)_+
Update \( R_{t+1}^+ = (R_t^+ + u_t - u'_t x_t)_+ \)

\[
R_T \in O \left( L\sqrt{nT} \right)
\]

\[
x_t = \text{Strategy}(\mathcal{A})
\]
Update(\( \mathcal{A}, -Ay_t \))
\[
y_t = \text{Strategy}(\mathcal{B})
\]
Update(\( \mathcal{B}, A'x_t \))
Regret matching-plus
(Tammelin 2014)

Choose $x_{t+1} \propto (R_t^+)_+$
Update $R_{t+1}^+ = (R_t^+ + u_t - u_t'x_t)_+$

$R_T \in O\left(L\sqrt{nT}\right)$

$x_t = \text{Strategy} (\mathcal{A})$
Update $(\mathcal{A}, -A y_t)$
$y_t = \text{Strategy} (\mathcal{B})$
Update $(\mathcal{B}, A' x_t)$

Output $\bar{x}_T = \frac{2}{T(T+1)} \sum_{t=1}^{T} tx_t, \bar{y}_T = \frac{2}{T(T+1)} \sum_{t=1}^{T} ty_t$
Matrix Games

Extensive-form Games
  Sequence Form Representation
  Dilated distance generating functions
  Counterfactual regret minimization

Tips and Tricks
Expressiveness of Matrix Games
Kuhn Poker (Kuhn 1950)

- The players ante a single chip
- Each player is dealt a random card from a deck containing a Jack, a Queen and a King
- The first player may check, or bet one chip
- When facing a bet, a player can call or fold forfeiting the pot
- Calling leads to a showdown, player with higher card wins

Natural strategy representation is $16 \times 16$. Can be represented as a $27 \times 64$ matrix game. The row player's actions determine \{bet, check/call, check/fold\} for each card. Can represent any finite scenario, but often not efficiently.
Expressiveness of Matrix Games
Kuhn Poker (Kuhn 1950)

- The players ante a single chip
- Each player is dealt a random card from a deck containing a Jack, a Queen and a King
- The first player may check, or bet one chip
- When facing a bet, a player can call or fold forfeiting the pot
- Calling leads to a showdown, player with higher card wins

*Natural* strategy representation is $16 \times 16$. 
Expressiveness of Matrix Games
Kuhn Poker (Kuhn 1950)

- The players ante a single chip
- Each player is dealt a random card from a deck containing a Jack, a Queen and a King
- The first player may check, or bet one chip
- When facing a bet, a player can call or fold forfeiting the pot
- Calling leads to a showdown, player with higher card wins

Natural strategy representation is $16 \times 16$. Can be represented as a $27 \times 64$ matrix game.
Expressiveness of Matrix Games

Kuhn Poker (Kuhn 1950)

- The players ante a single chip
- Each player is dealt a random card from a deck containing a Jack, a Queen and a King
- The first player may check, or bet one chip
- When facing a bet, a player can call or fold forfeiting the pot
- Calling leads to a showdown, player with higher card wins

*Natural* strategy representation is $16 \times 16$.
Can be represented as a $27 \times 64$ matrix game.
The row player’s actions determine \{bet, check/call, check/fold\} for each card.
Expressiveness of Matrix Games
Kuhn Poker (Kuhn 1950)

- The players ante a single chip
- Each player is dealt a random card from a deck containing a Jack, a Queen and a King
- The first player may check, or bet one chip
- When facing a bet, a player can call or fold forfeiting the pot
- Calling leads to a showdown, player with higher card wins

_Natural_ strategy representation is $16 \times 16$. Can be represented as a $27 \times 64$ matrix game. The row player’s actions determine \{bet, check/call, check/fold\} for each card. Can represent any finite scenario, but often not efficiently.
Extensive-form game
(Osborne and Rubinstein 1994, Fudenburg and Tirole 1991)

- Defined $\mathcal{H}$ the set of histories,
Extensive-form game
( Osborne and Rubinstein 1994, Fudenburg and Tirole 1991)

- Defined $\mathcal{H}$ the set of histories,
- We denote the root history as $\phi \in \mathcal{H}$,
Extensive-form game
(Osborne and Rubinstein 1994, Fudenburg and Tirole 1991)

- Defined $\mathcal{H}$ the set of histories,
- We denote the root history as $\phi \in \mathcal{H},$
- Let $A(h)$ be the set of actions available from $h$, $ha$ is the history after taking action $a \in A(h)$ from $h,$
Extensive-form game
(Osborne and Rubinstein 1994, Fudenburg and Tirole 1991)

- Defined $\mathcal{H}$ the set of histories,
- We denote the root history as $\phi \in \mathcal{H}$,
- Let $A(h)$ be the set of actions available from $h$, $ha$ is the history after taking action $a \in A(h)$ from $h$,
- $\mathcal{Z} \subseteq \mathcal{H}$ are the terminal histories—histories with no children,
Extensive-form game
(Osborne and Rubinstein 1994, Fudenburg and Tirole 1991)

- Defined $\mathcal{H}$ the set of histories,
- We denote the root history as $\phi \in \mathcal{H}$,
- Let $A(h)$ be the set of actions available from $h$, $ha$ is the history after taking action $a \in A(h)$ from $h$,
- $Z \subseteq \mathcal{H}$ are the terminal histories—histories with no children,
- $P : \mathcal{H} \to \{x, y, c\}$ is the player choice function, determining which player, or chance acts at a non-terminal history,
Extensive-form game
(Osborne and Rubinstein 1994, Fudenburg and Tirole 1991)

- Defined $\mathcal{H}$ the set of histories,
- We denote the root history as $\phi \in \mathcal{H}$,
- Let $A(h)$ be the set of actions available from $h$, $ha$ is the history after taking action $a \in A(h)$ from $h$,
- $\mathcal{Z} \subseteq \mathcal{H}$ are the terminal histories—histories with no children,
- $P : \mathcal{H} \rightarrow \{x, y, c\}$ is the player choice function, determining which player, or chance acts at a non-terminal history,
- For each history where $P(h) = c$, $\sigma(h) \in \Delta_{A(h)}$ defines chance’s strategy,
Extensive-form game
(Osborne and Rubinstein 1994, Fudenburg and Tirole 1991)

- Defined $\mathcal{H}$ the set of histories,
- We denote the root history as $\phi \in \mathcal{H}$,
- Let $A(h)$ be the set of actions available from $h$, $ha$ is the history after taking action $a \in A(h)$ from $h$,
- $\mathcal{Z} \subseteq \mathcal{H}$ are the terminal histories—histories with no children,
- $P : \mathcal{H} \rightarrow \{x, y, c\}$ is the player choice function, determining which player, or chance acts at a non-terminal history,
- For each history where $P(h) = c$, $\sigma(h) \in \Delta_{A(h)}$ defines chance’s strategy,
- $u_i : \mathcal{Z} \rightarrow \mathbb{R}$ is player $i$’s utility function,
Extensive-form game
(Osborne and Rubinstein 1994, Fudenburg and Tirole 1991)

- Defined $\mathcal{H}$ the set of histories,
- We denote the root history as $\phi \in \mathcal{H}$,
- Let $A(h)$ be the set of actions available from $h$, $ha$ is the history after taking action $a \in A(h)$ from $h$,
- $\mathcal{Z} \subseteq \mathcal{H}$ are the terminal histories—histories with no children,
- $P : \mathcal{H} \to \{x, y, c\}$ is the player choice function, determining which player, or chance acts at a non-terminal history,
- For each history where $P(h) = c$, $\sigma(h) \in \Delta_{A(h)}$ defines chance’s strategy,
- $u_i : \mathcal{Z} \to \mathbb{R}$ is player i’s utility function,
- Again, the game is zero-sum: $u_x(z) = -u_y(z)$.
Extensive-form game: Imperfect information
(Osborne and Rubinstein 1994, Fudenburg and Tirole 1991)

- $I_i$, player $i$’s information partition, partitions the histories where $i$ acts,
Extensive-form game: Imperfect information
(Osborne and Rubinstein 1994, Fudenburg and Tirole 1991)

- $\mathcal{I}_i$, player $i$’s information partition, partitions the histories where $i$ acts,
- $I \in \mathcal{I}_i$ is an information set, and two histories $h, h' \in \mathcal{I}$ are indistinguishable. This requires $A(h) = A(h')$. 
Extensive-form game: Imperfect information
(Osborne and Rubinstein 1994, Fudenburg and Tirole 1991)

- $\mathcal{I}_i$, player $i$’s information partition, partitions the histories where $i$ acts,
- $I \in \mathcal{I}_i$ is an information set, and two histories $h, h' \in \mathcal{I}$ are indistinguishable. This requires $A(h) = A(h')$.
- A strategy for player $i$ is $\sigma_i : \mathcal{I}_i \to \Delta_{A(I)}$. 
We additionally require that a player cannot be forced by the rules of the game to forget what they at one point knew.
We additionally require that a player cannot be forced by the rules of the game to forget what they at one point knew.

A game has **perfect recall** if all indistinguishable histories, $h, h' \in I_i$, share the same sequence of past decisions.
Perfect recall implies that each information set/action pair, \((I, a)\), uniquely defines an entire sequence of decisions.
Perfect recall implies that each information set/action pair, \((I, a)\), uniquely defines an entire sequence of decisions.

Let \(\Gamma_i = \{(I, a) \mid I \in \mathcal{I}_i, a \in A(I)\} \cup \{\emptyset\}\) be the set of player \(i\)’s sequences, where \(\emptyset\) is the empty sequence.
Perfect recall implies that each information set/action pair, 
\((I, a)\), uniquely defines an entire sequence of decisions.

Let \(\Gamma_i = \{(I, a) \mid I \in \mathcal{I}_i, a \in A(I)\} \cup \{\phi\}\) be the set of player \(i\)'s sequences, where \(\phi\) is the empty sequence.

Each information set has a unique parent sequence, which we denote \(\text{parent}(I)\).
Sequence form representation
(von Stengel 1996)

- Perfect recall implies that each information set/action pair, \((I, a)\), uniquely defines an entire sequence of decisions.
- Let \(\Gamma_i = \{(I, a) \mid I \in \mathcal{I}_i, a \in A(I)\} \cup \{\phi\}\) be the set of player \(i\)'s sequences, where \(\phi\) is the empty sequence.
- Each information set has a unique parent sequence, which we denote \(\text{parent}(I)\).
- Let \(\text{Reach}(I, a)\) be the set of information sets directly reachable from taking \(a\) at \(I\).
We call \( x : \Gamma_x \to \mathbb{R} \) a \textit{realization plan}. We require a realization plan satisfy:

\begin{itemize}
  \item \( x(u) \geq 0, \forall u \in \Gamma \),
\end{itemize}
We call \( x : \Gamma_x \rightarrow \mathbb{R} \) a realization plan. We require a realization plan satisfy:

- \( x(u) \geq 0, \forall u \in \Gamma \),
- \( x(\phi) = 1 \), and
We call \( x : \Gamma_x \to \mathbb{R} \) a \textbf{realization plan}. We require a realization plan satisfy:

- \( x(u) \geq 0, \forall u \in \Gamma \),
- \( x(\phi) = 1 \), and
- \( x(\text{parent}(I)) = \sum_{a \in A(I)} w(I, a), \forall I \in \mathcal{I}_i. \)
We call $x : \Gamma_x \to \mathbb{R}$ a **realization plan**. We require a realization plan satisfy:

- $x(u) \geq 0, \forall u \in \Gamma,$
- $x(\phi) = 1,$ and
- $x(\text{parent}(I)) = \sum_{a \in A(I)} w(I, a), \forall I \in \mathcal{I}_i.$
We call $x : \Gamma_x \rightarrow \mathbb{R}$ a \textbf{realization plan}. We require a realization plan satisfy:

- $x(u) \geq 0, \forall u \in \Gamma$, 
- $x(\phi) = 1$, and 
- $x(\text{parent}(I)) = \sum_{a \in A(I)} w(I, a), \forall I \in \mathcal{I}_i$.

We can encode these as linear constraints:

$\Sigma_1 = \{x \mid Ex = e, x \geq 0\}$ and $\Sigma_2 = \{y \mid Fy = f, y \geq 0\}$. 

(von Stengel 1996)
We call $x : \Gamma_x \rightarrow \mathbb{R}$ a realization plan. We require a realization plan satisfy:

- $x(u) \geq 0, \forall u \in \Gamma,$
- $x(\phi) = 1,$ and
- $x(\text{parent}(I)) = \sum_{a \in A(I)} w(I, a), \forall I \in \mathcal{I}_i.$

We can encode these as linear constraints:

$\Sigma_1 = \{ x \mid Ex = e, x \geq 0 \}$ and $\Sigma_2 = \{ y \mid Fy = f, y \geq 0 \}.$

We define $\sigma_x(I, \cdot) \propto x(I, \cdot).$
Sequence form Minimax Problem

\[
\min_{x \in \Sigma_1} \max_{y \in \Sigma_2} \ y'Ax
\]

That is, the payoffs are a bi-linear product of the realization plans.
Sequence form linear programming
(Koller and Pfeffer 1995, Koller et. al. 1996)

\[
\min_{x,u} f'u \text{ subject to:} \\
Fu \geq -A'x \\
Ex = e \\
x \geq 0
\]
Sequence form linear programming
(Koller and Pfeffer 1995, Koller et. al. 1996)

$$\min_{x, u} f'u \text{ subject to:}$$

$$Fu \geq -A'x$$

$$Ex = e$$

$$x \geq 0$$

$u$ is indexed by $y$'s sequences and represents the value of that sequence to the opponent.
Matrix Games

Extensive-form Games
- Sequence Form Representation
- Dilated distance generating functions
- Counterfactual regret minimization

Tips and Tricks
Dilated prox function
(Hoda et al. 2010)

\[ h(x) = \sum_{I \in \mathcal{I}_i} x (\text{parent}(I)) h_\Delta (\sigma_x(I, \cdot)) \]
Dilated prox function
(Hoda et. al. 2010)

\[ h(x) = \sum_{I \in \mathcal{I}_i} x(\text{parent}(I)) h_\Delta (\sigma_x(I, \cdot)) \]

We minimize \( h \) recursively, solving *terminal* information sets first, then adding the value of \( h_I(\sigma_x(I, \cdot)) \) to the utility \( \text{parent}(I) \).
Dilated prox function
(Hoda et. al. 2010)

\[ h(x) = \sum_{I \in I_i} x(\text{parent}(I)) h_{\Delta}(\sigma_x(I, \cdot)) \]

We minimize \( h \) recursively, solving terminal information sets first, then adding the value of \( h_I(\sigma_x(I, \cdot)) \) to the utility \( \text{parent}(I) \).

\[ \tilde{g}(I, a) = g(I, a) + \sum_{I' \in \text{Reach}(I, a)} \frac{x(I')}{x(I, a)} h_{I'}(\sigma_x(I', \cdot)) \]
Choosing $h_\Delta(x) = x \log(x)$,

$$h(x) = \sum_{I \in I_i} \beta_I x(\text{parent}(I)) h_\Delta(\sigma_x(I, \cdot))$$

With the appropriate choice of $\beta$, we can improve convergence of optimization-style algorithms.
Weighted dilated entropy prox functions
(Hoda et. al. 2010, Kroer et. al. 2015)

Choosing $h_\Delta(x) = x \log(x)$,

$$h(x) = \sum_{I \in \mathcal{I}_i} \beta_I x(\text{parent}(I)) h_\Delta(\sigma_x(I, \cdot))$$

With the appropriate choice of $\beta$, we can improve convergence of optimization-style algorithms. Roughly, allow the strategy to change more rapidly towards the root of the information tree.
Matrix Games

Extensive-form Games
  Sequence Form Representation
  Dilated distance generating functions
  Counterfactual regret minimization

Tips and Tricks
Counterfactual regret
(Zinkevich et. al. 2008)

Can we build no-regret algorithms for realization plans, using standard no-regret learning algorithms? Yes!
Counterfactual regret
(Zinkevich et. al. 2008)

Can we build no-regret algorithms for realization plans, using standard no-regret learning algorithms? Yes!

\[ u^t(I, a) = u^t(I, a) + \sum_{I' \in \text{Reach}(I,a)} u_t(I') \]
Counterfactual regret
(Zinkevich et. al. 2008)

Can we build no-regret algorithms for realization plans, using standard no-regret learning algorithms? Yes!

\[ u^t(I, a) = u^t(I, a) + \sum_{I' \in \text{Reach}(I, a)} u^t(I') \]

\[ u^t(I) = \sum_{a \in A(I)} \sigma_x(I, a) u^t(I, a) \]
Counterfactual regret
(Zinkevich et. al. 2008)

Can we build no-regret algorithms for realization plans, using standard no-regret learning algorithms? Yes!

\[ u^t(I, a) = u^t(I, a) + \sum_{I' \in \text{Reach}(I, a)} u^t(I') \]

\[ u^t(I) = \sum_{a \in A(I)} \sigma_x(I, a) u^t(I, a) \]

\[ r^t(I, a) = u^t(I, a) - u^t(I) \]
Counterfactual regret
(Zinkevich et. al. 2008)

Can we build no-regret algorithms for realization plans, using standard no-regret learning algorithms? Yes!

\[ u^t(I, a) = u^t(I, a) + \sum_{I' \in \text{Reach}(I, a)} u_t(I') \]

\[ u^t(I) = \sum_{a \in A(I)} \sigma_x(I, a) u^t(I, a) \]

\[ r^t(I, a) = u^t(I, a) - u^t(I) \]

We call \( u^t(I, a) \) the **counterfactual utility** at time \( t \) of taking sequence \( (I, a) \), and \( r^t(I, a) \), the immediate **counterfactual regret**.
Counterfactual regret minimization
(Zinkevich et. al. 2008)

Theorem

Overall regret is bounded by the sum of per information set counterfactual regret.

$$R_T \leq \sum_{I \in \mathcal{I}} (R_T(I))_+$$
Counterfactual regret minimization
(Zinkevich et. al. 2008)

Theorem

*Overall regret is bounded by the sum of per information set counterfactual regret.*

\[ R^T \leq \sum_{I \in \mathcal{I}} (R^T(I))_+ \]

Theorem

*Two algorithms minimizing counterfactual regret in self-play converge to a Nash equilibrium.*
Matrix Games

Extensive-form Games

Tips and Tricks
Why regret matching? Why CFR?

CFR has an inferior iteration complexity, and regret matching a suboptimal regret bound, why?

- Computationally cheap! \((\exp)\) is expensive
Why regret matching? Why CFR?

CFR has an inferior iteration complexity, and regret matching a suboptimal regret bound, why?

- Computationally cheap! $\exp$ is expensive
- No parameter tuning (sort of)
Why regret matching? Why CFR?

CFR has an inferior iteration complexity, and regret matching a suboptimal regret bound, why?

- Computationally cheap! \((\exp)\) is expensive
- No parameter tuning (sort of)
- Pruning! (some actions have probability zero)
Why regret matching? Why CFR?

CFR has an inferior iteration complexity, and regret matching a suboptimal regret bound, why?

▶ Computationally cheap! \((\exp)\) is expensive
▶ No parameter tuning (sort of)
▶ Pruning! (some actions have probability zero)
▶ Paper is easier to follow (+ online resources)
Monte-Carlo counterfactual regret minimization

Like pure regret-matching, we can using different types of sampling to our advantage:

▶ **Chance sampling**—sample chance’s strategy
Monte-Carlo counterfactual regret minimization

Like pure regret-matching, we can use different types of sampling to our advantage:

- **Chance sampling**—sample chance’s strategy
- **External sampling**—sample chance and the opponent’s strategy
Monte-Carlo counterfactual regret minimization

Like pure regret-matching, we can use different types of sampling to our advantage:

- **Chance sampling**—sample chance’s strategy
- **External sampling**—sample chance and the opponent’s strategy
- **Outcome sampling**—sample everything!
Monte-Carlo counterfactual regret minimization

Like pure regret-matching, we can using different types of sampling to our advantage:

- **Chance sampling**—sample chance’s strategy
- **External sampling**—sample chance *and* the opponent’s strategy
- **Outcome sampling**—sample everything!
- **Public chance sampling**—sample only jointly observed chance events
Warm starting
(Brown and Sandholm 2016)

If we have a good strategy profile, can we use it to start CFR in a good spot? Yes!

- Play the strategy against itself to compute initial regrets
Warm starting
(Brown and Sandholm 2016)

If we have a good strategy profile, can we use it to start CFR in a good spot? Yes!

► Play the strategy against itself to compute initial regrets
► How long to play it against itself? Depends, just like step-size.
Regret-based pruning
(Brown and Sandholm 2015)

- Observation: our strategy at information sets that we don’t reach doesn’t impact our opponent’s regret
Regret-based pruning
(Brown and Sandholm 2015)

- Observation: our strategy at information sets that we don’t reach doesn’t impact our opponent’s regret
- Idea: play a best response in those information sets
Regret-based pruning
(Brown and Sandholm 2015)

- Observation: our strategy at information sets that we don’t reach doesn’t impact our opponent’s regret
- Idea: play a best response in those information sets
- Following through with the details allows us to prune (i.e., delay updating) these information sets
Safe Endgame Solving

Can we reconstruct the solution to an endgame in isolation? Yes!
▶ The naive first approach does not theoretically work
Safe Endgame Solving

Can we reconstruct the solution to an endgame in isolation? Yes!
- The naive first approach does not theoretically work
- We need to know the counterfactual values to the opponent
Safe Endgame Solving

Can we reconstruct the solution to an endgame in isolation? Yes!

- The naive first approach does not theoretically work
- We need to know the counterfactual values to the opponent
- Create a gadget game, where the opponent can opt out and receive those values
Can we reconstruct the solution to an endgame in isolation? Yes!

- The naive first approach does not theoretically work
- We need to know the counterfactual values to the opponent
- Create a gadget game, where the opponent can opt out and receive those values
- A substantial part of both DeepStack and Libratus
Questions

Thank you! Questions?
kevin.waugh@gmail.com

Tomorrow: Computer Poker Workshop
Thursday morning: Invited Panel on DeepStack and Libratus