Computer Poker Tutorial @ EC 2016, part 2: Equilibrium Computation

Marc Lanctot

Google DeepMind

Jul 25th, 2016
Outline

- Optimization using the sequence form
- Counterfactual Regret (CFR) Minimization
- Selected CFR extensions
- Solving 2-player Heads-up Limit Texas Hold’em
- Two open problems

Slides and references available at:
http://mlanctot.info/ecpokertutorial2016/
Extensive-form Games

Player 1 sees $J\spadesuit J\heartsuit$

Player 1 bets

Player 2 bets

Player 1 calls

... ≫ payoffs received ≫

Information set $I$, represents state of game from player 1’s perspective

A choice, $q = (I, a)$ is an action $a \in A(I)$ taken from a specific $I$

Information sets: $I \in \mathcal{I}$, (context-specific) choices $q = (I, a) \in Q$. 
Defaults

Unless otherwise noted, assume:

- Notation based on [Osborne & Rubinstein ’94]
- Perfect recall
- Mixed (randomized) strategies are used $\sigma$
- Two players: $N = 2$
  - subscript $i$ refers to a player $i$
  - subscript $-i$ refers to the opponent(s) of player $i$
- Zero-sum: for every game outcome $z$, $\sum_{i=1}^{N} u_i(z) = 0$
  - Set of Nash eq. profiles $\{\sigma^*\} \Leftrightarrow$ set of minimax profiles
  - Expected values for eq. $u_i(\sigma^*)$ are unique
  - Nash strategies for player $i$ are interchangeable:
    $u_i(\sigma^*_{i,1}, \sigma^*_i) = u_i(\sigma^*_{i,2}, \sigma^*_i)$
Sequence-Form Representation

Refs: [Koller, Megiddo, von Stengel ’94][von Stengel ’07]

Let $Q_i = \{(I, a) \mid I \in \mathcal{I}, a \in A(I)\}$ be the set of choices for player $i$. 
Let $Q_i = \{(I, a) \mid I \in \mathcal{I}, a \in A(I)\}$ be the set of choices for player $i$.

Encode realization plan for player $i$ using constraints; $\Delta Q := \{x\}$ such that

- $x_i(q_{\emptyset}) = 1$
- $x_i(q) = \sum_{q' \in \text{succ}_i(q)} x_i(q')$
Let $Q_i = \{(I, a) \mid I \in \mathcal{I}, a \in A(I)\}$ be the set of choices for player $i$.

Encode realization plan for player $i$ using constraints; $\Delta Q := \{x\}$ such that

- $x_i(q_\epsilon) = 1$
- $x_i(q) = \sum_{q' \in \text{succ}_i(q)} x_i(q')$
Let $Q_i = \{(I, a) \mid I \in \mathcal{I}, a \in A(I)\}$ be the set of choices for player $i$.

Encode realization plan for player $i$ using constraints; $\Delta Q := $ all $x$ such that

- $x_i(q_\emptyset) = 1$
- $x_i(q) = \sum_{q' \in \text{succ}_i(q)} x_i(q')$

⇒ Much more space-efficient than mixture over pure strategies!
Sequence-Form Linear Programming

For two-player zero-sum, setup an optimization problem:

$$\max_{x \in \Delta Q_1} \min_{y \in \Delta Q_2} xAy = \min_{y \in \Delta Q_2} \max_{x \in \Delta Q_1} xAy,$$

Subject to $Ex = e$, $x \geq 0$, $x \cdot 1 = 1$. 
Sequence-Form Linear Programming

For two-player zero-sum, setup an optimization problem:

\[
\max_{x \in \Delta Q_1} \min_{y \in \Delta Q_2} x A y = \min_{y \in \Delta Q_2} \max_{x \in \Delta Q_1} x A y,
\]

Subject to \(E x = e, x \geq 0, x \cdot 1 = 1\).

Here:

- \(A\) has an entry for every \(q_1\) and \(q_2\) that result in terminal states.
- \(E\) encodes the structure of \(Q_i\)
- \(e\) is \((1, 0, 0, 0, ...)^T\)
Sequence-Form Linear Programming

For two-player zero-sum, setup an optimization problem:

$$\max_{x \in \Delta Q_1} \min_{y \in \Delta Q_2} xAy = \min_{y \in \Delta Q_2} \max_{x \in \Delta Q_1} xAy,$$

Subject to $Ex = e$, $x \geq 0$, $x \cdot 1 = 1$.

Here:
- $A$ has an entry for every $q_1$ and $q_2$ that result in terminal states.
- $E$ encodes the structure of $Q_i$
- $e$ is $(1, 0, 0, 0, \ldots)^T$

Storing $A$ requires $O(|Q_1||Q_2|)$ space in the worst case.
For two-player zero-sum, setup an optimization problem:

\[
\max_{x \in \Delta Q_1} \min_{y \in \Delta Q_2} xAy = \min_{y \in \Delta Q_2} \max_{x \in \Delta Q_1} xAy,
\]

Subject to \(Ex = e, x \geq 0, x \cdot 1 = 1\).

Here:

- \(A\) has an entry for every \(q_1\) and \(q_2\) that result in terminal states.
- \(E\) encodes the structure of \(Q_i\).
- \(e\) is \((1, 0, 0, 0, \ldots)^T\).

Storing \(A\) requires \(O(\|Q_1\|\|Q_2\|)\) space in the worst case.

For large games, in practice, \(A\) is sparse and so memory requirements are closer to \(|Q_1| + |Q_2|\).
Double-Oracle Methods

Refs: [Zinkevich et al. ’06][Bosansky et. al ’14]

In extensive-form:
- Each row/column corresponds to a realization plan
- Heuristics to compute "good" best responses
- Could still require enumerating entire space
Nesterov’s Excessive Gap Technique (EGT)

Refs: [Gilpin et al. ’07][Hoda et al. ’10]

Recall:

\[
\max_{x \in \Delta Q_1} f(y) = \min_{y \in \Delta Q_2} \phi(x),
\]

where \(f(y) = \min_{y \in \Delta Q_2} z\), \(\phi(x) = \max_{x \in \Delta Q_1} z\), \(z = xAy\).

Use a strongly convex function \(d_i\) on \(\Delta Q_i\), and define:

\[
f_{\mu_2}(y) = \min_{y \in \Delta Q_2} \{z + \mu_2 d_2(y)\}
\]

\[
\phi_{\mu_1}(x) = \max_{x \in \Delta Q_1} \{z - \mu_1 d_1(x)\}
\]

Then \(f_{\mu_2}(y) \geq \phi_{\mu_1}(x) \Rightarrow 0 \leq \phi(y) - f(x) \leq \mu_1 d_1^\top + \mu_2 d_2^\top\), so iteratively compute \((x^k, y^k, \mu_1^k, \mu_2^k)\) such that \(\mu_i^{k+1} < \mu_i^k\) by gradient descent.

**Theorem:** EGT computes an \(\epsilon\)-equilibrium in \(O(1/\epsilon)\) iterations.
Counterfactual Regret Minimization (CFR)

CFR [Zinkevich et al. 2008] is iterative strategy-updating algorithm:

\[ t = 1 \]

Player 1 strategies: \( \sigma_1^1 \)
Player 2 strategies: \( \sigma_2^1 \)
Counterfactual Regret Minimization (CFR)

CFR [Zinkevich et al. 2008] is iterative strategy-updating algorithm:

\[

t = 1 \quad \text{to} \quad t = 2
\]

Player 1 strategies:
\[
\sigma_1 \rightarrow \sigma_1
\]

Player 2 strategies:
\[
\sigma_2 \rightarrow \sigma_2
\]

Let \( R_T^i \) be the external regret of using \( \sigma_t \) after \( T \) steps:

\[
R_T^i = \max_{a \in A} E \left[ \sum_{t=1}^{T} (u_i(a, \sigma_t - i) - u_i(\sigma_t^i, \sigma_t - i)) \right]
\]

\[
R_T^i / T \leq \epsilon \Rightarrow \text{the average profile } (\bar{\sigma}_T^1, \bar{\sigma}_T^2) \text{ is a } 2\epsilon \text{-Nash.}
\]

\( \sigma \) is \( \epsilon \)-Nash if a player can do better by switching to \( \sigma' \).

\( \sigma \) is Nash if no player can do better by switching strategies.
Counterfactual Regret Minimization (CFR)

CFR [Zinkevich et al. 2008] is iterative strategy-updating algorithm:

\[ t = 1 \quad t = 2 \quad t = 3 \quad \ldots \]

Player 1 strategies: \( \sigma_1^1 \rightarrow \sigma_1^2 \rightarrow \sigma_1^3 \quad \ldots \)

Player 2 strategies: \( \sigma_2^1 \rightarrow \sigma_2^2 \rightarrow \sigma_2^3 \quad \ldots \)

Let \( R_T^i \) be the external regret of using \( \sigma_t^i \) after \( T \) steps:

\[ R_T^i = \max_{a \in A} E \left[ \sum_{t=1}^T \left( u_i(a, \sigma_t^i - i) - u_i(\sigma_t^i, \sigma_t^{i-1}) \right) \right] \]

\[ R_T^i / T \leq \epsilon \Rightarrow \text{the average profile} \ (\bar{\sigma}_T^1, \bar{\sigma}_T^2) \ \text{is a} \ 2\epsilon \text{-Nash.} \]

\( \sigma \) is \( \epsilon \)-Nash if a player can do \( \epsilon \) better by switching to \( \sigma' \).

\( \sigma \) is Nash if no player can do better by switching strategies.
Counterfactual Regret Minimization (CFR)

CFR [Zinkevich et al. 2008] is iterative strategy-updating algorithm:

\[ t = 1 \rightarrow t = 2 \rightarrow t = 3 \rightarrow \ldots \]

Player 1 strategies: \[ \sigma_1 \rightarrow \sigma_1 \rightarrow \sigma_1 \rightarrow \ldots \]
Player 2 strategies: \[ \sigma_2 \rightarrow \sigma_2 \rightarrow \sigma_2 \rightarrow \ldots \]

Let \( R^T_i \) be the external regret of using \( \sigma^t \) after \( T \) steps:

\[
R^T_i = \max_{a \in A} \mathbb{E} \left[ \sum_{t=1}^{T} (u_i(a, \sigma^t_{-i}) - u_i(\sigma^t_i, \sigma^t_{-i})) \right]
\]

\[
\frac{R^T_i}{T} \leq \epsilon \Rightarrow \text{the average profile} \ (\bar{\sigma}_1^T, \bar{\sigma}_2^T) \ \text{is a} \ 2\epsilon \text{-Nash.}
\]

- \( \sigma \) is \( \epsilon \)-Nash if a player can do \( \epsilon \) better by switching to \( \sigma'_i \).
- \( \sigma \) is Nash if no player can do better by switching strategies.
An extensive-form game is represented in tree form.

Example:

![Game Tree](image-url)
An extensive-form game is represented in tree form.

- $h \in H$ is possible history;
- $z \in Z, Z \subseteq H$ is a terminal history.

**Example:**

```
+1 +2 −2 +1
+1 −1
b a
c d c d
fe f
```

```
P1
a b
c d
c1 c2

P2
c d
e f
c d
e f
```

```
Chance

+1 +3
+1 +2 −1
−1 +2 −2
```

```
P1
a
b
c
```
An **extensive-form game** is represented in tree form.

- $h \in H$ is possible history; $z \in Z, Z \subseteq H$ is a terminal history.

- An information set $I_i \in \mathcal{I}$ is an information set for player $i$.

**Example:**

```
         *  
     c₁   c₂  
     /     /  
P1     P1  
     /     /  
   a     a  
  /     /  
P2     P2  
 /     /  
     b     b  
     /     /  
   c     c  
  /     /  
     d     d  
     /     /  
   e     e  
  /     /  
     f     f  
     /     /  
   e     e  
      |     |  
     +1   +1  
     +2   +2  
     -2   -2  
     +1   +1  
```

$P1$'s action set: $A(I_1) = \{a, b\}$.

$P1$'s action set: $A(I_2) = \{c, d\}$.

$P2$'s action set: $A(I_1') = \{c, d\}$.

$P2$'s action set: $A(I_2') = \{e, f\}$.
An **extensive-form game** is represented in tree form.

- $h \in H$ is possible history;
- $z \in Z, Z \subseteq H$ is a terminal history.

An information set $I_i \in \mathcal{I}$ is an information set for player $i$.

$A(I_i)$ is the action set for $i$ at information set $I_i$. 

**Example:**

```
Chance
P1
  a
  b
  a
  b
P2
  c
  d
  c
  d

I_1
I_2
I'_2
I'
```

```
-1 +3
+1 +2 −2 +1
+1 −1
c d c d
fe e f
P1
Chance
P2
I'2
I2
I1
a a b b
```
A strategy $\sigma_i \in \Sigma_i$ is a distribution from $I_i \rightarrow A(I_i)$. 

A strategy $\sigma_{-i}$ is a strategy for the opponents of $i$ and chance.

A strategy profile $\sigma = (\sigma_1, \sigma_2)$.

$u_i(z)$ is the payoff to player $i$ when players play $z$.

$\pi_{\sigma}(h)$ is a product of probabilities along history $h$.

$\pi_{\sigma_i}(h)$ is player $i$'s contribution.
A strategy $\sigma_i \in \Sigma_i$ is a distribution from $I_i \to A(I_i)$.

- A strategy $\sigma_{-i}$ is a strategy for the opponents of $i$ and chance.
- A strategy profile $\sigma = (\sigma_1, \sigma_2)$. 

\[
\begin{align*}
\text{P1} & \quad \text{P2} \\
& \quad \text{Chance} \\
\text{I}_2 & \quad \text{I}_1 \\
\text{I}_2 & \quad \text{I}_1 \\
\text{I}_1 & \quad \text{I}_1 \\
\end{align*}
\]
A strategy $\sigma_i \in \sum_i$ is a distribution from $I_i \rightarrow A(I_i)$.

A strategy $\sigma_{-i}$ is a strategy for the opponents of $i$ and chance.

A strategy profile $\sigma = (\sigma_1, \sigma_2)$.

$u_i(z)$ is the payoff to player $i$ when players play $z$. 

\begin{align*}
\text{Chance} \\
P1 & \quad \bullet \\
\text{I}_1 & \quad \text{I}_2 \\
\text{P2} & \quad \bullet \\
\text{I}'_1 & \quad \text{I}'_2
\end{align*}

\begin{align*}
\pi_{\sigma_i}(h) & \quad \text{is a product of probabilities along history } h. \\
\pi_{\sigma_i}(h) & \quad \text{is player } i\text{'s contribution.}
\end{align*}
A strategy $\sigma_i \in \Sigma_i$ is a distribution from $I_i \rightarrow A(I_i)$.

A strategy $\sigma_{-i}$ is a strategy for the opponents of $i$ and chance.

A strategy profile $\sigma = (\sigma_1, \sigma_2)$.

$u_i(z)$ is the payoff to player $i$ when players play $z$.

$\pi^\sigma(h)$ is a product of probabilities along history $h$. $\pi_i^\sigma(h)$ is player $i$’s contribution.
CFR Algorithm (Overview)

Refs: [Zinkevich et al. ’08][Hart & Mas-Colell ’00]

1. Minimize **average immediate counterfactual regret** $R_{i,\text{imm}}^T(I)$

2. **Theorem 3**: Overall regret bounded by

   $$\frac{R_i^T}{T} \leq \sum_{I \in \mathcal{I}_i} R_{i,\text{imm}}^T(I)$$

3. **Theorem 4**: Using regret-matching to update strategies, $\sigma^t$ at each information set, then

   $$\frac{R_i^T}{T} \leq \frac{\Delta_{u,i} |\mathcal{I}_i| \sqrt{|A_i|}}{\sqrt{T}}$$

   where $\Delta_{u,i}$ is a payoff range for $i$. 
CFR Algorithm (Example)

Define **counterfactual value** as

\[ v_i(\sigma, I) = \sum_{h \in I, z \in Z} \pi_{-i}(h)\pi^\sigma(h, z)u_i(z) \]

Define \( v_i(\sigma_{(I\rightarrow a)}, I) \) similarly, except take \( a \) at \( I \)
CFR Algorithm (Example)

Define **counterfactual value** as

\[ v_i(\sigma, I) = \sum_{h \in I, z \in Z} \pi^\sigma_i(h) \pi^\sigma(h, z) u_i(z) \]

Define \( v_i(\sigma_{(I \rightarrow a)}, I) \) similarly, except take \( a \) at \( I \)

Repeat until sufficiently small \( \epsilon \):

1. Walk the game tree computing

\[ r(I, a) = v_i(\sigma_{(I \rightarrow a)}, I) - v_i(\sigma, I) \]
CFR Algorithm (Example)

Define **counterfactual value** as

\[ v_i(\sigma, I) = \sum_{h \in I, z \in Z} \pi^\sigma_i(h) \pi^\sigma(h, z) u_i(z) \]

Define \( v_i(\sigma(I \rightarrow a), I) \) similarly, except take \( a \) at \( I \)

Repeat until sufficiently small \( \epsilon \):

1. Walk the game tree computing
   \[ r(I, a) = v_i(\sigma(I \rightarrow a), I) - v_i(\sigma, I) \]
   - Recursively compute \( r(I, a) \) at a particular node
   - Add to accumulated values
     \[ r[I, a] += r(I, a) \]

\[ \begin{align*}
  a &= .4 \\
  b &= .6 \\
  z &= \text{Player} \\
  u &= 2.2 \\
  v &= .33((.4)(2.2) + (.6)(1.9)) \\
      &= 0.66 \\

  c_1 &= .33 \\
  c_2 &= .66 \\
\end{align*} \]
**CFR Algorithm (Example)**

Define **counterfactual value** as

\[
v_i(\sigma, I) = \sum_{h \in I, z \in Z} \pi_{\sigma_i}(h) \pi(\sigma(h, z)) u_i(z)
\]

Define \(v_i(\sigma(I \rightarrow a), I)\) similarly, except take \(a\) at \(I\)

Repeat until sufficiently small \(\epsilon\):

1. **Walk the game tree computing**
   \[r(I, a) = v_i(\sigma(I \rightarrow a), I) - v_i(\sigma, I)\]
   - Recursively compute \(r(I, a)\) at a particular node
   - Add to accumulated values \(r[I, a] = r(I, a)\)

\[c_1 = .33 \quad c_2 = 0.66\]

\[a = .4 \quad b = .6\]

\[u = 1.9 \quad v = 0.33((0.4)(2.2)+(0.6)(1.9)) = 0.66\]

\[r[I,a] = 0.33(0.4)(2.2) - 0.66\]
Define **counterfactual value** as

\[ v_i(\sigma, I) = \sum_{h \in I, z \in Z} \pi^\sigma_i(h) \pi^\sigma(h, z) u_i(z) \]

Define \( v_i(\sigma(I \rightarrow a), I) \) similarly, except take \( a \) at \( I \)

**Repeat until sufficiently small \( \epsilon \):**

1. **Walk the game tree computing** \( r(I, a) = v_i(\sigma(I \rightarrow a), I) - v_i(\sigma, I) \)
   1. Recursively compute \( r(I, a) \) at a particular node
   2. Add to accumulated values \( r[I, a] += r(I, a) \)

\[
\begin{align*}
  v & = 0.33((0.4)(2.2)+(0.6)(1.9)) \\
  & = 0.66 \\
  r[I,a] & += 0.33(.4)(2.2) - 0.66 \\
  r[I,b] & += 0.33(.6)(1.9) - 0.66
\end{align*}
\]

\( u = 1.9 \)
Define **counterfactual value** as

\[ v_i(\sigma, I) = \sum_{h \in I, z \in Z} \pi_{-i}^\sigma(h) \pi^\sigma(h, z) u_i(z) \]

Define \( v_i(\sigma(I \to a), I) \) similarly, except take \( a \) at \( I \)

Repeat until sufficiently small \( \epsilon \):

1. Walk the game tree computing 
   \( r(I, a) = v_i(\sigma(I \to a), I) - v_i(\sigma, I) \)
   - Recursively compute \( r(I, a) \) at a particular node
   - Add to accumulated values 
     \( r[I, a] += r(I, a) \)
CFR Algorithm (Example)

Define **counterfactual value** as

\[ v_i(\sigma, I) = \sum_{h \in I, z \in Z} \pi_{-i}(h) \pi^\sigma(h, z) u_i(z) \]

Define \( v_i(\sigma(I \rightarrow a), I) \) similarly, except take \( a \) at \( I \)

Repeat until sufficiently small \( \epsilon \):

1. Walk the game tree computing
   \[ r(I, a) = v_i(\sigma(I \rightarrow a), I) - v_i(\sigma, I) \]
   1. Recursively compute \( r(I, a) \) at a particular node
   2. Add to accumulated values
      \[ r[I, a] += r(I, a) \]

\[
\begin{align*}
v &= 0.66(0.4)(-0.6) + (0.6)(0.7) \\
&= 0.1188 \\
r[I,a] &+ 0.66(0.4)(-0.6) - 0.1188 \\
v &= -0.6 \\
u &= 0.7
\end{align*}
\]
Define **counterfactual value** as

\[
v_i(\sigma, I) = \sum_{h \in I, z \in Z} \pi^\sigma_i(h) \pi^\sigma(h, z) u_i(z)
\]

Define \(v_i(\sigma(I \rightarrow a), I)\) similarly, except take \(a\) at \(I\)

Repeat until sufficiently small \(\epsilon\):

1. Walk the game tree computing \(r(I, a) = v_i(\sigma(I \rightarrow a), I) - v_i(\sigma, I)\)
   1. Recursively compute \(r(I, a)\) at a particular node
   2. Add to accumulated values \(r[I, a] += r(I, a)\)
CFR Algorithm (Example)

Define **counterfactual value** as

\[ v_i(\sigma, I) = \sum_{h \in I, z \in Z} \pi_{-i}(h) \pi^\sigma(h, z) u_i(z) \]

Define \( v_i(\sigma_{(I\to a)}, I) \) similarly, except take \( a \) at \( I \)

Repeat until sufficiently small \( \epsilon \):

1. Walk the game tree computing \( r(I, a) = v_i(\sigma_{(I\to a)}, I) - v_i(\sigma, I) \)
   1. Recursively compute \( r(I, a) \) at a particular node
   2. Add to accumulated values \( r[I, a] += r(I, a) \)
2. \( \sigma_i^{t+1}(I) \leftarrow \text{RegretMatching}(r[I]) \)
CFR Algorithm (Example)

Define **counterfactual value** as

\[ v_{i}(\sigma, I) = \sum_{h \in I, z \in Z} \pi_{-i}(h) \pi^{\sigma}(h, z) u_{i}(z) \]

Define \( v_{i}(\sigma(I \rightarrow a), I) \) similarly, except take \( a \) at \( I \)

Repeat until sufficiently small \( \epsilon \):

1. Walk the game tree computing \( r(I, a) = v_{i}(\sigma(I \rightarrow a), I) - v_{i}(\sigma, I) \)
   1. Recursively compute \( r(I, a) \) at a particular node
   2. Add to accumulated values \( r[I, a] += r(I, a) \)
2. \( \sigma_{i}^{t+1}(I) \leftarrow \text{RegretMatching}(r[I]) \)
3. Update average profile \( \bar{\sigma} \)
CFR Extension Outline

Many (20+ !) follow-up papers on CFR.

I will cover a subset:

- Restricted Nash Responses
- Monte Carlo CFR
- Imperfect Recall Abstraction
- Multiplayer and non-zero-sum
- Sequence-Form Replicator Dynamics
- CFR-BR
- CFR+
Restricted Nash Responses

Game $G$ is some game.

$G^R$ is a restricted copy (e.g. player $-i$ plays $\sigma_{fixed}$)

$\text{Nash}_i(G') \Leftrightarrow$ best trade-off between $\text{Nash}_i(G)$ and $\text{BR}_i(\sigma_{fixed})$
Monte Carlo CFR

Refs: [Lanctot et al. '09], [Gibson et al. '12], [Johanson et al. '12], [Burch et al. '12]

Sample parts of the tree: **sampled counterfactual values** $\tilde{v}_i(\sigma, I)$.

Unbiased estimator: $\mathbb{E}[\tilde{v}_i(\sigma, I)] = v_i(\sigma, I)$.

**Theorem**: with probability $1 - p$, $\delta$ is $i$’s min prob sampling $z$

$$R_i^T / T \leq \left( M_i(\sigma_i^*) \sqrt{|\max_I A(I)|} + \frac{\sqrt{2|I_i||B_i|}}{\sqrt{p}} \right) \left( \frac{1}{\delta} \right) \left( \frac{\Delta_{u,i}}{\sqrt{T}} \right)$$

E.g. chance sampling $\rightarrow$ sample only chance outcomes
Monte Carlo CFR: External Sampling

Refs: [Lanctot et. al ’09][Gibson ’14]

Theorem: with prob $1 - p$:

$$R_i^T / T \leq \left( M_i(\sigma_i^*) \sqrt{\max_I A(I)} + \frac{\sqrt{2|I_i||B_i|}}{\sqrt{p}} \right) \left( \frac{\Delta_{u,i}}{\sqrt{T}} \right)$$

- Has worked well in (> 2)-player and large action spaces
- Tartanian7, 2014 winner of 2P NL, used variant of ext. sampling
Monte Carlo CFR: Public Chance Sampling

Refs: [Johanson et. al ’11][Jackson ’12]

Sample only *public* chance events!
Monte Carlo CFR: Public Chance Sampling

Refs: [Johanson et. al ’11][Jackson ’12]

Sample only *public* chance events!

Vectorize the tree walk (one element per opponent private card)
Monte Carlo CFR: Public Chance Sampling

Refs: [Johanson et. al ’11][Jackson ’12]

Sample only *public* chance events!

Vectorize the tree walk (one element per opponent private card)

Same bound as E.S. but can use equiv. classes at leaf nodes!
Monte Carlo CFR: Public Chance Sampling

Refs: [Johanson et. al ’11][Jackson ’12]

Sample only *public* chance events!

Vectorize the tree walk (one element per opponent private card)

Same bound as E.S. but can use equiv. classes at leaf nodes!

![Graph showing best response over time for Liar’s Dice (2,2)]

Liar’s Dice (2,2)

Slumbot, 2012 winner of HULHE, used PCS
Generalized Monte Carlo CFR

Refs: [Gibson et al. '12]

Given any estimator for counterfactual values $\hat{v}(I, a)$ with bounded range $\hat{\Delta}$:

**Theorem:** with prob $1 - p$,

$$
\frac{R_T}{T} \leq |I_i| \left( \frac{\hat{\Delta}_i \sqrt{\max_I |A(I)|}}{\sqrt{T}} + \sqrt{\frac{\text{Var}}{pT} + \frac{\text{Cov}}{p} + \frac{E^2}{p}} \right),
$$

where:

- **Var** is max variance of diff in regret and est. regret at $t$,
- **Cov** is max covariance of diff in regret and est. regret at $t, t'$,
- **E** is the max expectation of diff in regret and est. regret (bias) at $t$, over all time steps $t$ (and $t'$), info sets $I$, actions $a \in A(I)$. 
Imperfect Recall Abstraction


History $h \in \tilde{I}$, define $X_i(h) = (\tilde{I}_1, a_1), (\tilde{I}_2, a_2), \cdots$ as player $i$’s choice sequence for all $I_k$ belonging to $i$ in $h$. 
Imperfect Recall Abstraction


History $h \in \tilde{I}$, define $X_i(h) = (\tilde{I}_1, a_1), (\tilde{I}_2, a_2), \cdots$ as player $i$’s choice sequence for all $I_k$ belonging to $i$ in $h$.

Perfect recall: for all $h, h' \in \tilde{I} \iff X_i(h) = X(h')$
Imperfect Recall Abstraction


History $h \in \tilde{I}$, define $X_i(h) = (\tilde{I}_1, a_1), (\tilde{I}_2, a_2), \cdots$ as player $i$’s choice sequence for all $I_k$ belonging to $i$ in $h$.

Perfect recall: for all $h, h' \in \tilde{I} \iff X_i(h) = X(h')$

Not every $h \in \tilde{I}$ is necessarily relevant for computing approx. $\sigma(\tilde{I})!!$
Imperfect Recall Abstraction


History $h \in \tilde{I}$, define $X_i(h) = (\tilde{I}_1, a_1), (\tilde{I}_2, a_2), \cdots$ as player $i$’s choice sequence for all $I_k$ belonging to $i$ in $h$.

Perfect recall: for all $h, h' \in \tilde{I} \iff X_i(h) = X(h')$

Not every $h \in \tilde{I}$ is necessarily relevant for computing approx. $\sigma(\tilde{I})$!!

Purposely forget parts of $h \in \tilde{I}$ and $h' \in \tilde{I}'$; $\rightarrow$ merge $I = \tilde{I} \cup \tilde{I'}$. 
Imperfect Recall Abstraction


History $h \in \mathcal{I}$, define $X_i(h) = (\mathcal{I}_1, a_1), (\mathcal{I}_2, a_2), \cdots$ as player $i$’s choice sequence for all $I_k$ belonging to $i$ in $h$.

Perfect recall: for all $h, h' \in \mathcal{I} \iff X_i(h) = X(h')$

Not every $h \in \mathcal{I}$ is necessarily relevant for computing approx. $\sigma(\mathcal{I})$!!

Purposely *forget* parts of $h \in \mathcal{I}$ and $h' \in \mathcal{I}'$; $\rightarrow$ merge $I = \mathcal{I} \cup \mathcal{I}'$.

Benefits:

1. Huge savings in memory
Imperfect Recall Abstraction


History $h \in \tilde{I}$, define $X_i(h) = (\tilde{I}_1, a_1), (\tilde{I}_2, a_2), \cdots$ as player $i$’s choice sequence for all $I_k$ belonging to $i$ in $h$.

Perfect recall: for all $h, h' \in \tilde{I} \iff X_i(h) = X(h')$

Not every $h \in \tilde{I}$ is necessarily relevant for computing approx. $\sigma(\tilde{I})!!$

Purposely forget parts of $h \in \tilde{I}$ and $h' \in \tilde{I}'$; $\rightarrow$ merge $I = \tilde{I} \cup \tilde{I}'$.

Benefits:

1. Huge savings in memory
2. Often clear what should be forgotten
Imperfect Recall Abstraction


History $h \in \mathcal{I}$, define $X_i(h) = (\bar{I}_1, a_1), (\bar{I}_2, a_2), \cdots$ as player $i$’s choice sequence for all $I_k$ belonging to $i$ in $h$.

Perfect recall: for all $h, h' \in \mathcal{I} \iff X_i(h) = X(h')$

Not every $h \in \mathcal{I}$ is necessarily relevant for computing approx. $\sigma(\mathcal{I})$!!

Purposely *forget* parts of $h \in \mathcal{I}$ and $h' \in \mathcal{I}'$; $\rightarrow$ merge $I = \mathcal{I} \cup \mathcal{I}'$.

Benefits:

1. Huge savings in memory
2. Often clear what should be forgotten
3. CFR algorithm still runs(!)
   - But does it still work/converge?
   - In theory: yes! Under some (somewhat restrictive) assumptions.
   - In practice: yes, very well!
Multi (> 2) player and non-zero sum

Refs: [Abou Risk & Szafron ’10][Gibson & Szafron ’11][Gibson et al. ’13][Gibson ’14]

Generally not much known about CFR in this case.

But here again, algorithm is still well-defined.

Gibson 2014:

- Regret min. removes iteratively strictly-dominated strategies.
- Extend to *dominated actions* and counterfactual values.
- CFR removes iterative strictly-dominated actions.
- 2-player game: If $R^T_i / T < \epsilon$, converges to $2(\epsilon + \delta_u)$-Nash.

Sequence-Form Replicator Dynamics

Refs: [Gatti et al. ’13][Lanctot ’14]

Recall $Q$ set of choices $(I, a)$, and $x_i(q)$ realization weight on $q$:

For player $i$, for each $q \in Q_i$, update:

$$x_i(q, t + 1) = x_i(q, t) \frac{u_i(x_i \rightarrow g_q)}{u_i(x)}$$
Sequence-Form Replicator Dynamics

Refs: [Gatti et al. ’13][Lanctot ’14]

Recall $Q$ set of choices $(I, a)$, and $x_i(q)$ realization weight on $q$:

For player $i$, for each $q \in Q_i$, update:

$$x_i(q, t + 1) = x_i(q, t) \frac{u_i(x_i \rightarrow g_q)}{u_i(x)}$$

$x_i \rightarrow g_q$ is $x$

except player $i$

uses $g_q(x_i)$
Recall $Q$ set of choices $(I, a)$, and $x_i(q)$ realization weight on $q$:

For player $i$, for each $q \in Q_i$, update:

$$x_i(q, t + 1) = x_i(q, t) \frac{u_i(x \rightarrow g_q)}{u_i(x)}$$

$$g_q(x_i, q') = \begin{cases} 
1 & \text{if } q' \in X_i(q), \\
\frac{x_i(q')}{\text{Ancestor}(q, q')} & \text{if } X_i(q) \sqsubseteq X_i(q'), \\
0 & \text{otherwise},
\end{cases}$$
Sequence-Form Replicator Dynamics

Refs: [Gatti et al. '13][Lanctot '14]

For player $i$, for each $q \in Q_i$, update:

$$x_i(q, t + 1) = x_i(q, t) \frac{u_i(x_i \rightarrow g_q)}{u_i(x)}$$

except player $i$ uses $g_q(x_i)$

$$g_q(x_i, q') = \begin{cases} 
1 & \text{if } q' \in X_i(q), \\
\frac{x_i(q')}{\text{Ancestor}(q, q')} & \text{if } X_i(q) \sqsubseteq X_i(q'), \\
0 & \text{otherwise,}
\end{cases}$$

- $g_q(x_i)$ is a “projection”: $i$ plays $q$ if possible, else plays $x_i$
- Implements a form of counterfactual regret minimization
- In 3-player Kuhn poker, finds ”best” equilibrium!
Minimize regret against a best responder
Best responder uses *full unabstracted space*
Use accelerated algorithms for computing best response
Used in diabetes patient simulation [Chen & Bowling ’12]
Refs: [Tammelin et al. ’11]

Regret matching plus (RM$^+$): never accumulate negative regret!
Refs: [Tammelin et al. ’11]

Regret matching plus (RM$^+$): never accumulate negative regret!

**Theorem 1:** $T$ steps: RM$^+$ has external regret $\Delta_u \sqrt{|A|T}$.
Regret matching plus (RM\(^+\)): never accumulate negative regret!

**Theorem 1:** \( T \) steps: RM\(^+\) has external regret \( \Delta_u \sqrt{|A|T} \).

Tracking regret [Herbster & Warmuth ’98]: hindsight strategy can change \((k - 1)\) times.

**Theorem 2:** \( T \) step: RM\(^+\) has tracking regret \( k\Delta_u \sqrt{|A|T} \).

**Theorem 3:** \( T \) step: CFR\(^+\) has regret \( O(|I_1| + |I_2|) \sqrt{|A|T} \).
Solving 2-player HULHE

Refs: [Bowling et al. ’15]
Solving 2-player HULHE

Refs: [Bowling et al. ’15]
Solving 2-player HULHE

Refs: [Bowling et al. ’15]

Legend: fold raise call
Open Problem #1: Stronger-than-Nash?

Can a new variant of CFR converge to a:

- Sequential equilibrium?
- Trembling-hand perfect equilibrium?
- Strong equilibrium?
Does/can CFR converge to an (extensive-form) correlated equilibrium?
Other work

- FSICFR (chance-sampling variant) [Neller & Hnath ’11]
- CFR with decomposition [Burch et al. ’12][Jackson ’14]
- Regret transfer [Brown and Sandholm ’14]
- Regret-based Pruning [Brown and Sandholm ’14]
- Automated abstraction and solving [Brown and Sandholm ’15]
- Warm starting CFR [Brown and Sandholm ’16]
- Online search [Lisý, Lanctot, and Bowling ’15][Heinrich & Silver ’15]
- Relationship to optimization [Waugh and Bagnell ’15]
- Fictitious Self-play [Heinrich, Lanctot, and Silver ’15]
- End-to-end learning [Waugh et al. ’15][Heinrich and Silver ’16]
- Application to security domains [Lisy, Davis, and Bowling ’16]
- ...
Thanks, Questions, Info

Thank you for listening! Any questions?

Part 1: Sam Ganzfried
sam.ganzfried@gmail.com
http://www.ganzfriedresearch.com

Part 2: Marc Lanctot
marc.lanctot@gmail.com
http://mlanctot.info
http://mlanctot.info/ecpokertutorial2016