Regret and Poker

AI For Traditional Games
Prof. Nathan Sturtevant
Winter 2013

UNIVERSITY OF DENVER

(One possible) Formal definition of regret

- Let $u^t(a)$ be the utility (payoff) of the action taken at step t
- Let $\sigma^t(a)$ be the probability of taking action a
  - Recall that a strategy is written as σ (Lecture 3)
- Internal regret at time T of not taking action a, R^T_a, is:

$$\sum_{t=1}^{T} (u^t(a) - \sum_b \sigma^t(b) u^t(b))$$

- The second part is the actual payoff returned
- The first part is the payoff that we would get if we could swap the action we took with the action A

Another regret-minimizing algorithm

- Define a policy at information set I and time T+1 as:

$$\sigma_i^{T+1}(I)(a) = \begin{cases} \frac{R_i^{T,+}(I,a)}{\sum_{a \in A(I)} R_i^{T,+}(I,a)} & \text{if } \sum_{a \in A(I)} R_i^{T,+}(I,a) > 0 \\ \frac{1}{|A(I)|} & \text{otherwise.} \end{cases}$$

- Where |A(I)| is the number of actions at information set I
- The + notation is max(R(), 0); eg non-negative values
- This produces a mixed strategy
  - Try it on R/P/S with 2 players
- If you "train" for 2 players, the average strategy will converge to e-Nash

What about large trees?

- 2-player limit texas hold 'em
  - 52 choose 2 cards for us
  - 50 choose 2 cards for opponent
  - 48 choose 3 on flop
  - 45 on turn
  - 44 on river
- Between each of these must have strategy for play
  - b / c / f (limit of 3/4 raises per betting round)

UNIVERSITY OF DENVER

Poker

- Strategy for the whole game is too large to learn as a whole strategy
  - But, strategy can be decomposed to each “independent” decision in the game
  - At each information set, can decide how to act relatively independently of the whole game
- Key insight: The regret of a whole strategy is bounded by the sum of regret at each action set

UNIVERSITY OF DENVER

CFR

- Build a ii tree for both players
- Repeat until converged:
  - Sample a perfect-information world
  - Traverse the tree and compute the regret for each action, and update the action probabilities
- The AVERAGE strategy played during training will be a N.E.

UNIVERSITY OF DENVER

CFR traversal

- Compute π -- probability of reaching this state by others players (incrementally updated as you move down the tree)
- Apply each action and recursively compute expected value of each move
  - multiply by σ and sum over all moves to get the payoff of current strategy
- The difference between the value of the move and the payoff is the regret

UNIVERSITY OF DENVER

CFR: Traversing details

- At each state, maintain the average immediate regret:

$$R^T_{i,imm}(I,a) = \frac{1}{T}\sum_{t=1}^{T} \pi^{\sigma^t}_{-i}(I)\left(u_i(\sigma^t|_{I\to a}, I) - u_i(\sigma^t, I)\right)$$

- π is the probability of reaching information set I given the strategies at time *t* and that player *i* is trying to reach that information set
- *u* is the utility of a strategy, possibly given that we switch our action only at this information set
  - Computing this requires traversing the underlying tree
  - Also requires knowing both players strategies

UNIVERSITY OF DENVER

CFR Strategy computation

- The immediate regret informs how we act at each time step
- The average strategy (defined by imm. regret) over time will be a N.E.
- The average strategy is:

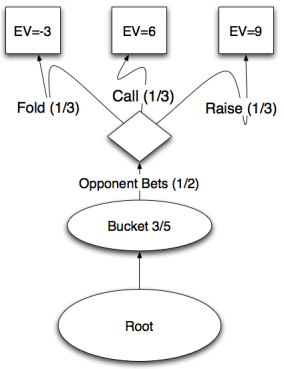
$$\bar{\sigma}_i^T(I)(a) = \frac{\sum_{t=1}^{T} \pi_i^{\sigma^t}(I)\sigma^t(I)(a)}{\sum_{t=1}^{T} \pi_i^{\sigma^t}(I)}.$$

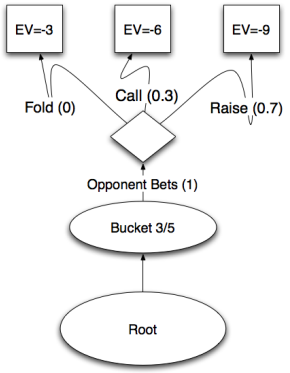
- Note that this uses π_i, not π_{-i}
  - That is, a different weighting than the imm. regret

UNIVERSITY OF DENVER

Example

- (From Michael Johanson's Msc thesis)





UNIVERSITY OF DENVER

Designing for poker

- Poker is too large to solve with CFR
  - Build abstract game instead
- Translate state/moves from real game into abstract game
- Make action in abstract game solved by CFR
- Take actions from abstract game and convert back into real game

UNIVERSITY OF DENVER

Poker abstractions

- Suit isomorphisms (not really an abstraction)
- Only two things to abstract:
  - States, Actions
- Actions:
  - Restrict the number of bets
  - If someone bets more than we can handle, just call

UNIVERSITY OF DENVER

Abstracting state

- Simple example from Kuhn poker:
  - Can have 1, 2 or 3
  - Can ‘merge’ information sets together to reduce number of states
    - Can do so asymmetrically if necessary
  - eg first player cannot distinguish 2 from 3
  - eg both players cannot distinguish 1 from 3

UNIVERSITY OF DENVER

Larger scale abstractions

- Group (bucket) hands together that are similar
  - The best way to do this is somewhat an open question
- Example grouping metrics:
  - Strength of the hand (chances of winning)
  - Potential of hand (how likely is it to improve)
- Ideally, hands with the same betting sequences should be abstracted together

UNIVERSITY OF DENVER

More abstraction

- Flop/turn/river increase the size of the game tree
- Don’t represent the cards that come down
  - Instead, just represent a transition between buckets
- What problems does this create?
  - Program doesn’t know what cards it has
  - Doesn’t know if it has the ‘nuts’ (best hand possible)

UNIVERSITY OF DENVER

What is the cost of abstraction?

- Are larger abstractions strictly better?
  - eg if my abstract game is closer to the real game, will my program play the game better?
- Yes & No
  - Abstraction is pathological; the size of the abstraction doesn’t predict the performance of the best response
  - BUT, bigger abstractions make less dominated errors
    - So, in practice, they are likely to perform better