R-MAX - A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning

Ronen I. Brafman and Moshe Tennenholtz

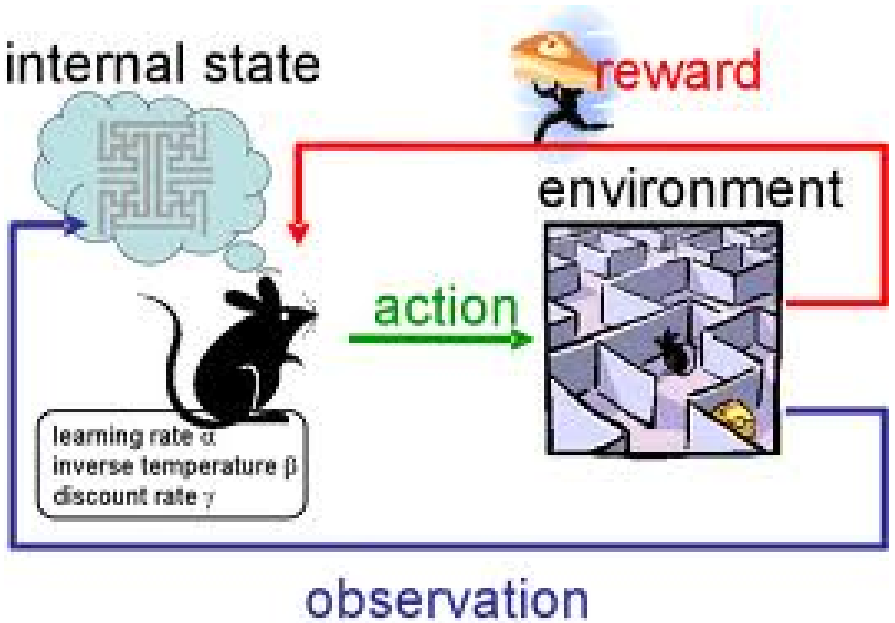
presented by Janyl Jumadinova
January 11, 2013

Overview

- A model-based reinforcement learning algorithm is presented for stochastic games
- It improves upon previous reinforcement learning algorithms
- Optimality and convergence are discussed and proved
- Simplified version of the proposed algorithm is given for repeated games
- No experimental results are presented

Reinforcement learning

Agent interacts with unknown environment and tries to choose actions that maximize its cumulative payoff



Reinforcement learning algorithms

- E^3 algorithm (Kearns and Singh, 1998)
  - first provably near-optimal polynomial time algorithm for learning in Markov decision processes (MDPs)
- Agent learns by updating its environment's model using statistics it collects as long as it can be done efficiently
- Then agent uses its learned model to compute the optimal policy
- E^3 handles exploration-exploitation trade-off explicitly
- Therefore, it is difficult to generalize it to multi-agent settings

Reinforcement learning algorithms

- E^3 was extended to single controller stochastic games (SCSG) - **LSG algorithm** (Brafman and Tennenholtz, 2000)
  - one player controls the transition

- E^3 was extended to **structured MDPs** (Kearns and Koller, 1999)
  - MDPs whose transition model can be factored as a dynamic Bayesian network (DBN)

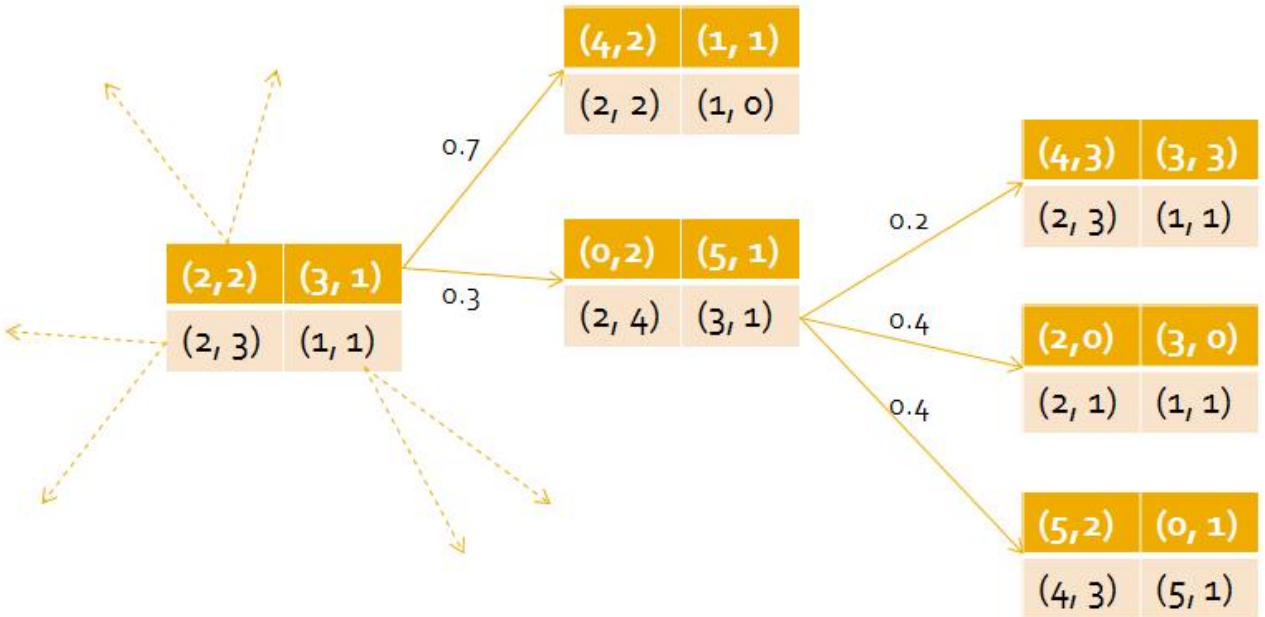
R-MAX algorithm: Summary

- R-MAX generalizes E^3 to adversarial context
- Agent using R-MAX always attempts to optimize its behavior w.r.t. fictitious model
  - it is always either optimal or it leads to efficient learning
  - agent usually doesn't know whether its optimizing or learning $\rightarrow$ it will exploit or explore efficiently
  - polynomial number of parameters to learn
  - if learning is done efficiently, can ensure that agent spends polynomial number of steps exploring and the rest of the time exploiting

R-MAX algorithm: Summary

- Exploration-exploitation trade-off is implicit
- Converges in polynomial time to a near-optimal solution
- R-MAX uses zero-sum stochastic game model $\rightarrow$ more general than MDP
- Approach is not new - "optimism in the face of uncertainty heuristic"
  - When faced with the choice between a known and unknown reward, always try the unknown reward

Stochastic Games (SGs)



- 2 player, fixed-sum (constant-sum) games
- Person under our control - agent
- The other player - adversary
- Assume that the size of the action set of both agent and adversary are the same

Stochastic Games (SGs)

- Players play a sequence of games from some given set of games
- After playing each game, the players receive the appropriate payoff and move to a new game
- The new game depends on the previous games and the players' actions in it
- Payoffs are normalized between 0 and a constant R_{max}
- Number of actions is constant

Stochastic Games (SGs)

Definition 1 *A* fixed-sum, two player, stochastic-game [SG] M *on states* $S = \{1, \ldots, N\}$, *and actions* $A = \{a_1, \ldots, a_k\}$, *consists of:*

- **Stage Games:** *each state* $s \in S$ *is associated with a two-player, fixed-sum game in strategic form, where the action set of each player is* A. *We use* R^i *to denote the reward matrix associated with stage-game* i.
- **Probabilistic Transition Function:** $P_M(s, t, a, a')$ *is the probability of a transition from state* s *to state* t *given that the first player (the* agent*) plays* a *and the second player (the* adversary*) plays* a'.

Stochastic Games: notation

- Set of possible histories of length t is $(S \times A^2 \times R)^t \times S$
- The set of possible histories, H, is the union of the sets of possible histories $\forall t \geq 0$
- Policy, $\pi : (S \times A^2 \times R)^t \times S \rightarrow A$
- Value of the policy π is $U_M(s, \pi, \rho, T)$ - expected average reward if agents follows π for T steps
  - M is SG, T is a natural number
  - ρ is adversary's policy
- $U_M(s, \pi, T) = min_{\rho \text{ is a policy}} U_M(s, \pi, \rho, T)$
- $U_M(s, \pi) = lim\ inf_{T \rightarrow \infty} U_M(s, \pi, T)$
- $U_M(\pi) = min_{s \in S} U_M(s, \pi)$

Mixing time

- Kearns and Singh (1998) defined *ϵ-return mixing time* of π as the smallest value of T after which π guarantees an expected payoff of at least $U(\pi) - \epsilon$
- Here, the authors adjust this definition slightly to account for adversary

Policy π belongs to the set $\Pi(\epsilon, T)$ with *ϵ-return mixing time* at most T if for any starting state s, adversary policy ρ, and $\forall t \geq T$, $U(s, \pi, \rho, T) \geq U(\pi) - \epsilon$

- $Opt(\Pi(\epsilon, T))$ - optimal expected T-step undiscounted average return among policies in $\Pi(\epsilon, T)$

Assumptions

1. Agent always knows what state it is in
2. Agent knows what actions were taken by its adversary and what payoffs were obtained after each stage
3. Maximal possible reward, R_{max}, is known apriori
4. ϵ-return mixing time of any policy is known

The R-MAX Algorithm

- Stochastic game M consists of a set $S = \{G_1, ..., G_N\}$ of stage games
- R^i - reward matrix for game i
- $R^i_{m,l}$ - pair consisting of agent's and adversary's rewards after playing actions a_m and a_l in game G_i respectively
- $P_M(s, t, a, a')$ - probability transition function
- $\epsilon > 0$
- T (ϵ-return mixing time of the optimal policy) is known

The R-MAX Algorithm

- Maintain an internal model of the stochastic game
- Calculate an optimal policy according to the model and follow it
- Update model based on observations
- Calculate a new optimal policy and repeat

The R-MAX Algorithm

Input

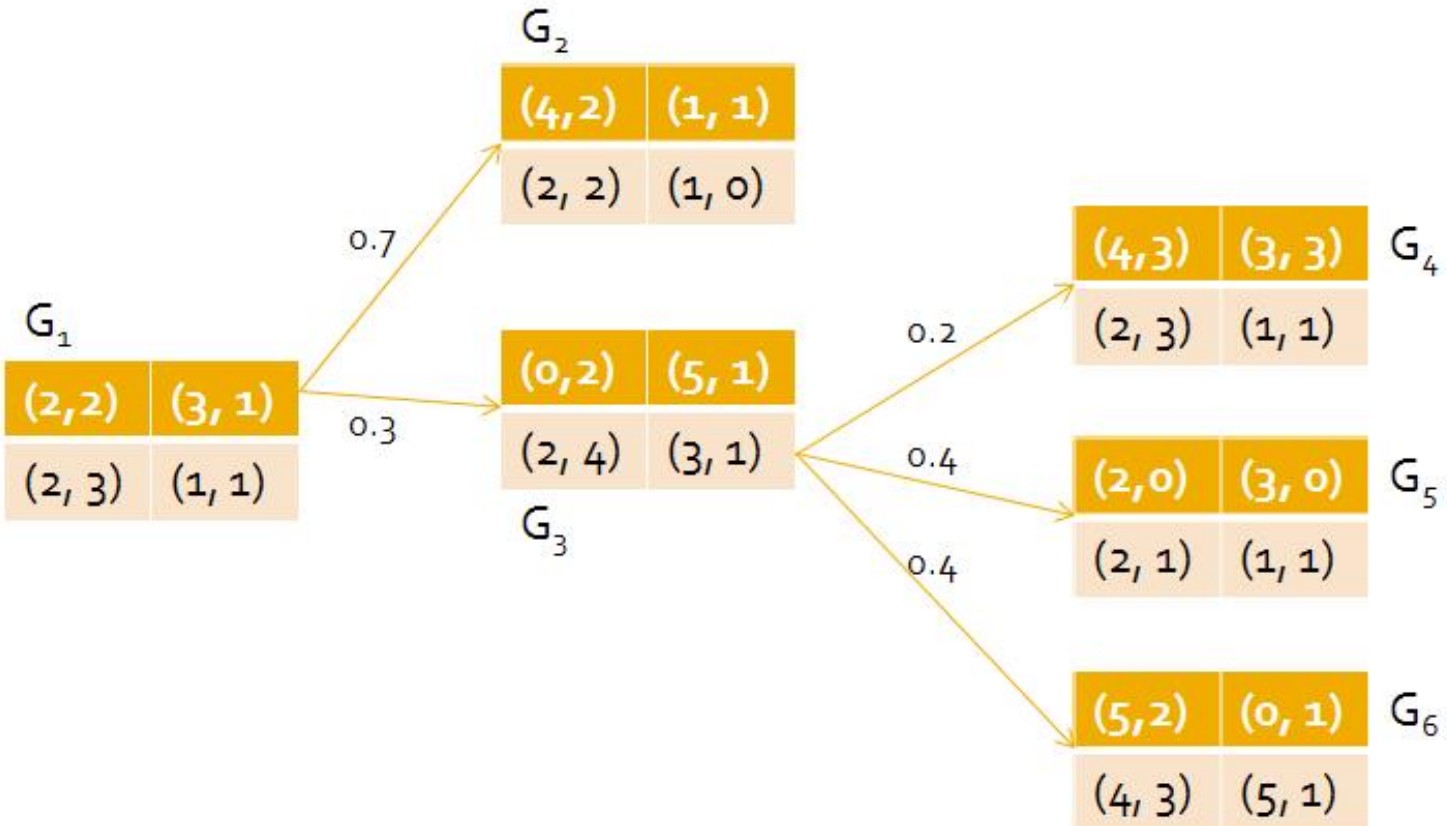
- N - number of games
- k - number of actions in each game
- ϵ - the error bound
- d - the probability of failure
- R_{max} - the maximum reward value
- T - the ϵ-return mixing time of an optimal policy
- $K_1 = max(\lceil \frac{4NTR_{max}}{\epsilon}^3 \rceil, \lceil -6ln^3(\frac{\delta}{6Nk^2}) \rceil) + 1$

The R-MAX Algorithm

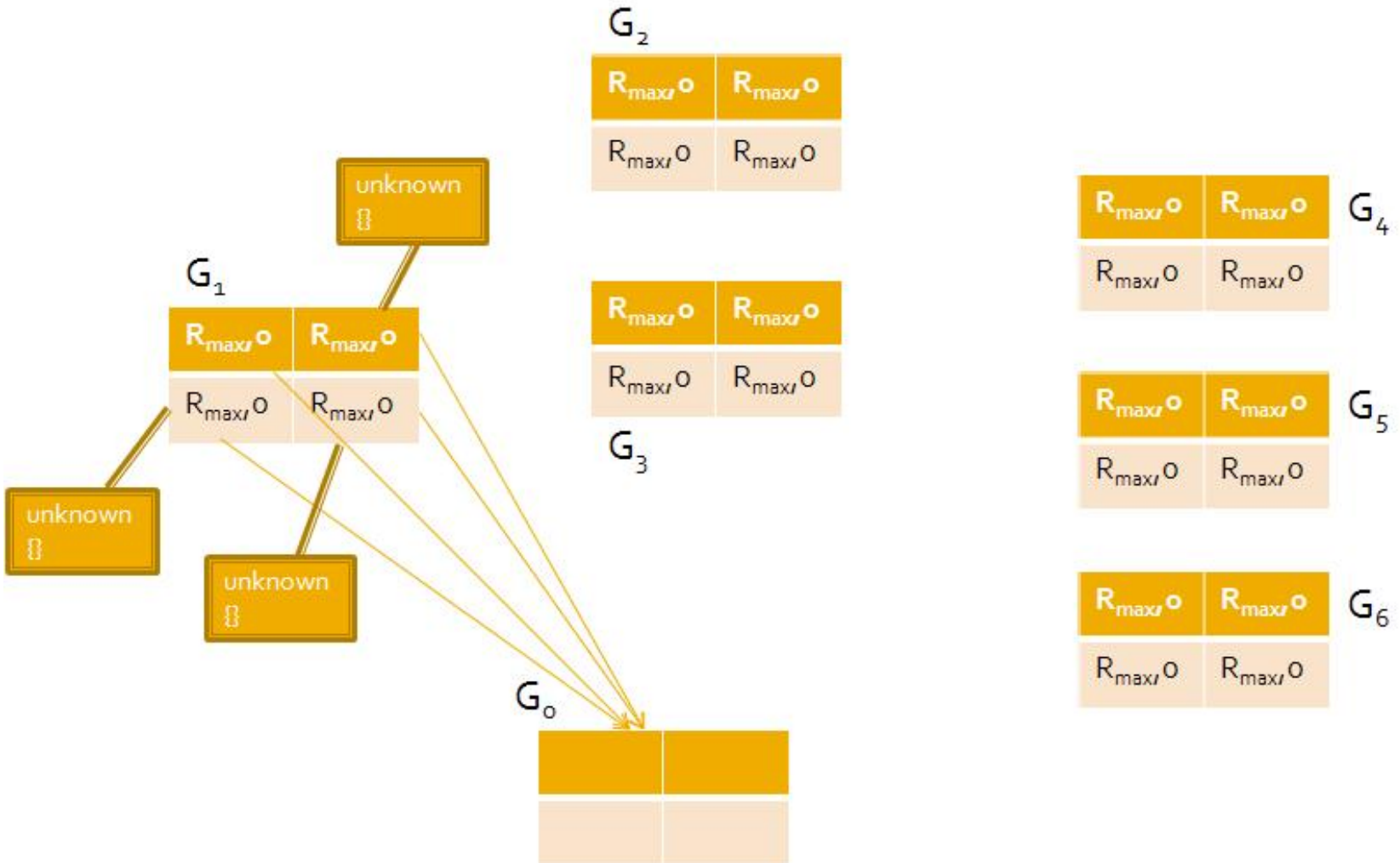
Initializing the internal model

- Create states $\{G_1 \ldots G_n\}$ to represent the stages in the stochastic game
- Create a fictitious game G_0
- Initialize all rewards to $(R_{max}, 0)$
- Set all transfer functions to point to G_0
- Associate a boolean known/unknown variable with each entry in each game, initialized to unknown
- Associate a list of states reached with each entry, which is initially empty

The R-MAX Algorithm



The R-MAX Algorithm



The R-MAX Algorithm

Repetition

- Compute an optimal policy for T steps based on the current internal model
- Execute that policy for T steps
- After each step:
  - If an entry was visited for the first time, update the rewards based on observations
  - Update the list of states reached from that entry
  - If the list of states reached now contains K_1 elements
    - mark that entry as known
    - update the transition function
    - compute a new policy

Proof of Near-Optimality

Aim: prove that if the agent follows the R-MAX algorithm for T steps, the average reward will be within ϵ of the optimum

Outline of the proof:

- After T steps the agent either obtains near-optimum average reward or learns something new
- There are a polynomial number of parameters to learn $\rightarrow$ the agent can completely learn its model in polynomial time
- The adversary can block learning, but if it does so then the agent will obtain near-optimum reward
- Either way the agent wins!

Optimality and Convergence

Definition 3 *Let M and $\bar{M}$ be SGs over the same state and action spaces. We say that $\bar{M}$ is an α-approximation of M if for every state s we have:*

1. *If $P_M(s,t,a,a')$ and $P_{\bar{M}}(s,t,a,a')$ are the probabilities of transition from state s to state t given that the joint action carried out by the agent and the adversary is (a,a'), in M and $\bar{M}$ respectively, then, $P_M(s,t,a,a') - \alpha \leq P_{\bar{M}}(s,t,a,a') \leq P_M(s,t,a,a') + \alpha$*
2. *For every state s, the same stage-game is associated with s in $\bar{M}$ and in M (and thus, the rewards will be identical).*

Optimality and Convergence

Extension of the Simulation Lemma (by Kearns and Singh 1998)

Lemma 4 *Let M and $\bar{M}$ be SGs over N states, where $\bar{M}$ is an $\frac{\epsilon}{NTR_{max}}$-approximation of M, then for every state s, agent policy π, and adversary policy ρ, we have that*

$$|U_{\bar{M}}(s, \pi, \rho, T) - U_M(s, \pi, \rho, T)| \leq \epsilon.$$

- If the agent's model is a sufficiently close approximation of the true game, then an optimal policy in the model will be near-optimal in the game
- R-MAX guarantees that the model is sufficiently close by waiting until there are K_1 samples before marking an entry known

Optimality and Convergence

Induced SG

Definition 5 *Let M be an SG. Let L be the set of entries (G_i, a, a') marked* unknown. *That is, if $(G_i, a, a') \in L$ then the entry corresponding to the joint action (a, a') in the stage-game G_i is marked as* unknown. *Define M_L to be the following SG: M_L is identical to M, except that M_L contains an additional state G_0. Transitions and rewards associated with all entries in M_L that are* not *in L are identical to those in M. For any entry in L or in G_0, the transitions are with probability 1 to G_0, and the reward is R_{max} for the agent and 0 for the adversary.*[5]

- $R^{M_L} - max$: optimal policy for the induced SG M_L
  - M is a given stochastic game with a set of L unknown states

Optimality and Convergence

Implicit explore or exploit lemma

Lemma 6 *Let M be an SG, let L and M_L be as above. Let ρ be an arbitrary policy for the adversary, let s be some state, and let $0 < \alpha < 1$. Then either (1) $V_{R-max} > Opt(\Pi_M(\epsilon, T)) - \alpha$, where V_{R-max} is the expected T-step average reward for the R^{M_L}-max policy on M; or (2) An unknown entry will be played in the course of running* R-MAX *on M for T steps with a probability of at least $\frac{\alpha}{R_{max}}$.*

- The difference between the expected reward based on the model and the actual reward will be less than the exploration probability times R_{max}
- The value of exploration in the fictitious model is very hight (probabilistic maximin)

Optimality and Convergence

Main Theorem

Theorem 2 *Let M be an SG with N states and k actions. Let $0 < \delta < 1$, and $\epsilon > 0$ be constants. Denote the policies for M whose ϵ-return mixing time is T by $\Pi_M(\epsilon, T)$, and denote the optimal expected return achievable by such policies by $Opt(\Pi_M(\epsilon, T))$. Then, with probability of no less than $1 - \delta$ the* R-MAX *algorithm will attain an expected return of $Opt_M(\Pi(\epsilon, T)) - 2\epsilon$ within a number of steps polynomial in $N, k, T, \frac{1}{\epsilon}$, and $\frac{1}{\delta}$.*

Main Theorem

Sketch of the proof: combining previous two lemmas

- In unknown states the agent has an unrealistically high expectation of reward (R_{max})
  - According to the implicit explore or exploit lemma the probability of exploration is high
- In known states according to the simulation lemma agent will obtain near-optimal reward
- Agent will always be in either a known or unknown state
- Therefore, agent will always explore with high probability or obtain near-optimal reward
- If agent explores for long enough then (almost) all states will be known, and near-optimal reward is guaranteed

Repeated Games

- A repeated game is a stochastic game in which the set of stage games contains a single game
- Adaptive Competitive Decision Process is a class of repeated games with incomplete information
  - can observe adversary's actions and payoffs
  - value of R_{max} is known
- The mixing time of any policy is 1 (since one stage game)
- Have the same guarantees as in SGs but with simpler algorithm

R-MAX algorithm for repeated games

Simplified R-MAX

Initialization Initialize the game model with payoffs of R_{max} for every joint action for the agent and 0 for the adversary. Mark all joint actions as *unknown*.

Play Repeat the following process:

Policy Computation Compute an optimal policy for the game based on the current model and play it.

Update If the joint action played is marked unknown, update the game matrix with its observed payoffs and mark is *known*.

Other variations of the R-MAX algorithm

- Remove assumption that we know the ϵ-return mixing time of an optimal policy, T
- Remove assumption that we know R_{max}

Issues

- For complicated games N, k, T are all likely to be high, so polynomial time will still not be computationally feasible
- Do not consider how the agent's behavior might impact the adversary's behavior

Conclusion

- R-Max is a model-based reinforcement learning algorithm that is guaranteed to converge on the near-optimal average reward in polynomial time
  - zero-sum stochastic games, MDPs, and repeated games
- Theoretical justification for optimism under uncertainty heuristic are provided
  - Guarantee that the agent either obtains near-optimal reward or learns efficiently