

Approximate Universal Artificial Intelligence

A Monte-Carlo AIXI Approximation

Joel Veness[†] Kee Siong Ng^{*} Marcus Hutter^{*'} David Silver[‡]

[†] University of New South Wales

['] National ICT Australia

^{*} The Australian National University

[‡] University College London

September 8, 2010

General Reinforcement Learning Problem

Worst case scenario: Environment is unknown. Observations may be noisy. Effects of actions may be stochastic. No explicit notion of state. Perceptual aliasing. Rewards may be sparsely distributed.

Notation:

- ▶ Agent interacts with an unknown environment μ by making actions $a \in \mathcal{A}$.
- ▶ Environment responds with observations $o \in \mathcal{O}$ and rewards $r \in \mathcal{R}$. For convenience, we sometimes use $x \in \mathcal{O} \times \mathcal{R}$.
- ▶ $x_{1:n}$ denotes x_1, x_2, \dots, x_n , $x_{<n}$ denotes x_1, x_2, \dots, x_{n-1} and $ax_{1:n}$ denotes $a_1, x_1, a_2, x_2, \dots, a_n, x_n$.

MC-AIXI-CTW in context

Some approaches to (aspects of) the general reinforcement learning problem:

- ▶ Model-free RL with function approximation (e.g. TD)
- ▶ POMDP (assume an observation / transition model, maybe learn parameters?)
- ▶ Learn some (hopefully compact) state representation, then use MDP solution methods

Our approach:

- ▶ Directly approximate AIXI, a universal Bayesian optimality notion for general reinforcement learning agents.

AIXI: A Bayesian Optimality Notion

$$a_t^{AIXI} = \arg \max_{a_t} \sum_{x_t} \dots \max_{a_{t+m}} \sum_{x_{t+m}} \left[\sum_{i=t}^{t+m} r_i \right] \sum_{\rho \in \mathcal{M}} 2^{-K(\rho)} \rho(x_{1:t+m} | a_{1:t+m}),$$

- ▶ Expectimax + (generalised form of) Solomonoff Induction
- ▶ Model class \mathcal{M} contains all enumerable chronological semi-measures.
- ▶ Kolmogorov Complexity used as an Ockham prior.
- ▶ $m := b - t + 1$ is the "remaining search horizon".
 b is the maximum age of the agent

Caveat: Incomputable. **Not an algorithm!**

Describing Environments, AIXI Style

- ▶ A *history* h is an element of $(\mathcal{A} \times \mathcal{X})^* \cup (\mathcal{A} \times \mathcal{X})^* \times \mathcal{A}$.
- ▶ An *environment* ρ is a sequence of conditional probability functions $\{\rho_0, \rho_1, \rho_2, \dots\}$, where for all $n \in \mathbb{N}$, $\rho_n: \mathcal{A}^n \rightarrow \text{Density}(\mathcal{X}^n)$ satisfies

$$\forall \mathbf{a}_{1:n} \forall \mathbf{x}_{<n} : \rho_{n-1}(\mathbf{x}_{<n} | \mathbf{a}_{<n}) = \sum_{x_n \in \mathcal{X}} \rho_n(\mathbf{x}_{1:n} | \mathbf{a}_{1:n}), \rho_0(\epsilon | \epsilon) = 1.$$

- ▶ The ρ -probability of observing x_n in cycle n given history $h = \mathbf{a}x_{<n}\mathbf{a}_n$ is

$$\rho(x_n | \mathbf{a}x_{<n}\mathbf{a}_n) := \frac{\rho(\mathbf{x}_{1:n} | \mathbf{a}_{1:n})}{\rho(\mathbf{x}_{<n} | \mathbf{a}_{<n})}$$

provided $\rho(\mathbf{x}_{<n} | \mathbf{a}_{<n}) > 0$.

Learning a Model of the Environment

We will be interested in agents that use a *mixture environment model* to learn the true environment μ .

$$\xi(x_{1:n}|a_{1:n}) := \sum_{\rho \in \mathcal{M}} w_0^\rho \rho(x_{1:n}|a_{1:n})$$

- ▶ $\mathcal{M} := \{\rho_1, \rho_2, \dots\}$ is the model class
- ▶ w_0^ρ is the prior weight for environment ρ .
- ▶ Satisfies the definition of an environment model.

Therefore, can predict by using:

$$\xi(x_n|ax_{<n}a_n) = \sum_{\rho \in \mathcal{M}} w_{n-1}^\rho \rho(x_n|ax_{<n}a_n), \quad w_{n-1}^\rho := \frac{w_0^\rho \rho(x_{<n}|a_{<n})}{\sum_{v \in \mathcal{M}} w_0^v v(x_{<n}|a_{<n})}$$

Theoretical Properties

Theorem: Let μ be the true environment. The μ -expected squared difference of μ and ξ is bounded as follows. For all $n \in \mathbb{N}$, for all $\mathbf{a}_{1:n}$,

$$\sum_{k=1}^n \sum_{\mathbf{x}_{1:k}} \mu(\mathbf{x}_{<k} | \mathbf{a}_{<k}) \left(\mu(x_k | \mathbf{a}\mathbf{x}_{<k} \mathbf{a}_k) - \xi(x_k | \mathbf{a}\mathbf{x}_{<k} \mathbf{a}_k) \right)^2 \leq \min_{\rho \in \mathcal{M}} \left\{ -\ln w_0^\rho + D_{KL}(\mu(\cdot | \mathbf{a}_{1:n}) \| \rho(\cdot | \mathbf{a}_{1:n})) \right\},$$

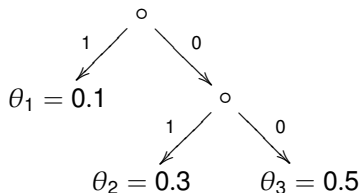
where $D_{KL}(\cdot \| \cdot)$ is the KL divergence of two distributions.

Roughly: The predictions made by ξ will converge to those of μ if a model close (w.r.t. KL Divergence) to μ is in \mathcal{M} .

Prediction Suffix Trees

A prediction suffix tree is a simple, tree based variable length Markov model. For example, using the PST below, having initially been given data 01:

$$\begin{aligned}\Pr(010|01) &= \Pr(0|01) \times \Pr(1|010) \times \Pr(0|0101) \\ &= (1 - \theta_1)\theta_2(1 - \theta_1) \\ &= 0.9 * 0.3 * 0.9 \\ &= 0.243\end{aligned}$$



Context Tree Weighting

- ▶ Context Tree Weighting is an online prediction method.
- ▶ CTW uses mixture of prediction suffix trees.
- ▶ Smaller suffix trees are given higher initial weight, which helps to avoid overfitting when data is limited.
- ▶ Let C_D denote the class of all prediction suffix trees of maximum depth D , then CTW computes in time $O(D)$:

$$\Pr(x_{1:t}) = \sum_{M \in C_D} 2^{-\Gamma_D(M)} \Pr(x_{1:t}|M)$$

- ▶ $\Gamma_D(M)$ is description length of context tree M .
- ▶ **This is truly amazing**, as computing the sum naively would take time double-exponential in D !

Model Class Approximation

Action-Conditional Context Tree Weighting Algorithm:

Approximate model class of AIXI with a mixture over *all* action-conditional Prediction Suffix Tree structures of maximum depth D .

- ▶ PSTs are a form of variable order Markov model.
- ▶ Context Tree Weighting algorithm can be adapted to compute a mixture of over $2^{2^{D-1}}$ environment models in time $O(D)$!
- ▶ Inductive bias: smaller PST structures favoured.
- ▶ PST parameters are learnt using KT estimators. KL-divergence term in previous theorem grows $O(\log n)$.
- ▶ Intuitively, efficiency of CTW is due to clever exploitation of shared structure.

Greedy Action Selection

- ▶ Action $a \in \mathcal{A}$ has **value** $V(a) = \mathbf{E}[R|a]$ = expected return.
- ▶ Consider **Bandit setting**: No history or state dependence.
- ▶ **Optimal action/arm**: $a^* := \arg \max_a V(a)$ (unknown).
- ▶ $V(a)$ unknown \Rightarrow **frequency estimate**
 $\hat{V}(a) := \frac{1}{T(a)} \sum_{t \in T(a)} R_t$, R_t = actual return at time t .
 $T(a) := \{t \leq T : a_t = a\}$ = #times arm a taken so far.
- ▶ **Greedy action**: $a_{T+1}^{greedy} = \arg \max_a \hat{V}(a)$
- ▶ **Problem**: If a^* accidentally looks bad (low early $\hat{V}(a)$), it will never be taken again = **explore/exploit dilemma**.
- ▶ **Solution**: Optimism in the face of uncertainty ...

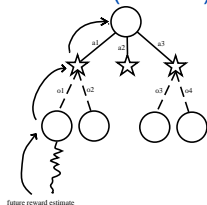
Upper Confidence Algorithm for Bandits

- ▶ **UCB action:** $a_{T+1}^{UCB} := \arg \max_a V^+(a)$
 $V^+(a) := \hat{V}(a) + C \sqrt{\frac{\log T}{T(a)}}$, $C > 0$ suitable constant.
- ▶ If **arm under-explored** (i.e. $T(a) \ll \log T$)
⇒ $V^+(a)$ huge ⇒ UCB will take arm a
⇒ Every arm taken infinitely often ⇒ $\hat{V}(a) \rightarrow V(a)$
- ▶ If sub-optimal **arm over-explored** (i.e. $T(a) \gg \log T$)
⇒ $V^+(a) \approx \hat{V}(a) \rightarrow V(a) < V(a^*) \leftarrow \hat{V}(a^*) < V^+(a^*)$
⇒ UCB will **not** take arm a
- ▶ **Fazit:** $T(a) \propto \log T$ for all suboptimal arms.
 $T(a^*) = T - O(\log T)$, i.e. only $O(\log T) \ll T$ subopt. actions
- ▶ **UCB is theor. optimal** explore/exploit strategy for Bandits.
Application: Use “heuristically” in expectimax tree search...

Expectimax Approximation

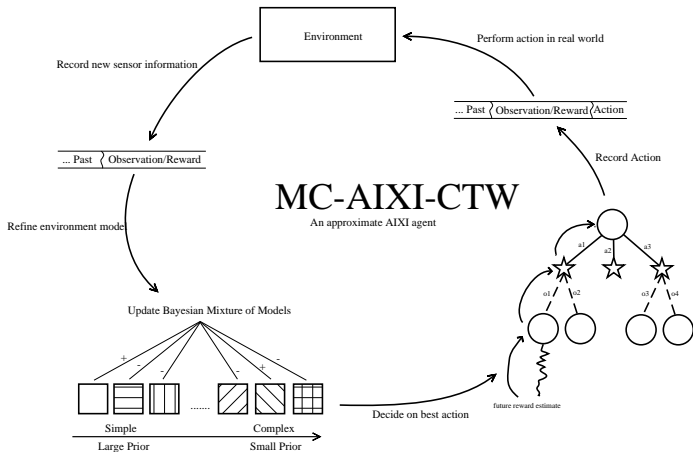
Monte Carlo approximation of expectimax Tree Search (MCTS) Upper Confidence Tree (UCT) algorithm:

- ▶ **Sample** observations from CTW distribution.
- ▶ **Select** actions with highest Upper Confidence Bound (UCB) V^+ .
- ▶ **Expand** tree by one leaf node (per trajectory).
- ▶ **Simulate** from leaf node further down using (fixed) playout policy.
- ▶ **Propagate back** the value estimates for each node.
Repeat until timeout.



- With sufficient time, **converges** to the expectimax solution.
- **Value of Information** correctly incorporated when instantiated with a mixture environment model.
- Gives Bayesian solution to the **exploration/exploitation** dilemma.

Agent Architecture (MC-AIXI-CTW = UCT+CTW)



Relationship to AIXI

Given enough thinking time, MC-AIXI-CTW will choose:

$$a_t = \arg \max_{a_t} \sum_{x_t} \cdots \max_{a_{t+m}} \sum_{x_{t+m}} \left[\sum_{i=t}^{t+m} r_i \right] \sum_{M \in \mathcal{C}_D} 2^{-\Gamma_D(M)} \Pr(x_{1:t+m} | M, a_{1:t+m})$$

In contrast, AIXI chooses:

$$a_t = \arg \max_{a_t} \sum_{x_t} \cdots \max_{a_{t+m}} \sum_{x_{t+m}} \left[\sum_{i=t}^{t+m} r_i \right] \sum_{\rho \in \mathcal{M}} 2^{-K(\rho)} \Pr(x_{1:t+m} | a_{1:t+m}, \rho)$$

Algorithmic Considerations

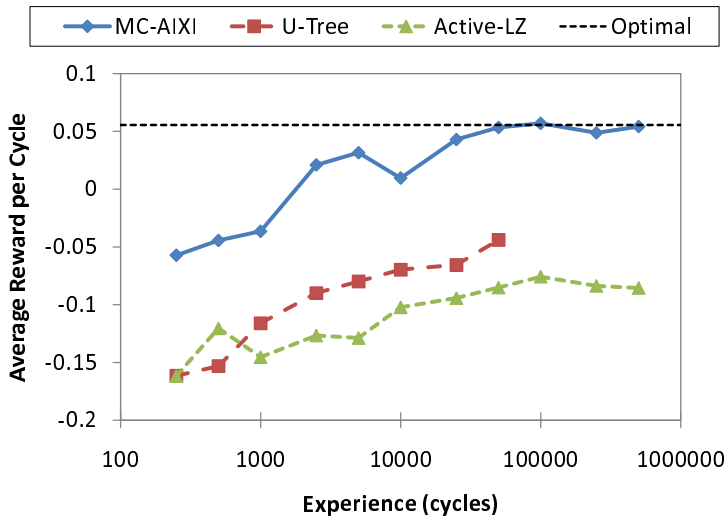
- ▶ Restricted the model class to gain the desirable computational properties of CTW
- ▶ Approximated the finite horizon expectimax operation with a MCTS procedure
- ▶ $O(Dm \log(|\mathcal{O}||\mathcal{R}|))$ operations needed to generate m observation/reward pairs (for a single simulation)
- ▶ $O(tD \log(|\mathcal{O}||\mathcal{R}|))$ space overhead for storing the context tree.
- ▶ Anytime search algorithm
- ▶ Search can be parallelized
- ▶ $O(D \log(|\mathcal{O}||\mathcal{R}|))$ to update the context tree online

Experimental Setup

- ▶ Agent tested on a number of POMDP domains, as well as TicTacToe and Kuhn Poker.
- ▶ Agent required to *both* learn *and* plan.
- ▶ The context depth and search horizon were made as large as possible subject to computational constraints.
- ▶ ϵ -Greedy training, with a decaying ϵ
- ▶ Greedy evaluation

Comparison to Other RL Algorithms

Learning Scalability - Kuhn Poker



Resources Required for (Near)Optimal Performance

Domain	Experience	Simulations	Search Time
Cheese Maze	5×10^4	500	0.9s
Tiger	5×10^4	10000	10.8s
4×4 Grid	2.5×10^4	1000	0.7s
TicTacToe	5×10^5	5000	8.4s
Biased RPS	1×10^6	10000	4.8s
Kuhn Poker	5×10^6	3000	1.5s

- ▶ Timing statistics collected on an Intel dual quad-core 2.53Ghz Xeon.
- ▶ Toy problems solvable in reasonable time on a modern workstation.
- ▶ General ability of agent will scale with better hardware.

Limitations

- ▶ PSTs inadequate to represent many simple models compactly. For example, it would be unrealistic to think that the current MC-AIXI-CTW approximation could cope with real-world image or audio data.
- ▶ Exploration/exploitation needs more attention. Can something principled *and* efficient be done for general Bayesian agents using large model classes?

Future Work

- ▶ Uniform random rollout policy used in ρ UCT. A learnt policy should perform much better.
- ▶ All prediction was done at the bit level. Fine for a first attempt, but no need to work at such a low level.
- ▶ Mixture environment model definition can be extended to continuous model classes.
- ▶ Incorporate more (action-conditional) Bayesian machinery.
- ▶ Richer notions of context.

References

- ▶ For more information, see:

A Monte-Carlo AIXI Approximation,
J. Veness, K.S. Ng, M. Hutter, W. Uther, D. Silver
<http://jveness.info/publications/default.html>

Highlights: a direct comparison to U-Tree / Active-LZ, improved model class approximation (FAC-CTW) and more relaxed presentation.

- ▶ Video of the latest version playing Pacman
<http://www.youtube.com/watch?v=yfsMHTmGDKE>