Research School of Computer Science

College of Engineering and
Computer Science

An Implementation of MC-AIXI-CTW

Jarrah Bloomfield* Luke English† Andrew Haigh‡ Joshua Nelson§ Anthony Voutas¶

COMP4620 - Advanced Artificial Intelligence
Assignment 2

vspace1.4cm

Bottom of the page October 31, 2012

u4669875* u4667010† u4669875‡ u4850020§ u4519169¶

Chapter 1

Description of the MC-AIXI-CTW implementation

The AIXI agent is a formal, mathematical agent which represents a solution to the general reinforcement learning problem. Principally, AIXI consists of an expectimax search over a Bayesian mixture of Turing machines in order to choose optimal actions by predicting future observations and rewards based on past experience. Given infinite computational resources, AIXI represents the optimal reinforcement agent: maximising future expected reward in any unknown environment.

In the far more limited world of what is tractable, we require an approximation to AIXI. Here, we approximate AIXI via a combination of UCT search (Monte-Carlo Tree Search with the Upper Confidence Bound) (**?**) and Context Tree Weighting (**?**), yielding MC-AIXI-CTW.

Chapter 2

User Manual

2.1 Arguments

The agent can be compile with the `make` command. The agent then can then be run using

```
./main <environment> <logfile>
```

`<environment>` is a compulsory argument, which specifies the environment configuration file the agent is to use. In this implementation, it is one of the following

- `coinflip.conf`: Biased coin flip enviroment
- `grid.conf`: Gridworld environment
- `kuhnpoker.conf`: Kuhn poker environment
- `pacman.conf`: Pacman environment
- `rps.conf`: Biased rock paper scissors environment
- `tiger.conf`: "Tiger" Environment
- `composite.conf`: A combination of the above environments

`<logfile>` is an optional argument, which specifies the name of a log file to output results to.

2.2 Configuration files

`.conf` files are *configuration files*, specifying which environment is to be used, and relevant parameters for each environment. Each configuration file has the following parameters

- `environment`: The name of the environment to use. One of {4x4-grid, kuhn-poker, tiger, biased-rock-paper-scissor, pacman, composite}.
- `exploration`: The rate at which the agent explores, by making random decisions.
- `explore-decay` : The rate at the exploration rate decreases

In addition to this, some configurations have parameters that are specific to their environments.

- Coinflip
  - `coin-flip-p`: The probability of a flipping heads ($0 \leq$ `coin-flip-p` ≤ 1).
- Kuhn poker

  - `gamma`: A constant that determines the environment's Nash equilibrium strategy. ($0 \leq$ `gamma` ≤ 1)

- Pacman
  - `mapfile`: The location of the map file for the pacman board.

- tiger.conf
  - `left-door-p`: The probability that the gold is behind the left door
  - `listen-p`: The probability that a listening observation is correct.

- composite.conf
  - `environmentN`: Specifies the N^{th} environment, where $0 \leq N \leq 10$. The value of this parameter is an integer ≤ 10, and indicates which environment environmentN represents.
  - `startN`: Specifies the time step that at which the N^{th} environment starts, where $0 \leq N \leq 10$.
  - Paremeters required for the environemnts 1..N specified in `environment.cpp`.

Chapter 3

MC-AIXI-CTW Implementation

3.1 Context Tree Weighting

3.2 Upper Confidence Tree

3.3 Revert Function

Chapter 4

Experimentation

4.1 Sequence prediction

The CTW tree is the primarmy prediction mechanism in the MC-AIXI-CTW model. The CTW implementation of MC-AIXI-CTW was tested in isolation from the rest of the agent on a range of deterministic and non-deterministic sequence. This was useful for debugging purposes, and the results are an interesting biproduct of MC-AIXI-CTW.

4.1.1 Deterministic sequence prediction

Several simple sequences were given to the CTW tree, and the CTW tree was asked to continue the sequence. This can be seen in Figure 4.1. We can see that the CTW tree is able to correctly predict the next bit, or multiple bits, in the sequence.

4.1.2 Non deterministic sequence prediction

A partially non deterministic sequence was given to CTW for testing purposes. The sequence was given to CTW was of the form

$$S := x_0^{\text{obs}},\ x_0^{\text{rew}},\ x_1^{\text{act}},\ x_1^{\text{obs}},\ x_1^{\text{rew}}, x_2^{\text{act}},\ x_2^{\text{obs}},\ x_2^{\text{rew}}, ...$$

Where

$$x_i^{\text{act}} := r_{\text{act}}, \quad x_i^{\text{obs}} := r_{\text{obs}}, \quad \text{and} \quad x_i^{\text{rew}} := \begin{cases} 0 & \text{if } x_i^{\text{obs}} = x_i^{\text{act}} \\ 1 & \text{if } x_i^{\text{obs}} \neq x_i^{\text{act}} \end{cases}$$

Where

$$r_{\text{act}}, r_{\text{obs}} \text{ are random bits}$$

Sequence	Prediction	Sequence	Prediction
0^{1000}	0...	1^{1000}	1...
$(01)^{1000}$	01...	$(10)^{1000}$	10...
$(0110)^{500}$	0110...	$(1100)^{500}$	1100...
$(110)^{500}$	110...	$(001)^{500}$	001...

Figure 4.1: Some of the sequences given to the CTW tree, and the prediction of the next symbol.

	$x_{3000}^{\text{rew}} = 0$	$x_{3000}^{\text{rew}} = 1$
$x_{3000}^{\text{rew}} = 0$	1	0
$x_{3000}^{\text{rew}} = 1$	0	1

Figure 4.2: Predicted values for x_{3000}^{rew}

The idea of this sequence S is to simulate a coinflip environment history sequence. The random bits x_i^{obs} and x_i^{act} simulate random coin flips, and the reward bit x_i^{act} compares them. The action bit x_i^{act} is random in this sequence.

A sequence of this type with size 9000 (3000 iterations) is given to the CTW tree, up to the point

$$S = x_0^{\text{obs}},\ x_0^{\text{rew}},\ x_1^{\text{act}},\quad \ldots\quad , x_{3000}^{\text{act}},\ x_{3000}^{\text{obs}}$$

The CTW tree is then asked to predict which bit is the next in the sequence. The idea of this experiment is to determine if the CTW tree "understands" the rules of the game - if it understood, it would predict $x_{3000}^{\text{rew}} = (x_{3000}^{\text{act}} \wedge x_{3000}^{\text{obs}}) \vee (\neg x_{3000}^{\text{act}} \wedge \neg x_{3000}^{\text{obs}})$.

The results of this experiment are show in Figure 4.2. We see that it correctly predicts the reward bit given the action and observation bits. This provides some verification that the implementation of the CTW tree is correct.

4.2 Coin flip results

Coin flip is a simple environment where the agent is given a reward of 1 for correctly guessing the next bit, where the next bit is random with a certain bias such that the next bit is one with probability $\theta \in [0, 1]$, else the reward is 0.

4.3 Tiger results

The parameters used were:

mc-simulations	50
exploration	0.01
explore-decay	0.999

4.4 Grid world results

mc-simulations	50
exploration	0.01
explore-decay	0.99999

4.5 Biased Rock paper scissors results

In Biased Rock Paper Scissors (RPS), the agent plays against an environment which will randomly select rock, paper or scissors; but after winning with rock, the environment will repeat a rock move. This is interesting because it creates a bias in the environment's distribution, and the agent needs to try to exploit this bias.

The parameters used were:

mc-simulations	50
exploration	0.1
explore-decay	0.999

4.6 Kuhn Poker results

In Kuhn Poker, the agent plays against a Nash Equillibruim AI, with gamma set to 0.5. The agent is given a reward of 0 for a loss, 2 for a win or 1 if no showdown has occurred so far.

The parameters used were:

mc-simulations	50
exploration	0.01
explore-decay	0.9999

4.7 Pacman results

In Pacman

The parameters used were:

mc-simulations	50
exploration	0.01
explore-decay	0.99999

Table 5.1 Number of problems where complete solutions were found

The parameters used were:

Chapter 5

Extension