Atari Breakout with $\mathrm{LTL}_f/\mathrm{LDL}_f$ Goals

Ivan Bergonzani, Michele Cipriano, Armando Nania

Professor: Giuseppe De Giacomo

Elective in Artificial Intelligence: Reasoning Robots
Department of Computer, Control and Management Engineering
Sapienza University of Rome

Introduction

The main goal of the project is to train an agent acting on a non-Markovian reward decision process (NMRDP) environment.

Rewards are define through LTL_f/LDL_f formulas, defining complex behaviour made up of multiple steps over time.

The learning is done using classical reinforcement learning algorithm working on MDP environments.

Q-Learning

Q-Learning is a temporal difference (TD) reinforcement learning algorithm.

The Q-function is approximated through a Q-table that stores the expected rewards for taking action A in state S.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \Big]$$

Off-policy algorithm: action at time $t+1$ independent from current policy.

Converges to optimal policy if all state-action pairs are continuously updated.

SARSA

SARSA is a TD reinforcement learning algorithm which uses a quintuple $\langle S_t, A_t, R_t, S_{t+1}, A_{t+1} \rangle$ from which takes name.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \Big]$$

On-policy algorithm: action A_{t+1} chosen following current policy.

Converge to optimal policy if each state-action pair is visited infinitely often.

$\text{LTL}_f/\text{LDL}_f$ Non-Markovian Rewards

$\text{LTL}_f/\text{LDL}_f$ non-Markovian rewards + integration in our project.

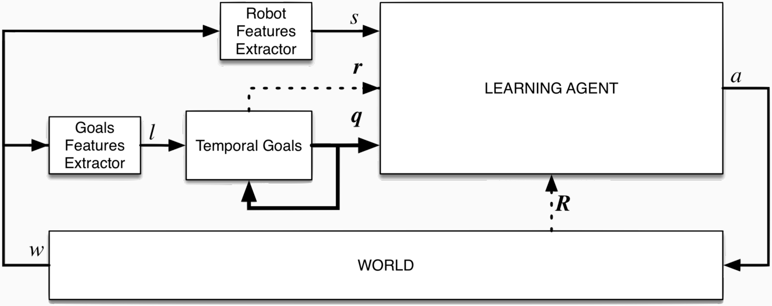


Figure 1: Pipeline describing how the agent is interacting with the world and how the robot features extractor and the goal features extractor are used in order to handle non-Markovian rewards.

$\mathrm{LTL}_f/\mathrm{LDL}_f$ Non-Markovian Rewards

Reward for destroying the block rows in order from top to bottom.

$$\langle(\neg\varphi_0 \wedge \neg\varphi_1 \wedge \neg\varphi_2)^*; (\varphi_0 \wedge \neg\varphi_1 \wedge \neg\varphi_2); (\varphi_0 \wedge \neg\varphi_1 \wedge \neg\varphi_2)^*;$$
$$(\varphi_0 \wedge \varphi_1 \wedge \neg\varphi_2); (\varphi_0 \wedge \varphi_1 \wedge \neg\varphi_2)*; (\varphi_0 \wedge \varphi_1 \wedge \varphi_2)\rangle tt$$

where φ_i indicates the i-th row.

$(\neg\varphi_0 \wedge \neg\varphi_1 \wedge \neg\varphi_2)^*$	all rows not destructed for an indefinite amount of time.
$(\varphi_0 \wedge \neg\varphi_1 \wedge \neg\varphi_2)$	first row destructed, other still there.

$\text{LTL}_f/\text{LDL}_f$ Non-Markovian Rewards

$\text{LTL}_f/\text{LDL}_f$ transformed to an equivalent automaton.

Automaton representation given in input to the learning agent.

Learning agent input: $\langle$environment state, automaton state$\rangle$

PyGame Breakout

Starting from the work presented in [4] the agents were trained on a modified PyGame Breakout with a 6×18 grid.

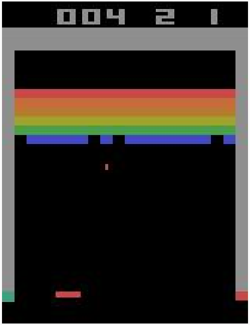
Game state already stored inside the class instance (ball position, paddle position, blocks status).

Atari Breakout

Breakout from OpenAI Gym: toolkit for comparing learning algorithm. Provides easy access to state and reward. No assumption on the agent.

Class stores pixels status. Need to extract information about the game state from it.

The environment is more stochastic.



Implementation

Atari Wrappers

Atari wrappers from OpenAI baselines:

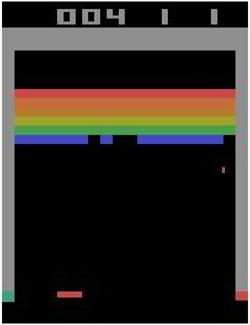
- `EpisodicLifeEnv`: make an "end of life" be the end of the episode resetting the environment only on the true game over, this helps in value estimation
- `FireResetEnv`: use "Fire" as starting action in order to launch the ball
- `MaxAndSkipEnv`: returns only the skip-th frame, this reduces the amount of frame the agent has to deal with (set to 4 in the main experiments)

Robot Features Extraction

- extracting **paddle position**
- extracting **ball position**
- features can be increased with **ball direction**
- **previous positions** could help the training too
- more features increase the state space
- OpenCV

Robot Features Extraction

Extracting **paddle position** from the observation: tensor of dimension $(210, 160, 3)$ where each value represents a color of the pixel.



Robot Features Extraction

Considering only the **lower part** of the observation which contains the paddle.

Robot Features Extraction

Converting the image to **gray-scale** in order to reduce the number of channels.
This makes it easier to apply a threshold function.

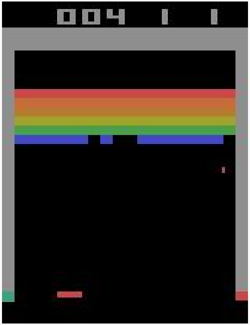
Robot Features Extraction

Determine the position of the paddle:

- apply a **threshold** function to obtain a black and white image
- find **contours**
- extract **centroid** of the paddle
- $\text{paddle}_x = 35$

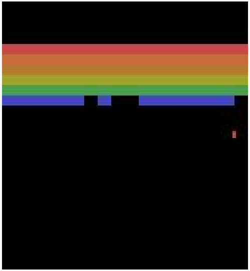
Robot Features Extraction

Similarly, extract **ball position** from initial observation.



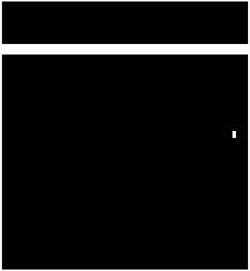
Robot Features Extraction

Considering only the **upper part** of the observation which contains the ball.



Robot Features Extraction

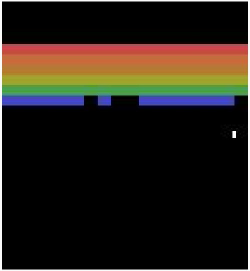
- `inRange` function to consider only pixels with the same color of the ball, obtain a black and white image
- remove the **remaining bricks** by checking sorrounding pixels, width of the ball is much smaller



Robot Features Extraction

Determine the position of the ball:

- find **contours**
- extract **centroid** of the ball
- $\text{ball}_x = 135, \text{ball}_y = 77$



Robot Features Extraction

Return the difference between the position of the paddle and the ball on the x axis and the position of the ball on the y axis:

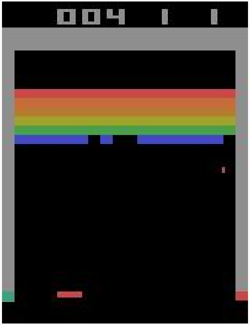
$$(\text{ball}_x - \text{paddle}_x + 143, \text{ball}_y) = (243, 77)$$

Goal Features Extraction

Extract a binary representation of the bricks from the observation:

$$
\begin{matrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0
\end{matrix}
$$

It will be used to evaluate fluents in temporal goals.



Temporal Goals

- string formula to break rows from top to bottom:

```
<(!l0 & !l1 & ...  & !l5)*;
 ( l0 & !l1 & ...  & !l5);
              ...
 ( l0 &  l1 & ...  & !l5)*;
 ( l0 &  l1 & ...  &  l5)>tt
```

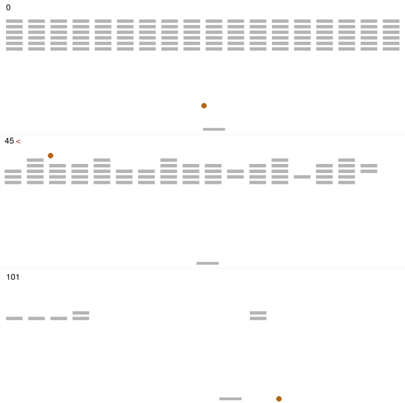
- parsed with `LDLfParser` (`FLLOAT`)
- `LDLfFormula` passed to abstract class `TemporalEvaluator`
- automaton is being created (`RLTG`, DFA built upon `Pythomata`) in order to create the extended MDP

Experiments

Experiments

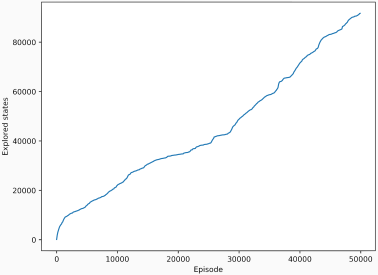
All the experiments.

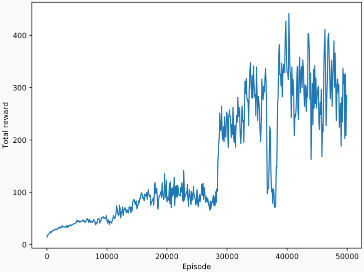
Experiments



Description.

Experiments



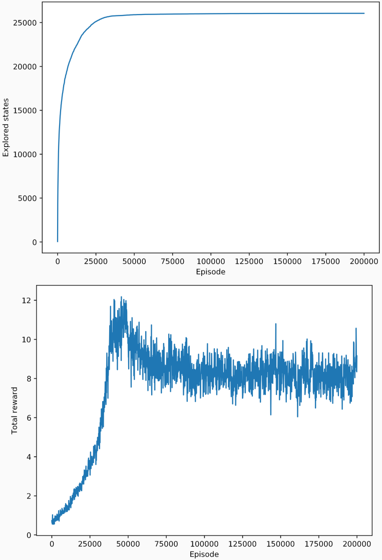


PyGame SARSA.

Experiments

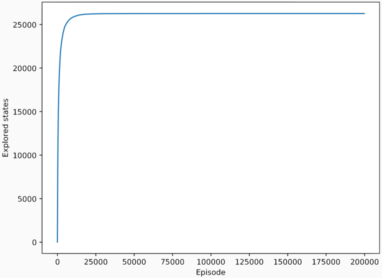
Atari frames.

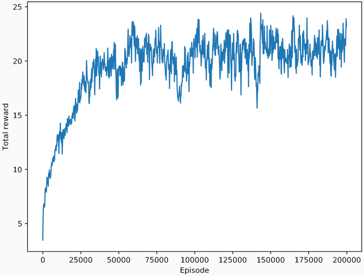
Experiments



Atari Q-Learning.

Experiments





Atari SARSA.

Conclusion

Conclusion.

Q&A

References i

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," 2016.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The Arcade Learning Environment: An Evaluation Platform for General Agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, jun 2013.

R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*.
The MIT Press, second ed., 2018.

G. De Giacomo, L. Iocchi, M. Favorito, and F. Patrizi, "Reinforcement Learning for LTLf/LDLf Goals," *CoRR*, vol. abs/1807.06333, 2018.

References ii

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

"Montezuma's Revenge Solved by Go-Explore, a New Algorithm for Hard-Exploration Problems (Sets Records on Pitfall, Too)." https://eng.uber.com/go-explore/.

- S. A. McIlraith and K. Q. Weinberger, eds., *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, AAAI Press, 2018.
- "A Python Implementation of the FLLOAT library." https://github.com/MarcoFavorito/flloat.
- "A library for generating automata from LTL and LDL formulas with finite-trace semantics in Python." https://github.com/MarcoFavorito/pythomata.
- "Framework for Reinforcement Learning with Temporal Goals defined by LDLf formulas.." https://github.com/MarcoFavorito/rltg.

References iv

G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "OpenAI Baselines." https://github.com/openai/baselines, 2017.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *CoRR*, vol. abs/1602.01783, 2016.