Simplifying the state space of Tetris in order to utilise Reinforcement Learning

Author: Donald Carr Supervisor : Philip Sterne

March 14, 2005

1 Background

ACM Classification System (1998) I.2.8 Problem Solving, Control Methods, and Search

Reinforcement learning is learning through interaction with an environment, and the consequences of these interactions, with the intention of maximising long term cumulative reward. It is therefore initially a process of trial and error, although as the entity gains experience it strives to maintain a balance between exploration of new interactions which may provide reward, and the exploitation of established beneficial interactions.

The agent functions on a policy which basically maps the perceived state of the environment to an action. The policy utilises a value function which determines the total amount of reward available in the infinite future, according to a weighting function which converges the value in the limit. The value function gets individual state values from a reward function, which associates a numerical value with individual states, and considers this in the context of the environment model.

Reinforcement learning has been successfully applied to simplified versions of Tetris[1][2], although there is no indication of anyone having had any success with devising reinforcement learning algorithms that can handle the game as it is played by humans. Simplifications normally involve a limited set of blocks with 2x2 dimensions, infinite stage height (game requirements remain constant) and uniform reward for row completion, ignoring higher reward for higher risk (4 block completion).

These simplifications are not a minor consideration, and are countering the state space explosion which occurs as the environment gets larger. This explosion is due to that fact that the possible environment states are described by the permutation of the discrete blocks comprising the environment, giving the state space 2^n complexity where the environment is composed of n blocks. The

[1]Yael Bdolah & Dror Livnat, 2000,http://www.math.tau.ac.il/ mansour/rl-course/student_proj/livnat/Tetris.html
[2]Stan Melax,http://www.melax.com/Tetris/

aforementioned simplifications have been applied to the game description rather then to the state space, greatly limiting the usefulness of the agent and resulting in the successful implementation of reinforcement learning in dull circumstances.

2 Aim

The aim of the project is to successfully create an agent which learns, via reinforcement learning methods, how to excel at full-blown Tetris. It should learn this in a reasonable time period, and be capable of playing the game with real time constraints just as a human would. The driving focus will therefore be on minimising the 2^n state space nominally required in considering all possible combinations of blocks, and achieving reinforcement learning with a lower order of complexity then that classically required by reinforcement learning.

3 Value to Science

Reinforcement learning holds the promise of delivering independent unprejudiced learning on the part of an agent supplied with a suitably accurate representation of the environment it functions in, and a corresponding reward function. The most pressing drawback associated with reinforcement learning is the vast state space that is required to described complex environments, and consequently the extensive processing time demanded. The previous implementations of Tetris have demonstrated the flexible nature of reinforcement learning, and successfully extending it to a sophisticated version of Tetris would display its pragmatic possibilities. This would greatly increase the field of possible application, such as in game AI, or possibly even massively complex environments where the state space complexity would normally rapidly overwhelm any conceivable processor due to the exponential nature of the complexity. If we can redefine any state space to a computational feasible size, reinforcement learning could be broadly applied to any scenario that afforded both accurate modelling and reward description.

4 Plan of Action

4 weeks	Research period
1 week	Code Tetris and selecting structures
3 weeks	Achieve basic learning with agent
5 weeks	Optimisation of state space
3 weeks	Testing

4.1 Literature survey

- Reinforcement Learning : An Introduction - Richard S. Sutton and Andrew G. Barto

  This is a practical introduction, covering everything from the history of reinforcement learning to the investigation of several established applications of it. It will be used as a conceptual base, and supply information about the wide span of approaches available within reinforcement learning.

  All the following items are from the Journal of Artificial intelligence, and are pertinent to my considerations

- Reinforcement Learning: A Survey - Leslie Pack Kaelbling, Michael L. Littman and Andrew W. Moore (vol 4, 1996)
- Evolutionary Algorithms for Reinforcement Learning - David E. Moriarty, Alan C. Schultz and John J. Grefenstette (vol 11, 1999)
- Accelerating Reinforcement Learning by Composing Solutions of Automatically Identified Subtasks - Chris Drummond (vol 16, 2002)
- Truncating Temporal Differences: On the Efficient Implementation of TD(lambda) for Reinforcement Learning - Pawel Cichosz (vol 2, 1995)
- Potential-Based Shaping and Q-Value Initialization are Equivalent - Eric Wiewiora (vol 19, 2003)
- Accelerating Reinforcement Learning through Implicit Imitation - Bob Price and Craig Boutilier (vol 19, 2003)
- Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition - Thomas G. Dietterich (vol 13, 2000)
- Learning to Coordinate Efficiently: A Model-based Approach - Ronen I. Brafman and Moshe Tennenholtz (vol 19, 2003)
- Infinite-Horizon Policy-Gradient Estimation - Jonathan Baxter and Peter L. Bartlett (vol 15, 2001)
- Reinforcement Learning Using Recursive Least-Squares Methods - Xu, X., He, H. and Hu, D. (vol 16, 2002)

4.2 Code problem

I will create a graphically simple implementation of Tetris, utilising either Java or c++. Since the focus of the project is on reinforcement learning, and not recoding Tetris, the possibility also exists of adopting an existing open-source implementation of Tetris, or possibly even coding a learning engine that can be incorporated into different implementations as desired.

There are numerous considerations to be made, and I will not launch straight into full Tetris. I will rather initially focus on achieving rudimentary learning, before increasing the sophistication of the system. I will also initially use conservative methods and data structures, and get more experimental and daring as my confidence rises.

The most important initial consideration is the representation of the environment, since the agent is fully dependent on the sensory perception we extend to it, and this can heavily impact on the complexity of the environment. The implementation and representation of the 2 remaining defining reinforcement structures (reward function and value function) will need to fulfil their required rolls and interact accordingly.

4.3 Optimisations

This will have to take two forms :

- Coding optimisations : Selecting of data structures and optimal programming techniques
- Structural optimisations : Optimisations inherent to Tetris state space, such as those we as humans utilise

The structural optimisations include symmetry, restricted slices of the environment and considering just the silhouette or the contours of the blocks. Further optimisations will hopefully be accrued over time, and greatly simplify the state space down to something more manageable.

5 Expected Results

An initially untrained unprejudiced persistent agent that graduates from the school of hard knocks and becomes adept at playing Tetris, making decisions and successfully playing within the real time requirements demanded of a human.

6 Possible Extensions

Time permitting, it would be both informative and constructive to try adapting the reinforcement engine devised around Tetris to other similar applications. The complexity of porting the algorithm will obviously depend on the similarities shared by the programs.

7 Bibliography

Light introduction

http://en.wikipedia.org/wiki/Reinforcement_learning

Previous implementations of Tetris using Reinforcement Learning

http://www.math.tau.ac.il/~mansour/rl-course/student_proj/livnat/Tetris.html
http://www.melax.com/Tetris/

Introductory Text : An introduction to reinforcement learning

http://www.cs.ualberta.ca/~sutton/book/ebook/index.html

The reinforcement learning repository of The Journal of Artificial Intelligence Research

http://www.sadl.uleth.ca/nz/cgi-bin/library?a=q&r=1&hs=1&e=q-000-00---0jair--00-0-0--0prompt-

An excellent base for information

http://www-anw.cs.umass.edu/rlr/