From Tetris to Relational Reinforcement Learning

Dr. Yang Gao (gaoy@nju.edu.cn)
Mr. Shen Ge, Mr. Weiwei Wang, Mr. Xingguo Chen
State Key Laboratory for Novel Software Technology,
Nanjing University
Outline

• Tetris
• Learn optimal policy by reinforcement learning (RL)
• RL + function approximation is enough?
• Features of Tetris
• Towards first order logic
• Markov logic networks
• Conclusion
Tetris (1)

- Rewards (scores) = number of cleared lines

Tetris is a falling-blocks puzzle video game originally designed and programmed by Alexey Pajitnov in 1985.
Tetris (2)

- Play the “offline” version of Tetris, where the initial board and piece sequence are known, is **NP-hard**. [Demaine et al., 2003]

- Artificial Tetris player [Ramon and Driessens, 2004]
  - 500,000 lines when they only include information about the falling block.
  - 5,000,000 lines when the next block is considered.
Known algorithms

- Average scores of various algorithms [Szita and Lorincz, 2006]
  - Non-reinforcement learning algorithms
  - Reinforcement learning algorithms
Abstract of Tetris

• State space (S):
  - $2^{200\times7\times4\times10(7)} > 10^{60}$

• Action (A):
  - Drop, turn, right, left

• Goal:
  - Maximize the expected rewards (scores).

Sequence decision problem.
Modeling Tetris

- **Markov Decision Process (MDP)**
  - A set of States: $S$
  - A set of Actions: $A$
  - Reward function: $r : S \times A \to \mathbb{R}$
  - State transition function: The next block's shape is undetermined.
    
    $$ P : S \times A \to S $$

- However, the model of Tetris is unknown in advance. Planning (or optimizing) is infeasible in Tetris.
Learn model or learn optimal policy?

• Learn model
  - By *Monte Carlo sampling*, can learn (or estimate) the model.
  - Given the estimated model, use planning technology to obtain the optimal policy.

• Learn optimal policy
  - By trial-and-error, get some experiences (or samples) \( (s, a, s', r) \)
  - Learn the optimal policy from experiences directly.
Key question: how to predict the long term rewards

• Return function

Discounted - parameter $\gamma < 1$.  
\[
\text{return} = \sum_{i=0}^{\infty} \gamma^i r(s_i, a_i)
\]

Undiscounted or average reward

\[
\text{return} = \lim_{N \to \infty} \frac{1}{N} \sum_{i=0}^{N-1} r(s_i, a_i)
\]

• Bellman equation

  - Using iterative method to compute the return (value) function
Bellman equation given the determined policy $\Pi$

The basic idea (in one episode):
\[
R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} \ldots
\]
\[
= r_{t+1} + \gamma \left( r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} \ldots \right)
\]
\[
= r_{t+1} + \gamma R_{t+1}
\]

So, in many episodes:
\[
V^\pi(s) = E_\pi \{ R_t \mid s_t = s \}
\]
\[
= E_\pi \{ r_{t+1} + \gamma V( s_{t+1} ) \mid s_t = s \}
\]

Or, without the expectation operator:
\[
V^\pi(s) = \sum_{s'} P_{ss'}^a \left[ R_{ss'}^a + \gamma V^\pi(s') \right]
\]

is unknown
Temporal-Difference learning

Simple Monte Carlo method:
\[ V(s_t) \leftarrow V(s_t) + \alpha \left[ R_t - V(s_t) \right] \]
\[ \text{target: the actual return after time } t. \]

The simplest TD method, TD(0):
\[ V(s_t) \leftarrow V(s_t) + \alpha [r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \]
\[ \text{target: an estimate of the return.} \]

The detail materials can be found in the talk of MLA'04.
**Q-learning**

- For each $(s, a)$, calculate/predict the $Q$ values.

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[ r_{s,a}^s + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$

- The optimal policy:

$$\Pi^* (s) \leftarrow \arg \max_a Q(s, a)$$
Average reward reinforcement learning algorithm

• Average reward $G$-learning algorithm

\[
G(s,a) \leftarrow G(s,a) + \alpha \left[ r_{s,s'}^a - g(s_0) + \max_{a'} G(s',a') - G(s,a) \right]
\]

if $s = s_0$, $g(s_0) \leftarrow \max_a G(s_0,a)$

reference state

The detail materials can be found in the talk of MLA’06.
How to speed up the learning process

- **Problem:** large state space
  - State space: 10*20 grids, 7 shapes and 10 locations.
  - Action space: 4 actions.

- **Solution:** in similar state-action pairs, the Q-value may be similar.

- **Technical points:** using function approximation to general the Q-values.

**Question:** after learn a Q(s,a), when will visit the state ‘s’ again?
RL + function approximation

- Neural network et al.

Subset of states \( Q_0 \) \( \rightarrow M(\Gamma(M(Q_0))) \) \( \rightarrow \Gamma(M(\Gamma(M(Q_0)))) \) \( \rightarrow \Gamma(M(\Gamma(M(Q_0))))) \rightarrow \cdots \)

Generalization of the value function to the entire state space

\( Q(s, a) \)
Large Scale MDP
Markov Decision Process
Tetris
Model

RL + FA
Average RL Algorithm
Algorithm

MLA’07, Nanjing
Features of states & actions

• Relative features
  - Height of wall (max, avg, min)
  - Number of Holes
  - Height difference adjacent cols
  - Canyon (width, height)
  - ...

• Macro actions
  - Fits
  - Increases height, ...
  - Number of deleted lines

Good features beat good learning! [Feng, MLA07]
Some discussions and thinking...

• Classical RL
  - Use look-up table
• RL + FA
  - Use function to generalize the Q-table
• Relative features
  - Use features to generalize the Q-table

Is it enough?
Relational domain

• **Challenges** [Tadepalli et al., 2004]
  - Function approximation
  - Prior Knowledge
  - Generalization across objects
  - Transfer learning across tasks
  - Run-time planning and reasoning
Relational reinforcement learning

• RRL
  - Reinforcement learning + relational representation

• Relational representation
  - Represents value function as a first order logic regression tree

• Algorithms
  - TG algorithm [Driessens et al, 2001]
  - RIB (instance based algorithm) [Driessens and Ramon, 2003]
  - KBR (kernel based algorithm) [Gartner et al, 2003]
Decision Tree

- Each **internal node** of a decision tree contains a **test**.
- Decision trees **partition** the whole **example space** and **assign class values** to each example.
- **Make prediction**
  - Starts in the root of the tree
  - Applies a test to the example
  - Propagates the example to the corresponding subtree
  - Leaf is the prediction
First order logical decision tree

- Differences between LDT and DT
  - Example: a relational database
  - Test: query
- Example 1: (s, a, Q)
  - Qvalue(1)
  - WidCanyon(b, 1)
  - HeightCanyon(b, 4)
  - Hole(3,2)
  - NumHoles(1)
  - Hight(3,3)
  - Shape(a, 'O')
  - Drop(1,'O',vert)

Q-value → Q-value

{ state → action }
For example: a LDT

\[ \text{Canyon(S)} \]

\[ \text{Shape(B,'O')} \]

\[ Q_{value}=0.8 \]

\[ \text{Drop(1,'O',vert)} \]
\[ \text{Drop(1,'O',hor)} \]

MLA'07, Nanjing
Relational RL algorithm

- **RRL algorithm** [Driessens et al, 2001]
  - 0. Represent the state and action with relational method, initialize the Q-values
  - 1. Run the first episode
    - Choose the action randomly
  - 2. Obtain examples \((s,a,Q)\)
  - 3. Use TG algorithm to expand tree
  - 4. Run next episode
    - Choose the action according to the tree
    - Update the Q-value
  - 5. Return step 2
TG algorithm (1)

• Build first order logical tree

  - Create an empty leaf
  - While (examples available)
    • Sort example down to leaf
    • Update statistics in leaf
    • If (split needed)
      - Create two empty leafs

• The heuristical rule is same as in C4.5.
Example 1

State:
WidCanyon(1,2),---column 1, width 2
HeightCanyon(1,3),---column 1, height 3
Hole(3,2),
NumHoles(1),
Height(3,3),
Height(4,3),
Height(5,3),

Action:
Drop(1,'O',Vertical)---Put Shape 'O' on column 1 with direction Vertical

Qvalue:
Qvalue(1)---1 line is cleared

MLA'07, Nanjing
Example 2

State:
WidCanyon(1,2),---column 1, width 2
HeightCanyon(1,3),---column 1, height 3
Hole(3,2), Hole(5,1)
NumHoles(2),
Height(1,1), Height(2,1),
Height(3,4), Height(4,3),
Height(5,3),
Action:
Drop(1,'O',Vertical)---Put Shape 'O' on column 1 with direction Vertical
Qvalue:
Qvalue(1)---1 line is cleared
Example 3

State:
WidCanyon(1,1),---column 1, width 1
HeightCanyon(1,3),---column 1, height 3
Hole(3,2),
NumHoles(1),
Height(2,3), Height(3,3),
Height(4,3), Height(5,3),

Action:
Drop(1,'L',Vertical)---Put Shape 'L' on column 1 with direction Vertical

Qvalue:
Qvalue(0)---No line is cleared
Example 4

State:
WidCanyon(1,2),---column 1, width 2
HeightCanyon(1,3),---column 1, height 4
Hole(3,1),
NumHoles(1),
Height(3,3),
Height(4,3),
Height(5,3),

Action:
Drop(1,L,Vert)---Put Shape 'L' on column 1 with direction Vertical

Qvalue:
Qvalue(0)---No line is cleared
How to build first order logical tree?

\[ \text{WidCanyon}(A, B), \text{HeightCanyon}(C, D), \text{NumHoles}(E), \text{Drop}(F, G, H) \]

Qvalue = 0 (example 3)
Qvalue = 1 (example 1, 2)
Qvalue = 0 (example 4)
TG algorithm (2)

• TG algorithm
  - When stop to split the leaf node?

• Tree updated algorithm
  - New examples, update the tree incrementally

• Test to choose an action
  - Test all possible actions, combine any possible example, according to the tree to get their Q-values.
Logical MDP

Large Scale MDP

Unknown MDP

Relational RL

RL + FA

Average RL Algorithm

Model

Algorithm

Tetris

MLA’07, Nanjing
Prior Knowledge

- Formula 1
  - 'If exist a canyon whose width is 2 and the shape of dropping block is I, put the block in the canyon, then the canyon’s width is 1.'
Markov logic networks

• What is MLN?
  - First order logic
    • Constants, variables, functions, predicates, formulas
  - Markov network
Explain

• Prior knowledge
  - 'If exist a canyon whose width is 2 and the shape of dropping block is I, put the block in the canyon, then the canyon’s width is 1.'

• First-order logic

\[\exists x, \exists y \quad \text{WidCanyon}(x, 2) \land \text{BlockShape}(y, O) \land \text{Drop}(y, x) \Rightarrow \text{WidCanyon}(x, 1)\]

• Clausal form

\[
\neg \text{WidCanyon}(x, 2) \lor \neg \text{BlockShape}(y, I) \lor \neg \text{Drop}(y, x) \lor \text{WidCanyon}(x, 1)
\]

• Weight
  - 0.8
Markov logic network

- WidCanyon(x,2)
- BlockShape(y,1)
- Drop(y,x)
- WidCanyon(x,1)
Example

State:
WidCanyon(1,1),---column 1, width 1
HeightCanyon(1,2),---column 1, height 2
Height(2,1),
Height(3,2),
Height(4,2),
Height(5,1)

Action:
Drop(1,'I',Vertical)---Put Shape 'O' on column 1 with direction Vertical

Qvalue:
Qvalue(1)---1 line is cleared
How to predict?

Compute the probability of \texttt{WidCanyon(x,1)} using MLN

Then compute the Q-value by LDT

Choose the predict (action) to maximize the Q-value
Conclusion

• **Traditional reinforcement learning**
  - Too large state space, to re-visit it.
• **RL + FA**
  - Propagate the Q values to similar states.
• **Features**
  - Similar states have same features.
• **Relational RL**
  - Compute which feature is most important.
• **Markov logic network**

Doing the task is not difficult, Describing the task is difficult.
Thanks ...


• [Driessens et al, 2001] Driessens K., Ramon J., Blockeel H. Speeding up relational reinforcement learning through the use of an incremental first order decision tree learner Lecture Notes in Computer Science 2167.

