High-Dimensional Function Approximation for Knowledge-Free Reinforcement Learning: a Case Study in SZ-Tetris

Wojciech Jaśkowski       Marcin Szubert
Pawel Liskowski          Krysztof Krawiec

Institute of Computing Science

POZNAŃ UNIVERSITY OF TECHNOLOGY

July 14, 2015
RL Perspective

1. **direct policy search** (e.g. EAs), good for Tetris, Othello
2. **value function-based methods** (e.g. TD), good for Backgammon

Comparison: Many factors involved: randomness, environment observability, problem structure, etc.
Introduction

RL Perspective

1. direct policy search (e.g. EAs), good for Tetris, Othello
2. value function-based methods (e.g. TD), good for Backgammon

Comparison: Many factors involved: randomness, environment observability, problem structure, etc.

Here: Policy Representation

- For High dimensions only value function-based methods?
- Modern EAs capable of searching high-dimensional spaces, e.g., VD-CMA-ES, R1-NES.
Introduction

RL Perspective

1. **direct policy search** (e.g. EAs), good for Tetris, Othello
2. **value function-based methods** (e.g. TD), good for Backgammon

Comparison: Many factors involved: randomness, environment observability, problem structure, etc.

Here: Policy Representation

- For High dimensions only value function-based methods?
- Modern EAs capable of searching high-dimensional spaces, e.g., VD-CMA-ES, R1-NES.

Research Question  How these modern EAs compare to value function-based methods for high-dimensional policy representations?
SZ-Tetris Domain

**SZ-Tetris**

- a single-player stochastic game,
- a constrained variant of Tetris, a popular yardstick in RL
- devised to studying ‘key problems of reinforcement learning’
- 10 × 20 board
- 17 actions: position + rotation
- 1 point for clearing a line
Hard for value function-based methods

There are many RL algorithms for approximating the value functions. **None of them really work on (SZ-)Tetris**, they do not even come close to the performance of the evolutionary approaches.¹

Hard for value function-based methods

There are many RL algorithms for approximating the value functions. **None of them really work on (SZ-)Tetris, they do not even come close to the performance of the evolutionary approaches.**

Not easy for direct search methods

Cross Entropy Method (ca. 117) < hand-coded policy: (ca. 183.6)

---

Hard for value function-based methods

There are many RL algorithms for approximating the value functions. **None of them really work on (SZ-)Tetris**, they do not even come close to the performance of the evolutionary approaches.¹

Not easy for direct search methods

Cross Entropy Method (ca. 117) < hand-coded policy: (ca. 183.6)

Need for better function approximator

**Challenge #1:** Find a sufficiently good feature set (...). A feature set is sufficiently good if CEM (or CMA-ES, or genetic algorithms, etc.) is able to learn a weight vector such that the resulting preference function reaches at least as good results as the hand-coded solution.¹

State-Evaluation Function and Action Selection

- **Known model → we use state-evaluation function**

\[ V : S \rightarrow \mathbb{R} \]

- **Greedy policy w.r.t \( V \):**

\[ \pi(s) = \arg\max_{a \in A} V(T(s,a)) \]

where \( T \) is a transition model.

**Evaluation functions:**

1. **state-value function** (estimates the expected future scores from a given state),

2. **state-preference function** (no interpretation, larger is better)
Function Approximation

\[ 2^{20 \times 10} \approx 10^{60} \text{ states (upper bound)} \rightarrow \]

we need a function approximator:

\[ V_\theta : S \rightarrow \mathbb{R} \]

Task: learn the best set of parameters \( \theta \).
Weighted Sum of Hand-Designed Features $\phi$

Bertsekas & Ioffe (B&I)

1. Height $h_k$ of the $k$th column of the board, $k = 1, \ldots, 10$.
2. Absolute difference between the heights of the consecutive columns.
3. Maximum column height $\max h$.
4. Number of ‘holes‘ on the board.

Linear evaluation function of features:

$$V_\theta(s) = \sum_{i=1}^{21} \theta_i \phi_i(s),$$
Systematic $n$-Tuple Network

### LUT

<table>
<thead>
<tr>
<th>Index</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>3.04</td>
</tr>
<tr>
<td>0001</td>
<td>-3.90</td>
</tr>
<tr>
<td>0010</td>
<td>-2.14</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td>-2.01</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td>6.12</td>
</tr>
<tr>
<td>1111</td>
<td>3.21</td>
</tr>
</tbody>
</table>

Successful for

2. Connect-4 [Thill, 2012],
3. 2048 [Szubert, 2015]

- Linear weighted function of (a large number of) binary features
- Computationally efficient

$$V_\theta(s) = \sum_{i=1}^{m} V^i(s) = \sum_{i=1}^{m} \text{LUT}^i \left[ \text{index} \left( s_{loc_1}, \ldots, s_{loc_{ini}} \right) \right]$$
Systematic $n$-tuple Network

<table>
<thead>
<tr>
<th>0123</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>3.04</td>
</tr>
<tr>
<td>0001</td>
<td>−3.90</td>
</tr>
<tr>
<td>0010</td>
<td>−2.14</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1100</td>
<td>−2.01</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1110</td>
<td>6.12</td>
</tr>
<tr>
<td>1111</td>
<td>3.21</td>
</tr>
</tbody>
</table>

LUT

Systematically cover the board with:

1. $3 \times 3$-tuples (size = 9),
   \[ |\theta| = 72 \times 2^9 = 36864 \]

2. $4 \times 4$-tuples (size = 16),
   \[ |\theta| = 68 \times 2^{16} = 4456448 \]
ESs maintaining a multi-variate Gaussian probability distribution: $\mathcal{N}(\mu, \Sigma)$:

1. Cross-Entropy Method [CEM, Rubinstein, 2004]:

   - full matrix $\Sigma$, smart self-adaptation ($O(n^2)$)

3. CMA-ES for high dimensions [VD-CMA-ES, Akimoto, 2014]
   - $\Sigma = D(I + vv^T)D$, where $D$ – diagonal matrix, $v \in \mathbb{R}^n$ ($O(n)$)
Learning of $V$

- After a move the agents gets a new experience $\langle s, a, r, s' \rangle$
- Modify $V$ in response to the experience by Sutton’s TD(0) update rule:

$$V(s) \leftarrow V(s) + \alpha (r + V(s') - V(s))$$

$\alpha$ — learning rate

General Idea

- Reconcile values of neighboring states $V(s)$ and $V(s')$, to make in the long run Bellman equation hold:

$$V(s) = \max_{a \in A(s)} \left( R(s, a) + \sum_{s' \in S} P(s, a, s') V(s') \right)$$
Results for evolutionary methods

<table>
<thead>
<tr>
<th>B&amp;I Features</th>
<th>3x3 Tuple Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>average score (cleared lines)</td>
<td>generation</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>150</td>
<td>150</td>
</tr>
<tr>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>250</td>
<td>250</td>
</tr>
<tr>
<td>300</td>
<td>300</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>150</td>
<td>150</td>
</tr>
<tr>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>250</td>
<td>250</td>
</tr>
<tr>
<td>300</td>
<td>300</td>
</tr>
</tbody>
</table>

117.0 ± 6.3 CEM
124.8 ± 13. CMA-ES
219.7 ± 2.8 VD-CMA-ES for 3 × 3
Results for TD(0)

<table>
<thead>
<tr>
<th>Training Games (x1000)</th>
<th>Average Score (Cleared Lines)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>3x3 Tuple Network</strong></td>
<td></td>
</tr>
<tr>
<td><strong>4x4 Tuple Network</strong></td>
<td></td>
</tr>
</tbody>
</table>

\[
183.3 \pm 4.3 \text{ TD}(0) \text{ for } 3 \times 3 \\
218.0 \pm 5.2 \text{ TD}(0) \text{ for } 4 \times 4 \\
219.7 \pm 2.8 \text{ VD-CMA-ES for } 3 \times 3 
\]
## Results Summary

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Function</th>
<th>Features</th>
<th># Games</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hand-coded</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>183.6 ± 1.4</td>
</tr>
<tr>
<td>CEM</td>
<td>B&amp;I</td>
<td>21</td>
<td>20 mln</td>
<td>117.0 ± 6.3</td>
</tr>
<tr>
<td>CMA-ES</td>
<td>B&amp;I</td>
<td>21</td>
<td>20 mln</td>
<td>124.8 ± 13.1</td>
</tr>
<tr>
<td>VD-CMA-ES</td>
<td>3×3-tuple network</td>
<td>36 864</td>
<td>100 mln</td>
<td>219.7 ± 2.8</td>
</tr>
<tr>
<td>TD(0)</td>
<td>3×3-tuple network</td>
<td>36 864</td>
<td>4 mln</td>
<td>183.3 ± 4.3</td>
</tr>
<tr>
<td>TD(0)</td>
<td>4×4-tuple network</td>
<td>4 456 448</td>
<td>4 mln</td>
<td>218.0 ± 5.2</td>
</tr>
</tbody>
</table>

Larger variance with TD(0) $4 \times 4 \rightarrow$ best strategy (nearly 300 points on average).
Best agent

play
play

GAME OVER

Next Piece:
Score: 183

Next Piece:
Score: 172

Press Enter to Start Again
Summary

**RL Perspective**

1. High-dimensional representation (systematic $n$-tuple network) to:
   - Make TD work at all on this problem

2. VD-CMA-ES vs. TD:
   - VD-CMA-ES can work with tens of thousands parameters (**needs large populations**)
   - CEM $< \text{TD} < \text{VD-CMA-ES}$ (on 3x3)
   - TD vs. VD-CMA-ES $\rightarrow$ memory vs. time trade-off

---

Summary

RL Perspective

1 High-dimensional representation (systematic $n$-tuple network) to:
   - Make TD work at all on this problem

2 VD-CMA-ES vs. TD:
   - VD-CMA-ES can work with tens of thousands parameters (needs large populations)
   - CEM $<$ TD $<$ VD-CMA-ES (on 3x3)
   - TD vs. VD-CMA-ES $\rightarrow$ memory vs. time trade-off

SZ-Tetris Perspective

1 The best player to date (nearly 300 lines on average) $>$ hand-coded strategy (183 lines).

2 Systematic $n$-Tuple Networks solve the Challenge #1 posed by Szita and Szepesvári

\footnote{Source code: http://github.com/wjaskowski/gecco-2015-sztetris}