Learning Tetris Using the Noisy Cross-Entropy Method

István Szita [1], András Lőrincz [2]

Department of Information Systems, Eötvös Loránd University
Pázmány Péter sétány 1/C, Budapest, Hungary H-1117

Abstract

The cross-entropy method is an efficient and general optimization algorithm. However, its applicability in reinforcement learning seems to be limited although it is fast, because it often converges to suboptimal policies. A standard technique for preventing early convergence is to introduce noise. We apply the noisy cross-entropy method to the game of Tetris to demonstrate its efficiency. The resulting policy outperforms previous RL algorithms by almost two orders of magnitude, and reaches over 300,000 points on average.

Key words: tetris, cross-entropy method, reinforcement learning

1 Introduction

Tetris was created in 1985 by Alexey Pajitnov, and since then it is one of the most popular computer games (for a detailed description of the game, see e.g. [Fahey, 2003]). Despite its simple rules, playing the game well requires a complex strategy and lots of practice. Furthermore, Demaine et al. have shown that Tetris is hard in a mathematical sense as well [Demaine et al., 2003]: finding the optimal strategy is NP-hard even if the sequence of tetrominoes is known in advance.

These properties make Tetris an appealing benchmark problem for testing reinforcement learning (and other machine learning) algorithms. [3]

[1] email: szityu@eotvos.elte.hu
[2] Corresponding author, email: andras.lorincz@elte.hu, WWW home page: http://nipg.inf.elte.hu
[3] See Table 1 for a summary of known approaches.

Reinforcement learning algorithms are quite effective in solving a variety of complex sequential decision problems. Despite this, RL approaches to Tetris have performed surprisingly poorly so far. The aim of this technical note is to show how to improve RL for this hard combinatorial problem. We put forth a modified version of the cross-entropy (CE) method [de Boer et al., 2004] here.

2 Applying the Cross-entropy Method to Tetris

2.1 The Value Function

Following the approach of Bertsekas and Tsitsiklis [Bertsekas and Tsitsiklis, 1996], we shall learn state-value functions that are linear combination of several basis functions. We use 22 such basis functions: maximal column height, individual column heights, differences of column heights and the number of holes (for a more detailed description, see [Bertsekas and Tsitsiklis, 1996]). More formally, if s denotes a Tetris state, and $\phi_i(s)$ is the value of basis function i in this state, then according to weight vector w, the value of state s is

$$V_w(s) := \sum_{i=1}^{22} w_i \phi_i(s). \tag{1}$$

2.2 Action Selection

For a given value function V_w, action selection is also done in a manner similar to Bertsekas and Tsitsiklis. To decide the place of a tetromino, we try to put it in every column and in every possible rotation, erase full rows (if there is any), and calculate the value of the resulting state, according to V_w. Finally, we choose the column and rotation with the highest value.

2.3 The Cross-Entropy Method

The cross-entropy (CE) method is a general algorithm for (approximately) solving global optimization tasks of the form

$$w^* = \arg\max_w S(w). \tag{2}$$

where S is a general objective function (e.g. we do not need to assume continuity). While most optimization algorithms maintain a simple solution candidate

w_t in each time step, the main idea of CE is to maintain a *distribution* of possible solutions, and update this distribution accordingly. An excellent reading on the details of the CE method is written by de Boer et al. [de Boer et al., 2004], here we give only a very brief overview.

The CE method starts with a parametric family of distributions $\mathcal{F}$ and an initial distribution of samples $f_0 \in \mathcal{F}$. Under this distribution, the probability of drawing a high-valued sample (value $\geq \gamma^*$) is presumably very low, therefore finding such samples by naive sampling is intractable. For any γ, let $g_{\geq\gamma}$ be the uniform distribution over the γ-level set , i.e. the set of samples having value greater than γ. If we could find the element $f_1 \in \mathcal{F}$ closest to $g_{\geq\gamma}$ w.r.t. the cross-entropy measure (also called the Kullback-Leibler distance), then we could use f_1 instead of f_0, under which γ-valued samples have greater probability. The main idea of the CE method is that for many distribution families, the parameters of f_1 can be estimated from samples of f_0. However, this estimation is tractable only if the probability of the γ-level set is not very low w.r.t. f_0, so we cannot directly compute the $\mathcal{F}$-distribution closest to $g_{\geq\gamma^*}$, but have to proceed iteratively: select a γ_0 appropriate for f_0, update the distribution parameters to obtain f_1, select γ_1, and so on, until we do not reach a sufficiently large γ^*. Below we sketch the special case when the distribution of parameters is Gaussian.

Let the distribution of the parameter vector at iteration t be $f_t \sim N(\mu_t, \sigma_t^2)$. After drawing n sample vectors $w_1, \ldots, w_n$ and evaluating their value $S(w_1), \ldots, S(w_i)$, we sort them and select the best $\lfloor \rho \cdot n \rfloor$ samples, where $0 < \rho < 1$ is the selection ratio. This is equivalent to setting $\gamma_t = S(w_{\lfloor \rho \cdot n \rfloor})$. Denoting the set of indices of the selected samples by $I \subseteq \{1, 2, \ldots, n\}$, the mean of the distribution is updated using

$$\mu_{t+1} := \frac{\sum_{i \in I} w_i}{|I|}, \tag{3}$$

and the deviation update is

$$\sigma_{t+1}^2 := \frac{\sum_{i \in I} (w_i - \mu_{t+1})^T (w_i - \mu_{t+1})}{|I|}. \tag{4}$$

2.4 The Cross-Entropy Method and Reinforcement Learning

The CE method has been applied to RL by several researchers: Menache et al. used a combination of radial basis functions as the value function of a maze task, and applied CE to learn the RBF parameters [Menache et al., 2005]; while Mannor et al. used CE to tune a parametrized policy [Mannor et al., 2003]. We apply CE to learn the weights of the basis functions, drawing each weight from an independent Gaussian distribution.

2.5 Preventing Early Convergence

Preliminary investigations showed that applicability of CE to RL problems is restricted severely by the phenomenon that the distribution concentrates to a single point too fast (see e.g. experiment 1 below). To prevent this, we adapt a trick frequently used in particle filtering: at each iteration, we add some extra noise to the distribution. In our case, this means that instead of (4), we use

$$\sigma_{t+1}^2 := \frac{\sum_{i \in I}(w_i - \mu_{t+1})^T (w_i - \mu_{t+1})}{|I|} + Z_{t+1}, \tag{5}$$

where Z_{t+1} is dependent on the iteration number, but is otherwise a constant vector.

3 Experiments

In the experiments we used the standard Tetris game described in [Bertsekas and Tsitsiklis, 1996], scoring 1 point for each cleared row. Each parameter had an initial distribution of $N(0, 100)$. We set $n = 100$ and $\rho = 0.1$. Each drawn sample was evaluated by playing a single game using the corresponding value function. After each iteration, we updated distribution parameters using (3) and (5), and evaluated the mean performance of the learnt parameters. This was accomplished by playing 30 games using the value function V_{μ_t}, and averaging the results. [4]

In our first experiment we tested the original CE method (corresponding to $Z_t = 0$). As expected, deviations converge to 0 too fast, so the mean performance settles at about 20,000 points. In the next test we used a constant noise rate of $Z_t = 4$, raising mean performance to a 70,000 point level. Our analysis showed that further improvement was counteracted by the amount of noise, which was *too high*. High noise prevented convergence of the distributions. Therefore in the last experiment we applied a decreasing amount of noise, $Z_t = \max(5 - \frac{t}{10}, 0)$. [5] With this setting, average score exceeded 300,000 points by the end of episode 50, and the best score exceeded 800,000 points. Results are summarized in Fig. 1 and also in Table 1. The distribution of scores for a fixed value function is conjectured to have an exponential distribution [Fahey, 2003]. Using the Kolmogorov-Smirnov test, this conjecture was

[4] The large number of evaluation games was necessary because Tetris strategies have large performance deviations. [Fahey, 2003]

[5] In the last two experiments, noise parameters were selected in an *ad hoc* manner, no optimization was carried out.

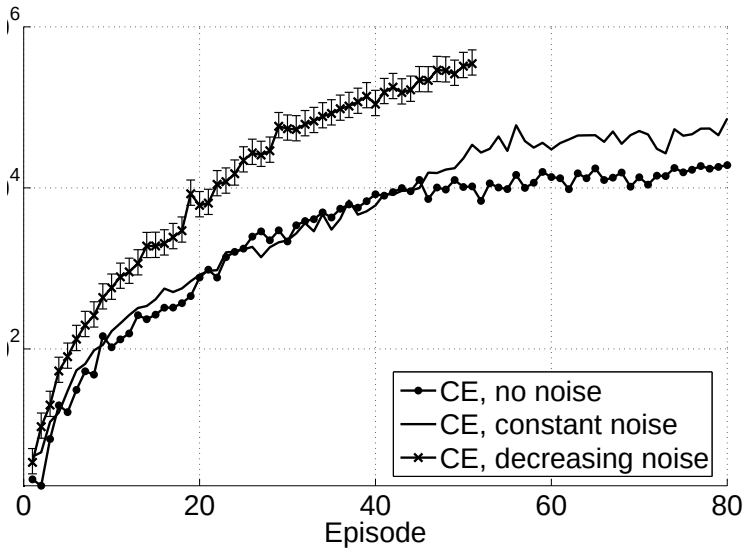

Fig. 1. **Tetris scores.** Mean performance of 30 evaluation runs versus iteration number. Note the log scale.

confirmed for each data point with 95% confidence. Assuming an exponential score distribution, we calculated the 95% confidence intervals of the mean (using Matlab's `expfit`). For better visibility, confidence intervals have been plotted only for the last experiment.

3.1 Comparison to Previous Work

Tetris was chosen as a benchmark problem by many researchers to test their RL algorithms. Table 1 summarizes results known to us, comparing them to our algorithm, and to two state-of-the-art non-RL algorithms as well. One of the non-RL algorithms applied a hand-coded policy, while the other used a genetic algorithm.

The comparison shows that our method improves on the performance of the best known RL algorithm by almost two orders of magnitude, and gets close to the best algorithms at present, the hand-coded algorithm and the GA.

Regarding running times, it is not easy to make a sensible comparison. One reason is that evaluation time scales linearly with the score achieved, so the better the algorithm is, the more CPU time it requires for evaluation. The *number of games played during learning* seems to be a better measure, and is shown in column 3 of Table 1. However, this measure is also somewhat misleading, because a low number of played games usually means that the learning algorithm gets stuck (or starts to deteriorate) fast. It should be noted that the computational overhead of the CE method is negligible, as it only requires the generation of a random vector for each sample and the computation of (3)

Table 1
Average Tetris scores of various algorithms.

Method / Reference	Mean Score	# games played during learning
Non-reinforcement learning		
Hand-coded (P. Dellacherie) [Fahey, 2003]	631,167	n.a.
GA [Böhm et al., 2004]	586,103	3000
Reinforcement learning		
RRL-KBR [Ramon and Driessens, 2004]	≈ 50	120
Policy iteration [Bertsekas and Tsitsiklis, 1996]	3,183	1500
LSPI [Lagoudakis et al., 2002]	$< 3{,}000$	≈ 17
LP+Bootstrap [Farias and van Roy, 2006]	4,274	n.a.
Natural policy gradient [Kakade, 2001]	$\approx 6{,}800$	≈ 10000
CE+RL	21,252	10000
CE+RL, constant noise	72,705	10000
CE+RL, decreasing noise	348,895	5000

and (5) once per episode.

3.2 Further Improvement?

The noisy CE method leaves plenty of room for further improvement: firstly, the parameters of the noise greatly affect performance (e.g. using hyperbolically decreasing noise, $Z_t = c/t$), but no optimization was carried out, partly because of the enormous running times (more than one month). However, optimization time may be shortened considerably by using the conjecture that the length of a game can be approximated from its starting sequence [Fahey, 2003].

Secondly, Böhm et al. showed that by using an exponential value function instead of the linear one, huge improvements can be achieved [Böhm et al., 2004]. The CE method can easily incorporate such value functions as well and that may lead to improvements for RL, too. Last but not

least, performance may be further improved by using more carefully crafted basis functions, as was explored by to Böhm and colleagues [Böhm et al., 2004].

References

[Bertsekas and Tsitsiklis, 1996] Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific.

[Böhm et al., 2004] Böhm, N., Kókai, G., and Mandl, S. (2004). Evolving a heuristic function for the game of Tetris. In Scheffer, T., editor, *Proc. Lernen, Wissensentdeckung und Adaptivität LWA - 2004*, pages 118–122.

[de Boer et al., 2004] de Boer, P., Kroese, D., Mannor, S., and Rubinstein, R. (2004). A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67.

[Demaine et al., 2003] Demaine, E. D., Hohenberger, S., and Liben-Nowell, D. (2003). Tetris is hard, even to approximate. In *Proc. 9th International Computing and Combinatorics Conference (COCOON 2003)*, pages 351–363.

[Fahey, 2003] Fahey, C. P. (2003). Tetris AI. `http://www.colinfahey.com`.

[Farias and van Roy, 2006] Farias, V. F. and van Roy, B. (2006). *Probabilistic and Randomized Methods for Design Under Uncertainty*, chapter Tetris: A Study of Randomized Constraint Sampling. Springer-Verlag UK.

[Kakade, 2001] Kakade, S. (2001). A natural policy gradient. In *Advances in Neural Information Processing Systems (NIPS 14)*, pages 1531–1538.

[Lagoudakis et al., 2002] Lagoudakis, M. G., Parr, R., and Littman, M. L. (2002). Least-squares methods in reinforcement learning for control. In *SETN '02: Proceedings of the Second Hellenic Conference on AI*, pages 249–260, London, UK. Springer-Verlag.

[Mannor et al., 2003] Mannor, S., Rubinstein, R. Y., and Gat, Y. (2003). The cross-entropy method for fast policy search. In *Proc. International Conf. on Machine Learning (ICML 2003)*, pages 512–519.

[Menache et al., 2005] Menache, I., Mannor, S., and Shimkin, N. (2005). Basis function adaption in temporal difference reinforcement learning. *Annals of Operations Research*, 134(1):215–238.

[Ramon and Driessens, 2004] Ramon, J. and Driessens, K. (2004). On the numeric stability of gaussian processes regression for relational reinforcement learning. In *ICML-2004 Workshop on Relational Reinforcement Learning*, pages 10–14.