Policy Search and Black-Box Policy Optimization

Sanjiban Choudhury
CRISIS !!!

Errors in neural network get amplified by dynamic programming (Bootstrapping)
QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation
To hell with Value Estimates!

Trust ONLY actual Returns
What if we focused on finding good policies … ?
Sometimes a policy is waaaaayy simpler than the value.
Can we just focus on finding a good policy?

\[ \pi_\theta : S_t \rightarrow a_t \]

Learn a mapping from states to actions

Roll-out policies in the real-world to estimate value
The Game of Tetris
What’s a good policy representation for Tetris?

(4 rotations)*(10 slots) - (6 impossible poses) = 34

\[ \pi_\theta : s_t \rightarrow a_t \]
Activity!
Think-Pair-Share

Think (30 sec): Ideas for how to represent policy for tetris?

Pair: Find a partner

Share (45 sec): Partners exchange ideas
Some inspiration for Tetris policy

Until 2008, the best artificial Tetris player was handcrafted, as reported by Fahey (2003). Pierre Dellacherie, a self-declared average Tetris player, identified six simple features and tuned the weights by trial and error.
Dellacherie Features

The contribution of the last piece to the cleared lines time the number of cleared lines.

The number of filled cells adjacent to the empty cells summed over all rows.

A well is a succession of empty cells and the cells to the left and right are occupied.
A *magic* formula ?!? 

\[-4 \times \text{holes} - \text{cumulative wells} - \text{row transitions} - \text{column transitions} - \text{landing height} + \text{eroded cells}\]
A magic formula ?!

\(-4 \times \text{holes} - \text{cumulative wells} - \text{row transitions} - \text{column transitions} - \text{landing height} + \text{eroded cells}\)

This linear evaluation function cleared an average of 660,000 lines on the full grid ...

... In the simplified implementation used by the approaches discussed earlier, the games would have continued further, until every placement would overflow the grid. Therefore, this report underrates this simple linear rule compared to other algorithms.
Can YOU do better than Dellacherie?
The Goal of Policy Optimization

\[ \pi_\theta(s) = \arg \min_a \theta^T f(s, a) \]  

\[ \min J(\theta) = \sum_{t=0}^{T-1} \mathbb{E}_{\pi_\theta} c(s_t, a_t) \]

#Think of \( f(s,a) \) being dellacherie features

#Think of \( c(s,a) \) as -num_rows_cleared
Cross Entropy

If you were ever stranded on an island ...
Credit: https://blog.otoro.net/2017/10/29/visual-evolution-strategies/

**Green**: Mean of distribution

**Blue**: Samples from distribution

**Red**: Best solution found so far
Let’s formalize!
The Cross Entropy Algorithm
The Cross Entropy Algorithm

\[ D_\theta \]

\[ \text{Init } D_\theta \]

Sample \( k \) times to get \{ \Theta_i \}_{i=1}^k \]
The Cross Entropy Algorithm

Init: $D_\theta$

Sample k times to get $\{x_i \mid j_{1:k}\}$

Evaluate each $\theta_i$:
- Execute policy multiple times
The Cross Entropy Algorithm

Evaluate each \( \theta_i \):

- Execute policy multiple times
The Cross Entropy Algorithm

Evaluate each $\theta_i$:
- Execute policy multiple times

Find top $E$ states
(e.g., 95%)
The Cross Entropy Algorithm

Evaluate each $\theta_i$:
- Execute policy multiple times

Find top $E$ elites (e.g. 25%)

Fit a new distribution $D_{\theta}$
Cross Entropy for Gaussian

Gaussian Distribution

\[ D_\theta := \mathcal{N}(\mu, \Sigma) \]

Mean

\[ \mu^t = \frac{1}{e} \sum_{i=1}^{e} \theta_i \]

Variance

\[ \Sigma^t = \frac{1}{e} \sum_{i=1}^{e} (\theta_i - \mu^t)^2 \]
Does it work?

Learning Tetris Using the Noisy Cross-Entropy Method

István Szita  
szityu@eotvos.elte.hu

András Lőrincz  
andras.lorincz@elte.hu

Department of Information Systems, Eötvös Loránd University, Budapest, Hungary  
H-1117

The cross-entropy method is an efficient and general optimization algorithm. However, its applicability in reinforcement learning (RL) seems to be limited because it often converges to suboptimal policies. We apply noise for preventing early convergence of the cross-entropy method, using Tetris, a computer game, for demonstration. The resulting policy outperforms previous RL algorithms by almost two orders of magnitude.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Grid Size</th>
<th>Lines Cleared</th>
<th>Feature Set Used</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tsitsiklis &amp; Van Roy (1996)</strong></td>
<td>Approximate value iteration</td>
<td>$16 \times 10$</td>
<td>$30$</td>
</tr>
<tr>
<td>Bertsekas &amp; Tsitsiklis (1996)</td>
<td>$\lambda$ - PI</td>
<td>$19 \times 10$</td>
<td>$2,800$</td>
</tr>
<tr>
<td>Lagoudakis et al. (2002)</td>
<td>Least-squares PI</td>
<td>$20 \times 10$</td>
<td>$\approx 2,000$</td>
</tr>
<tr>
<td>Kakade (2002)</td>
<td>Natural policy gradient</td>
<td>$20 \times 10$</td>
<td>$\approx 5,000$</td>
</tr>
<tr>
<td>Dellacherie</td>
<td>Hand tuned</td>
<td>$20 \times 10$</td>
<td>$660,000$</td>
</tr>
<tr>
<td>Ramon &amp; Driessens (2004)</td>
<td>Relational RL</td>
<td>$20 \times 10$</td>
<td>$\approx 50$</td>
</tr>
<tr>
<td>Böhm et al. (2005)</td>
<td>Genetic algorithm</td>
<td>$20 \times 10$</td>
<td>$480,000,000$ (Two Piece)</td>
</tr>
<tr>
<td>Farias &amp; Van Roy (2006)</td>
<td>Linear programming</td>
<td>$20 \times 10$</td>
<td>$4,274$</td>
</tr>
<tr>
<td>Romdhane &amp; Lamontagne (2008)</td>
<td>Case-based reasoning and RL</td>
<td>$20 \times 10$</td>
<td>$\approx 50$</td>
</tr>
<tr>
<td>Boumaza (2009)</td>
<td>CMA-ES</td>
<td>$20 \times 10$</td>
<td>$35,000,000$</td>
</tr>
<tr>
<td>Thiery &amp; Scherrer (2009a;b)</td>
<td>Cross entropy</td>
<td>$20 \times 10$</td>
<td>$35,000,000$</td>
</tr>
<tr>
<td>Gabillon et al. (2013)</td>
<td>Classification-based policy iteration</td>
<td>$20 \times 10$</td>
<td>$51,000,000$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Practical Issues and Fixes
Problem 1: What happens to the variance?

\[ \Sigma^t = \frac{1}{e} \sum_{i=1}^{e} (\theta_i - \mu^t)^2 \]

Collapses too quickly!

Simple fix: Add a bit of noise to the variance

\[ \Sigma^t = \frac{1}{e} \sum_{i=1}^{e} (\theta_i - \mu^t)^2 + \Sigma_{\text{noise}} \]
Problem 2: What if we have a bad batch of samples?

\[ \mu^t = \frac{1}{e} \sum_{i=1}^{e} \theta_i \]

The elites can be bad, and the mean can slingshot into a bad value

Simple fix: Slowly update mean

\[ \mu^t = \mu^{t-1} + \eta \frac{1}{e} \sum_{i=1}^{e} \theta_i \]
Problem 3: What if we never converge and do random walks?

Single-steps cancel out
Use small $\Sigma$

Progress correlated
Use large $\Sigma$

A very fancy version of Cross Entropy: CMA-ES
Tetris is cute…
But what about real robots?
Cross Entropy for Snake Robot Gaits

Using Response Surfaces and Expected Improvement to Optimize Snake Robot Gait Parameters

Matthew Tesch, Jeff Schneider, and Howie Choset

Uses a Gaussian Process to fit a distribution

Prove it can find the optimal gait with *minimal samples*
Cross Entropy Search for Motion Planning

Distribution over control trajectories
2560, 2.5 second trajectories sampled with cost-weighted average @ 60 Hz

Cross Entropy for Control

Georgia Tech Auto Rally (Byron Boots lab)
tl;dr

Can we just focus on finding a good policy?

\[ \pi_\theta : S_t \to a_t \]

Learn a mapping from states to actions
Roll-out policies in the real-world to estimate value

The Goal of Policy Optimization

\[ \pi_\theta^*(s) = \arg \min_{\theta} \theta^T J(s, a) \]

\[ \min_{\theta} J(\theta) = \sum_{t=0}^{T-1} E_{\pi_\theta} c(s_t, a_t) \]

The Cross Entropy Algorithm

Evaluate each \( \theta \)
Execute policy multiple times
Find top 'k' elites (Fig 15.2)
Fit a new distribution \( D_\theta \)