Some work around deep developmental learning

Olivier Sigaud

ISIR, Sorbonne Université
http://people.isir.upmc.fr/sigaud

November 29, 2018
Introduction

Developmental robotics challenges


Some work around deep developmental learning

Outline

Introduction
Some work around deep developmental learning

- Background
- General RL background

Reinforcement learning

- In SL, the learning signal is the correct answer
- In RL, the learning signal is a scalar
- How good is -10.45?
- Necessity of exploration
The exploration/exploitation trade-off

- Exploring can be (very) harmful
- Shall I exploit what I know or look for a better policy?
- Am I optimal? Shall I keep exploring or stop?
- Decrease the rate of exploration along time
Some work around deep developmental learning

---

### Background

#### General RL background

**Markov Decision Processes**

- $S$: states space
- $A$: action space
- $T : S \times A \rightarrow \Pi(S)$: transition function
- $r : S \times A \rightarrow \mathbb{R}$: reward function

---

Policy and value functions

- Goal: find a policy $\pi : S \rightarrow A$ maximizing the aggregation of rewards on the long run
- The value function $V^\pi : S \rightarrow \mathbb{R}$ records the aggregation of reward on the long run for each state (following policy $\pi$). It is a vector with one entry per state
- The action value function $Q^\pi : S \times A \rightarrow \mathbb{R}$ records the aggregation of reward on the long run for doing each action in each state (and then following policy $\pi$). It is a matrix with one entry per state and per action
Some work around deep developmental learning

---

### General RL background

#### RL Basics

- In dynamic programming, the agent knows the MDP
- In RL it doesn’t, it has to explore

- Two approaches:
  - Learn a model of \( T \): model-based (or indirect) reinforcement learning
  - Perform local updates at each step: model-free RL

- Model-free basics:
  - TD error (RPE): \( \delta = r_{t+1} + \gamma V^\pi(s_{t+1}) - V^\pi(s_t) \)
  - TD(0): \( V^\pi(s_t) \leftarrow V^\pi(s_t) + \alpha [r_{t+1} + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)] \)
  - \( V \) (or \( Q \)) converges when \( \delta \) converges to 0
  - TD(0) evaluates \( V^\pi(s) \) for a given policy \( \pi \), but how shall the agent act?

- Two solutions:
  - Work with \( Q^\pi(s, a) \) rather than \( V^\pi(s) \) (SARSA and Q-Learning)
  - Actor-critic methods (simultaneously learn \( V^\pi \) and update \( \pi \))
Q-Learning

- For each observed \((s_t, a_t, r_{t+1}, s_{t+1})\):
  \[
  \delta = r_{t+1} + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t)
  \]
- Update rule:
  \[
  Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \delta
  \]
- Policy: necessity of exploration (e.g. \(\epsilon\)-greedy)
- Convergence proved given infinite exploration


Some work around deep developmental learning

### Background

#### General RL background

From Q-Learning to Actor-Critic

<table>
<thead>
<tr>
<th>state / action</th>
<th>$a_0$</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_0$</td>
<td>0.66</td>
<td>0.88*</td>
<td>0.81</td>
<td>0.73</td>
</tr>
<tr>
<td>$e_1$</td>
<td>0.73</td>
<td>0.63</td>
<td>0.9*</td>
<td>0.43</td>
</tr>
<tr>
<td>$e_2$</td>
<td>0.73</td>
<td>0.9</td>
<td>0.95*</td>
<td>0.73</td>
</tr>
<tr>
<td>$e_3$</td>
<td>0.81</td>
<td>0.9</td>
<td>1.0*</td>
<td>0.81</td>
</tr>
<tr>
<td>$e_4$</td>
<td>0.81</td>
<td>1.0*</td>
<td>0.81</td>
<td>0.9</td>
</tr>
<tr>
<td>$e_5$</td>
<td>0.9</td>
<td>1.0*</td>
<td>0.0</td>
<td>0.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>state</th>
<th>chosen action</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_0$</td>
<td>$a_1$</td>
</tr>
<tr>
<td>$e_1$</td>
<td>$a_2$</td>
</tr>
<tr>
<td>$e_2$</td>
<td>$a_2$</td>
</tr>
<tr>
<td>$e_3$</td>
<td>$a_2$</td>
</tr>
<tr>
<td>$e_4$</td>
<td>$a_1$</td>
</tr>
<tr>
<td>$e_5$</td>
<td>$a_1$</td>
</tr>
</tbody>
</table>

- In $Q$-learning, given a $Q$-Table, get the max at each step
- Expensive if numerous actions (optimization in continuous action case)
- Storing the max is equivalent to storing the policy
- Update the policy as a function of value updates (only look for the max when decreasing max action)
- Note: looks for local optima, not global ones anymore
Some work around deep developmental learning

DQN

Parametrized representations

- To represent a continuous function, use features and a vector of parameters
- Learning tunes the weights
- Linear architecture: linear combination of features

- A deep neural network is not a linear architecture: deep layer parameters tune the features
- Parametrized representations:
  - In critic-based methods, like DQN: of the critic $Q(s_t, a_t | \theta)$
  - In policy gradient methods: of the policy $\pi(a_t | s_t, \mu)$
  - In actor-critic methods: both
DQN: the breakthrough

▶ DQN: Atari domain, Nature paper, small discrete actions set
▶ Learned very different representations with the same tuning

Some work around deep developmental learning

DQN

The Q-network in DQN

- Parametrized representation of the critic $Q(s_t, a_t | \theta)$
- The Q-network is the equivalent of the Q-Table
- Select action by finding the max (as in Q-Learning)
- Limitation: requires one output neuron per action
Learning the Q-function

- Supervised learning: minimize a loss-function, often the squared error w.r.t. the output:

\[ L(s, a) = (y^*(s, a) - Q(s, a|\theta))^2 \]  \hspace{1cm} (1)

with backprop on weights \( \theta \)

- For each sample \( i \), the Q-network should minimize the RPE:

\[ \delta_i = r_i + \gamma \max_a Q(s_{i+1}, a|\theta) - Q(s_i, a_i|\theta) \]

- Thus, given a minibatch of \( N \) samples \( \{s_i, a_i, r_i, s_{i+1}\} \), compute

\[ y_i = r_i + \gamma \max_a Q(s_{i+1}, a|\theta') \]

- And update \( \theta \) by minimizing the loss function

\[ L = 1/N \sum_i (y_i - Q(s_i, a_i|\theta))^2 \]  \hspace{1cm} (2)
Trick 1: Stable Target Q-function

- The target \( y_i = r_i + \gamma \max_a Q(s_{i+1}, a|\theta) \) is itself a function of \( Q \)
- Thus this is not truly supervised learning, and this is unstable
- Key idea: “periods of supervised learning”
- Compute the loss function from a separate target network \( Q'(\ldots|\theta') \)
- So rather compute \( y_i = r_i + \gamma \max_a Q'(s_{i+1}, a|\theta') \)
- \( \theta' \) is updated to \( \theta \) only each \( K \) iterations
Trick 2: Replay buffer shuffling

- In most learning algorithms, samples are assumed independently and identically distributed (iid)
- Obviously, this is not the case of behavioral samples \((s_i, a_i, r_i, s_{i+1})\)
- Idea: put the samples into a buffer, and extract them randomly
- Use training minibatches (make profit of GPU when the input is images)
- The replay buffer management policy is an issue


Deep Deterministic Policy Gradient

- Continuous control with deep reinforcement learning
- Works well on “more than 20” (27-32) domains coded with MuJoCo (Todorov) / TORCS
- End-to-end policies (from pixels to control)

Some work around deep developmental learning

DDPG

DDPG: ancestors

- Most of the actor-critic theory for continuous problem is for stochastic policies (policy gradient theorem, compatible features, etc.)
- DPG: an efficient gradient computation for deterministic policies, with proof of convergence
- Batch norm: inconclusive studies about importance

Some work around deep developmental learning

DDPG

General architecture

- Actor parametrized by $\mu$, critic by $\theta$
- All updates based on SGD (as in most deep RL algorithms)
Training the critic

- Same idea as in DQN, but for actor-critic rather than Q-Learning
- Minimize the RPE: \( \delta_t = r_t + \gamma Q(s_{t+1}, \pi(s_t)|\theta) - Q(s_t, a_t|\theta) \)
- Given a minibatch of \( N \) samples \( \{s_i, a_i, r_i, s_{i+1}\} \) and a target network \( Q' \), compute \( y_i = r_i + \gamma Q'(s_{i+1}, \pi(s_{i+1})|\theta') \)
- And update \( \theta \) by minimizing the loss function

\[
L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta))^2
\]
Training the actor

Deterministic policy gradient theorem: the true policy gradient is

$$\nabla_{\mu} \pi(s, a) = \mathbb{E}_{\rho(s)}[\nabla_a Q(s, a|\theta) \nabla_{\mu} \pi(s|\mu)]$$  \hspace{1cm} (4)$$

- $\nabla_a Q(s, a|\theta)$ is used as error signal to update the actor weights.
- Comes from NFQCA
- $\nabla_a Q(s, a|\theta)$ is a gradient over actions
- $y = f(w.x + b)$ (symmetric roles of weights and inputs)
- Gradient over actions $\sim$ gradient over weights

Some work around deep developmental learning

DDPG

Exploration in DDPG

- Action perturbation (versus param. perturbation)
- Adding to the action an Ornstein-Uhlenbenk (correlated) noise process
- Several papers found that using Gaussian noise does not make a difference


Some work around deep developmental learning

Goal Exploration Processes

Where are we now?
Continuous Mountain Car

- Loss of energy depending on action, reward +100 for reaching the goal
- Deceptive gradient issue: before finding the goal, the agent is driven towards doing nothing
- **Spoiler alert:** DDPG fails because of poor exploration
Goal Exploration Processes: algorithm

- Define a relevant outcome space/goal space
- To each policy parameter $\theta$ corresponds an outcome $O$

Goal Exploration Processes: algorithm

- Bootstrap phase: draw a few random $\theta$
- Store the resulting $(\theta, O)$ pairs into an archive

Goal Exploration Processes: algorithm

- Sample a goal at random in the outcome space
- May use the convex hull from bootstrap

Goal Exploration Processes: algorithm

- Find the nearest neighbor $O$ in archive and select the associated $\theta$
- Perturb the corresponding $\theta$ into $\theta'$ and get a new outcome $O'$

Goal Exploration Processes: algorithm

- One may sample unfeasible goals, favors outcome diversity
- As the archive fills up, performance improves

Why does GEP work better than random search?

- Very often, few parameter vectors map to interesting outcomes
- The GEP algorithm favors sampling these interesting outcomes
- If the mapping is the identity, similar to random search
GEP-PG

Combines GEP for exploration and DDPG for gradient-based search
Transfer is through the replay buffer
Strong evaluation methodology (openAI baselines, 20 seeds...)

Some work around deep developmental learning

GEP-PG

Experimental set-up

CMC: outcome/goal space

- Defined by hand, informs the search process about relevant dimensions
Half-Cheetah

- 17D observation vector, 6D action vector
- Outcome/goal space: average velocity and min height of head
DDPG fails on CMC

- Key factor: when does it find the reward first?
- DDPG is sensitive to the deceptive gradient issue
- But still better than pure random noise
GEP-PG performs better on CMC

- Efficient exploration solves the deceptive gradient problem
- But isn’t the GEP enough?
GEP-PG performs very well on half-cheetah

- SOTA results when submitted to ICML (SAC & TD3 do better now)


Some work around deep developmental learning

GEP-PG

Results

Sanity check

- GEP exploration is better than random exploration
- Random exploration is better than DDPG exploration!
Analyzing GEP-PG performance

- GEP-PG performance correlates with GEP performance and diversity
- But does not correlate with the size of the GEP buffer
- Thus, the better and the more diverse the replay buffer, the better DDPG
Take home messages

- State-of-the-art deep RL algorithms like DDPG can fail on simple 2D benchmarks like Continuous Mountain Car
- Efficient exploration is needed to improve over deep RL
- GEPs are good at exploring
- They are also more stable: the archive/population does not forget
- Better combinations than GEP-PG can be found (using SAC or TD3, advanced GEPs...)
Where are we now?
From GEPs to evolutionary methods

Evo. methods and GEPs are similar (episode-based, population)

Genetic Algorithms

- Inspired from theory of natural selection
- Many different implementations (here, tournament selection)

The Cross Entropy Method

1. Start with the normal distribution $N(\mu, \sigma^2)$
2. Generate $N$ vectors with this distribution
3. Evaluate each vector and select a proportion $\rho$ of the best ones. These vectors are represented in grey
4. Compute the mean and standard deviation of the best vectors
5. Add a noise term to the standard deviation, to avoid premature convergence to a local optimum
6. This mean and standard deviation define the normal distribution of next iteration

- A particular case of evolution strategy

Importance Mixing

- A mechanism to improve sample efficiency

Some work around deep developmental learning

Combining evolutionary methods and deep RL

Two Combinations

Combining evolutionary methods and deep RL is an emerging domain


Some work around deep developmental learning
Combining evolutionary methods and deep RL
Results

Results (1)

- CEM-TD3 outperforms CEM and TD3
Results (2)

- CEM-TD3 outperforms ERL
Results (3)

- On swimmer, the best is CEM
Results

- Changing from ReLu to \textit{tanh} significantly improves performance
- Strong incentive for neural architecture search
Where are we now?

- **RL**
  - DQN
  - DDPG

- **Explo.**
  - GEPs
  - GEP-PG

- **ERL/CEM-RL**

- **Curriculum Learning**
  - **CURIOS**

- **Results**
Goal Exploration Processes: curriculum learning

- Sample preferentially regions where learning progress is greater
- Known to improve performance on multitask learning

Curriculum based on competence progress

Experiments with Reacher using various accuracy requirements

Some work around deep developmental learning
- Towards curriculum learning
- Curriculum based on accuracy

Curriculum performance

- Random sampling of required accuracy is better than always using the strongest requirement
- Sampling based on competence progress is better than random sampling

Experimental setup

- Move various blocks to various position, stack them etc.
- Combine curriculum learning with Hindsight Experience Replay

A sophisticated architecture

Dedicated to dealing with tasks and goals

Results

- Generalization over task and goal is better than learning separated tasks
Conclusion

- State-of-the-art deep RL tools still fail on easy benchmarks
- Work needed on exploration, gradient descent, fundamental understanding

- Towards open-ended multi-task learning, zero-shot transfer learning
- Hot topics: curriculum learning, hierarchical RL, model-based RL...

Any question?
Some work around deep developmental learning

References


Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.

Reinforcement learning in feedback control.
Machine learning, 84(1-2), 137–169.

Evolution-guided policy gradient in reinforcement learning.
In Neural Information Processing Systems.

Continuous control with deep reinforcement learning.

Self-Improving Reactive Agents based on Reinforcement Learning, Planning and Teaching.

The cross-entropy method for fast policy search.
In Proceedings of the 20th International Conference on Machine Learning (pp. 512–519).

Human-level control through deep reinforcement learning.

Unsupervised learning of goal spaces for intrinsically motivated goal exploration.
In International Conference on Learning Representations (ICLR).
Some work around deep developmental learning

References

First-order and second-order variants of the gradient descent: a unified framework.

Parameter space noise for exploration.

Cem-rl: Combining evolutionary and gradient-based methods for policy search.

Sigaud, O. & Droniou, A. (2016).
Towards deep developmental learning.

Intrinsically motivated goal exploration processes as a central framework for open-ended learning of rich representations.
*In preparation.*

Policy search in continuous action domains: an overview.

Deterministic policy gradient algorithms.
*In Proceedings of the 30th International Conference in Machine Learning.*

Efficient natural evolution strategies.
*In Proceedings of the 11th Annual conference on Genetic and evolutionary computation (pp. 539–546).: ACM.*

Some work around deep developmental learning

---

References

*Reinforcement Learning: An Introduction.*
MIT Press.


*Learning with Delayed Rewards.*


Q-learning.

*Zhang, S. & Sutton, R. S. (2017).*

A deeper look at experience replay.