# Trends and developments in Reinforcement Learning

## London TensorFlow Meetup

Pierre H. Richemond

Imperial College London

# Table of contents

1. Introduction: the RL Setting
2. Theoretical Developments
3. Progress in continuous control
4. Progress in board games
5. Some practical tips

# Introduction: the RL Setting

# RL in one minute !

- Reinforcement learning = adaptive, extreme trial-and-error
- Deep RL = RL combined with deep neural networks
- Given a certain environment (game abstraction), an agent (player) selects between actions $a$ available to her, and in doing so, collects a reward $r$ and moves from state $s$ to state $s'$.
- The agent's objective is to pick actions and build a trajectory so as to maximize the sum of their rewards.
- Rewards and states are not known in advance, so that RL involves a trade-off between exploration of new and risky states and exploitation of known but potentially sub-optimal rewards.

# Just one more minute :)

Two main families of algorithmic methods for solving RL:

1. Q-learning consists in computing an approximation to an *action-value function* $Q(s, a)$ which for each state-action pair tells us how happy we will be, over the long run, if we take that action in that state.
2. Policy gradients consist in approximately following the gradient to the sum of rewards ('do more of what works best') so as to maximize it. They seek to compute $\pi(a|s) = \mathbb{P}(a|s)$.

Deep Q-Networks (DQN) and Asynchronous Advantage Actor Critic (A3C) are the most famous deep RL algorithms.
In general, RL methods are numerically unstable and require tricks to stabilize them and accelerate their convergence.

# Theoretical Developments

# Equivalence of soft Q-learning and policy gradients

- The single most important theoretical development of 2017.
- We've been doing reinforcement learning wrong all these years !
- Idea = turn the hard max in Q-learning into a soft max.
- Mathematically, relax $Q^*(s,a) = \max_{a \in \mathbb{A}} \quad Q(s,a)$ into

$$Q^*(s,a) = \beta \log \int exp(\frac{Q(s,a)}{\beta}) da$$

- This relaxation was already known since 2010 at least.

# Equivalence of soft Q-learning and policy gradients (2)

- Policy gradients often use entropy regularization to prevent early convergence to degenerate policies.
- Schulman shows that in this context, soft Q-learning and entropic policy gradients are the same.
- Easily proven with an entropic representation formula (Richemond & Maginnis, to appear.)
- This also proves that the A3C algorithm as originally published is 'wrong'/incomplete as it may fail to converge (Neu 2017).
- Strong theoretical connections both with thermodynamics and convex/online optimization.

# Wasserstein RL solves the heat equation

- New development, on ArXiv soon
- Optimal transport / *Wasserstein* distances have become popular in the context of GANs
- When doing trust region policy optimization in a small Wasserstein-2 ball rather than Kullback-Leibler, one proves that the policy function itself, in the small step limit, satisfies the heat equation

$$\partial_t \pi = -\nabla \cdot (\pi \nabla r) + \beta \Delta \pi$$

  with $\beta$ the strength of regularization.
- This might also be a justification for the emergence of *distributional* RL

# Progress in continuous control

# Video examples

Proximal Policy Optimization - Robust knocked over stand up
`https://www.youtube.com/watch?v=bqdjsmSoSgI`

Emergence of Locomotion Behaviours in Rich Environments
`https://www.youtube.com/watch?v=hx_bgoTF7bs&t=49s`

# TRPO-PPO

- *Trust Region Policy Optimization* is a very robust policy gradient baseline that iteratively refines policies by moving them within a small stepsize.
- This approach introduces strong sensitivity w.r.t. the stepsize as an important tunable hyperparameter, and makes it generally sample inefficient
- *Proximal Policy Optimization* is an evolution of TRPO that attempts to do away with those constraints
- The idea is to use a new objective function that includes clipping the ratio of the probability under the new and old policies
- Simple change in the code.

# ACKTR

- Combines actor-critic, trust region optimization, and Kronecker factorization (’K-FAC’)
- As this is a second-order (*natural gradient*) method, it’s more sample efficient
- Flipside is increased computational complexity
- Scales well with batch size
- Together PPO and ACKTR represent the state-of-the-art baseline in continuous control… until the official release of *soft actor-critic* (2018, OpenAI)

# Progress in board games

# AlphaGo Zero

- Arguably the landmark practical achievement of the year in reinforcement learning !
- AlphaGo the first learned the game by analyzing 30 million Go moves from human players (*GoKiFu* dataset) before refining that supervised policy with self-play
- This illustrates the difficulty of the mathematical optimization problem posed by RL, and the necessity to initialize close to a good solution.
- AlphaGo Zero plays significantly better, and does away with handcrafting or human knowledge completely (other than knowledge of game rules).

# AlphaGo Zero: Architecture (1)

- Problem with Go = large branching factor in the tree of possible moves.
- Neural networks enable dynamic pruning of that tree - a *policy* network limits the breadth of the subtree to explore, while a *value* network limits the depth of that subtree. *If those networks were perfect, little exploration would be needed - in order to get there, we can refine them iteratively.*
- Merging the policy and value network enforces feature reuse and learning efficiency
- In order to train deeper networks and enable more and more abstract feature extraction, *residual networks* are used

# AlphaGo Zero: Architecture (2)

- Monte Carlo Tree Search (MCTS) is another part of the algorithm that, just like a neural network, dynamically expands and compresses the tree of possibles. It averages results across many paths of the sub-tree, and can be viewed as a form of imagination, or slow thinking.
- The key component of AlphaGo Zero is to view MCTS as a *policy improvement* mechanism: you can't improve anymore when your fast thinking (NN) and your slow thinking (MCTS) coincide.
- The loss function is ”simple” for a program that complex:

$$L(\theta) = (z - v_\theta)^2 + \pi^T \log p_\theta + c||\theta||^2$$

# AlphaGo Zero: Insights

- *AlphaZero* is the generic, game-agnostic version of AlphaGoZero that also taught itself to play Chess and Shogi (Japanese chess where you can capture your opponent's pieces) at a superhuman level using the same algorithm.
- In stark contrast with DeepBlue or Stockfish, chess AlphaZero evaluates 'only' around 50,000 positions per move due to MCTS, rather than millions.
- AlphaGo has now become so strong it is used as an online teaching tool for human players - *AlphaGo Teach*

  `https://alphagoteach.deepmind.com/`
- According to David Silver, it is speculated that MCTS acts as a large averaging mechanism, rather than the 'maximum' operation in standard tree pruning, smoothing rather than propagating neural network approximation error, and hence enabling stable training of deep networks.

# Some practical tips

# Suggestions for experiment design

Factors influencing training, by order of importance:

- Fix your random seed for reproducibility !
- *Softmax temperature* is the single most critical factor according to Nachum & Haarnoja, followed by...
- *Reward scaling* (and more generally reward engineering) help a lot
- *Number of steps* in the calculation of returns
- *Discount factor* also an issue

Clip gradients to avoid NaNs, visualize them with TensorBoard, and finally be patient waiting for convergence...

# Key TensorFlow code repos

- As close to official as possible : `https://github.com/tensorflow/models/tree/master/research/pcl_rl` by Ofir Nachum the author of PCL, soon to be integrated in TensorFlow
- RLlib: A Composable and Scalable Reinforcement Learning Library `https://github.com/ray-project/ray/tree/master/python/ray/rllib`
- Imperial's Brendan Maginnis Atari-RL `https://github.com/brendanator/atari-rl`
- OpenAI's *baselines* `https://github.com/openai/baselines`

## Deep Reinforcement Learning That Matters

- Encouraging reproducibility in reinforcement learning
- 'RL algorithms have many moving parts that are hard to debug, and they require substantial effort in tuning in order to get good results.'
- False Discoveries everywhere ? Dependency on the random seed
- Multiple runs are necessary to give an idea of the variability. Reporting the best returns is not enough - at the very least, report top and bottom quantiles
- A necessary debate, as seen at NIPS 2017.

Don't be an alchemist

# Conclusion & perspectives

- Another landmark year both for theoretical and practical RL
- In imperfect information settings - Poker : *Libratus* and *DeepStack* become superhuman at Texas Hold'em
- In robotics, notable progress also in *hierarchical reinforcement learning*, *meta-learning* and *imitation learning*.
- Planning, robustness and sample efficiency remain the greatest challenges to current methods
- The next big frontier : Starcraft II, a real-time strategy game, partially observed, with delayed rewards and a huge action space. PySC2 available to train agents at

`https://github.com/deepmind/pysc2`

# Questions from the audience ?

# Thanks for your time !

Twitter @KloudStrife