Primers

In this course we assume that you already know some things from machine learning and have some coding experience. In this section, we have compiled some prerequisites you might want to brush up on before moving on.

If you see something you don't know, that's no big deal: just grab a primer (links below), cram through it and pretend you had already known this stuff for ages.

We do however expect you to know algebra, basic calculus and probability. If you lack any of these, taking this course might be excruciatingly painful. In any case, welcome!

Machine Learning Primer

Basic machine learning stack for pythonistas: Numpy, Pandas, Matplotlib, Scikit-learn (required right away). Please take a look if you can't confidently say you know those libraries.

https://colab.research.google.com/github/yandexdataschool/Practical_RL/blob/coursera/week1_intro/primer/recap_ml.ipynb

Deep Learning Primer

Here you have an option to brush up on your knowledge of practical deep learning, which you will need from Week 4 onwards. We provide you with two alternatives.

1. The current default option is to use Tensorflow, which is also used in other courses of this specialization. Our recap covers the basics of Tensorflow; you might also want to take a look at Keras, which provides a more high-level interface. Link: https://colab.research.google.com/github/yandexdataschool/Practical_RL/blob/coursera/week1_intro/primer/recap_tensorflow.ipynb
2. An alternative option is to use PyTorch, which is arguably more suited to research-like problems (like the ones we cover in this course). We rolled out support for it on Coursera only in April 2020, but it has been the primary framework for the on-campus version of this course at HSE and YSDA for a couple of years. Link: https://colab.research.google.com/github/yandexdataschool/Practical_RL/blob/coursera/week1_intro/primer/recap_pytorch.ipynb

If you do not have any preference, we suggest you go with PyTorch.

WEEK 2

Discrete Stochastic Dynamic Programming

Generalised Policy Iteration is a wonderful concept, that is extremely well covered in an introductory textbook by Richard Sutton. Throughout the course you are highly encouraged to consult with this wonderful masterpiece.

However, if you need more of mathematical rigor we recommend you to read about Banach fixed point theorem and its applications. You may benefit a lot from either a concise book by Csaba Szepesvári or an extensive introduction to Markov Decision Processes by Martin Puterman.

Week 3

Extras

More on TD

- When discounted returns misbehave - https://blog.openai.com/faulty-reward-functions/
- N-step temporal difference from Sutton's book - http://incompleteideas.net/book/bookdraft2018jan1.pdf , **chapter 8**
- N-step methods and Eligibility traces from Sutton's book - same book, **chapter 12**
- Blog post on eligibility traces - [url](http://pierrelucbacon.com/traces/)

Alternative materials in english

- Lecture by David Silver (english) - video part I, video part II

- Alternative lecture by Pieter Abbeel (english) - video

- Alternative lecture by John Schulmann (english) - video

- Blog post on q-learning Vs SARSA - url

Alternative materials in russian:

- Recap of value iteration & Q_learning - video

- Q-learning seminar - video

- More on value-based methods - video 2

- Sarsa & stuff - seminar2

Week4

TD vs MC

Difference between the TD and MC is a long standing topic in Reinforcement Learning. For the theoretical implications see the introductory textbook by Richard Sutton (section 6.3).

For the empirical and, probably, counterintuitive findings see the recent ICRL paper by Amiranashvili et.al. Some of the conclusions, made by the reviewer of this paper are as follows:

- Pure MC methods can outperform TD methods when the rewards are noisy.
- TD methods can outperform pure MC methods when the return is mostly dominated by the reward in the terminal state.
- MC methods tend to degrade less when the reward signal is delayed.
- MC methods seems to be on-par with TD methods when the reward is sparse.
- MC methods can outperform TD methods with more complex and high dimensional perceptual inputs.

There are a lot more details in the paper, so you are highly encouraged to at least skim through it.

Extras

We have put together a sheet of various open-source implementations of RL algorithms. Check it out and feel free to leave comments on the forum about it.

Here's a few more things you might enjoy:

- "The nuts and bolts" of deep RL (by J. Schulman) - video or slides
- A large overview article of deep RL (as of 2017) - arXiv

- Lecture covering some DQN modifications (also by J. Schulman) - video
- Karpathy's post on several approx RL methods, including DQN - blog

Week 5

Stability of policy-based vs value-based methods

This text originally appeared on the forum and was slightly edited to be included in the course.

In general, policy-based methods are less stable (and more noisy) than value-based methods. For example, see https://flyyufelix.github.io/2017/10/12/dqn-vs-pg.html which demonstrates how strongly performance can fluctuate in PG, in contrast to relatively stable DQN. There are different points of view on this issue, and some of the most important are considered below.

On- / off- policy

In Q-learning, the most popular off-policy method, we aim to estimate the Q^*Q_* — the action-value function of an optimal policy. This function does NOT depend on the current policy and thus is the same in the process of learning.

In A2C we need to estimate Q^\pi$Q_\pi$ — the action-value function of current policy (which is usually estimated with a one-sample estimate \hat Q^\pi(s_t, a_t) \approx r_t(s_t, a_t) + V^\pi(s_{t+1}) $Q^\pi(s_t,a_t) \approx r_t(s_t,a_t) + V_\pi(s_{t+1})$.

This function depends on the current policy and changes as we change the policy. This nonstationarity of the target function may be responsible for the observed wiggles in training curves.

Sample efficiency

The on-policy nature of policy-based methods imply the eagerness for data — we need much more data for such methods to reliably estimate the Q^\pi(s_t, a_t) $Q_\pi(s_t,a_t)$ compared to the estimation of Q^* Q_*.

This means that the finite-sample approximation of \mathbb{E} [\nabla\log\pi(a_t | s_t) (Q^\pi(s_t, a_t) - V^\pi(s_t))]E[∇log$\pi(a_t|s_t)(Q_\pi(s_t,a_t)-V_\pi(s_t))$] may point in the reverse direction of policy improvement, if the estimated advantage \hat Q^\pi(s_t, a_t) - \hat V^\pi(s_t)$Q^\pi(s_t,a_t)-V^\pi(s_t)$ is noisy (environment stochasticity) or unreliable (bad estimate of \hat V$V^$).

This is likely the case in the article linked at the top — the REINFORCE learning curve is much less wiggly compared to the A2C one, probably meaning the critic should be improved before updating the policy.

Convergence to local optima may also stem from the on-policy nature. Imagine that the policy (and value) happened to find a local maximum during the stochastic optimization. In this local maximum the policy generates suboptimal trajectories (e.g. agent is hiding from monsters under the stone) which are very uninformative of how to improve the policy. Thus A2C may not find its way out of this local maximum just by recalling previous interactions with monsters and occasional successful combats with them.

Q-learning, on the contrary, does “remember” previous interactions and any experience (including the hiding-under-the-stone interactions) can only increase the precision of Q^*Q_* approximation.

To overcome convergence to local optima it is usual to encourage high entropy policies (by adding policy entropy as a regularizer to the RL objective).

This helps in practice, but even with such a regularizer, one can show that mode seeking behaviour still takes place (see https://openreview.net/pdf?id=ryT4pvqll, especially Sec. 4).

Extras

Generalizing formula for advantage:

- You can also use N-step formula for A(s_t,a_t) = r_t + gamma * r_{t+1} + ... + gamma ^ N * V(s_{t + N})
- If you average them with coefficients like TD(lambda), you get GAE: https://arxiv.org/abs/1506.02438
- This alone can speed up early convergence of actor critic in honor section by a factor of 2~10 depending on parameters.

More: Policy gradient on steroids

- Lecture on NPG and TRPO by J. Schulman - video
- Alternative lecture on TRPO and open problems by... J. Schulman - video
- TRPO week in github course (russian and english materials, code labs)

More: Reinforcement learning in large/continuous action spaces

While you already know algorithms that will work with continuously many actions, it can't hurt to learn something more specialized.

- Deterministic policy gradient - article, blog post+code
- Q-learning with normalized advantage functions - article, some implementation code
- Stochastic value gradient - article
- Embedding large discrete action spaces for RL - article
- Lecture by A. Seleznev, 5vision (**russian**) - video

Week 6

Extras: exploration

Here's a few other things you might enjoy:

- David Silver's video lecture where he doesn't shun away from covering Gittins Indices and proving some stuff
- Variational Information Maximizing Exploration - explanation video, original article
- A survey of bayesian stuff in RL: article
- Gittins Indices - the more theoretically grounded approach to optimal exploration - article See also: David's lecture (the first dot in this list)

Extras: Planning

- Basic tree-based planning: lectures 2-6 from here

- David Silver's lecture on integrating learning and planning: video
- A blog post on AlphaGo Zero: blog