Deep Reinforcement Learning Algorithms

ANDREAS HOFMANN

This provides a summary and references for current standard deep reinforcement learning algorithms, including .

1 INTRODUCTION

ACM's consolidated

List algorithms, organized (see web sites)
model-free
model-based
learning value function
learning policy directly
Whether output action is discrete or continuous
Whether input is raw pixels, or a state estimate

1.1 To Do

Focus on continuous action models like mountain car.

Think about whether it makes sense to have hybrid discrete/continuous actions, maybe from two separate heads of the DNN.

2 DQN

Q value algorithm

2.1 Agent, Reward, Value Function, Q Function, and Policy

In Reinforcement Learning, an *agent* takes actions in an environment in order to accumulate *rewards*. The immediate reward at time increment t is given as R_t. When learning how an agent should behave, it is useful to consider *cumulative* reward, rather than just immediate reward. Cumulative reward is defined as the total reward accumulated by the agent over some time horizon:

$$C_t = R_t + R_{t+1} + R_{t+2} + \ldots = \sum_{k=0}^{T} R_{t+k} \tag{1}$$

Here, the agent takes actions at regular time increments, denoted by k. The time horizon, T, is specified in terms of the maximum value for k.

In practice, when learning agent behaviors, it is useful to favor immediate reward over future reward. This is expressed as *discounted* reward, using a discount rate, γ:

Author's address: Andreas Hofmann, hofma@csail.mit.edu.

XXXX-XXXX/2021/8-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \ldots = \sum_{k=0}^{T} \gamma^k R_{t+k} \tag{2}$$

With the discount rate, T is usually set at infinity

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k} \tag{3}$$

The discount rate implies a time horizon in that rewards diminish significantly in the distant future.

A policy, π, specifies an action, a, to be taken by the agent when the state is s. Two important functions can be defined with respect to a policy. First, a value function gives the total expected reward given a current state, s_t, if a particular policy, π, is followed.

$$v_\pi(s) = \mathbb{E}\left(G_t | S_t = s\right) \tag{4}$$

Second, an action-value or Q function gives the total expected reward given a current state, s_t, a current action, A_t, and a particular policy, π, to be followed (for future actions).

$$q_\pi(s,a) = \mathbb{E}\left(G_t | S_t = s, A_t = a\right) \tag{5}$$

The policy can be expressed in terms of the Q function as

$$\pi(s) = \operatorname*{argmax}_a q_\pi(s,a) \tag{6}$$

Note that this is circular; the Q function is, itself, a function of the policy. Thus, the challenge is how to compute (learn) the Q function. To accomplish this, we leverage the Bellman equation:

$$q_\pi(s,a) = r + \gamma q_\pi(s', \pi(s')) = r + \gamma \operatorname*{argmax}_{a'} q_\pi(s', a') \tag{7}$$

where r is the immediate reward for action a, s' is the next state resulting from taking action a at state s, and a' is the action to be taken at state s'.

2.2 Training: Loss Function, Agent Steps, Replay Memory, and Policy Update

The *temporal difference error*, δ, is defined as the difference between the two sides of 8:

$$\delta = q_\pi(s,a) = -\left(r + \gamma \operatorname*{argmax}_{a'} q_\pi(s', a')\right) \tag{8}$$

During training, RL algorithms strive to update $q_\pi(s,a)$ so that δ approaches 0. In (non-deep) RL, $q_\pi(s,a)$ is represented as a table, and is updated by:

$$q_\pi(s,a) = q_\pi(s,a) + \alpha\left(r + \gamma \operatorname*{argmax}_{a'} q_\pi(s', a') - q_\pi(s,a)\right) \tag{9}$$

where α is the learning rate.

In Deep Reinforcement Learning, $q_\pi(s,a)$ is represented using a Deep Neural Net rather than a table. Training (minimization of δ) is accomplished by defining a loss based on δ. In the PyTorch DQN tutorial (https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html), a *Huber Loss* function is used. This loss, $\mathcal{L}$, is defined as

$$\mathcal{L} = \begin{cases} \frac{1}{2}\delta^2 & \text{for } |\delta| \le 1, \\ |\delta| - \frac{1}{2} & \text{otherwise.} \end{cases} \tag{10}$$

Training occurs over some specified number of episodes (50, for example). Within each episode, the environment state is first re-initialized, and then the agent explores the environment by taking *steps*. For each step, the agent takes an action, resulting in a reward, and a transition from the current state to the next state. Thus, a step can be summarized by the tuple ⟨state action next-state reward⟩. An episode ends when the environment decides this (usually because the agent has entered a failure state), or at some specified limit on the number of steps per episode.

After each step, the result tuple is entered into *replay memory*. Replay memory is a buffer of these tuples with a limited size (10000 for example). As result tuples are added to the end of replay memory, old results are dropped from the beginning. Thus, replay memory stores the last n (10000 for example) results of experiment steps taken by the agent.

After each agent step, the Q model is updated (optimized). This is accomplished not by using the results from the most recent step, but rather, by sampling a *batch* of results from replay memory. A typical batch size is 128. Leveraging the Huber loss function defined above, the loss for the entire batch is defined as

$$\mathcal{L}_B = \frac{1}{|B|} \sum_{(s,a,s',r)\in B} \mathcal{L} \tag{11}$$

Model weights are then updated using standard back propagation, based on this loss for the batch.

Experience replay memory ensures that the experience samples in a batch are not correlated; that they are not overly biased towards results from the most recent policy (model weights). This stabilizes and improves DQN training.

As in traditional reinforcement learning, the action selected by an agent during a step is sometimes *greedy* (following the current learned policy), and is sometimes *exploratory* (random). The probability of selecting a random action starts high (at 0.9, for example), and then decays exponentially to a small value (0.05, for example).

2.3 Policy and Target Networks

During training, two parallel networks are used: *policy* network, and *target* network. The policy network weights are updated at each training step, using standard back propagation. The target network weights are kept fixed for most steps; they are updated at a specified interval (every 10 training steps for example) from the policy network weights. The policy network is used to compute the left side of 8. The target network is used to compute $q_\pi(s', a')$ for the right side of 8. Using a separate target network in this way improves training stability.

2.4 Template Styles

`\documentclass[STYLE]{acmart}`

Journals use one of three template styles. All but three ACM journals use the `acmsmall` template style:

- `acmsmall`: The default journal template style.
- `acmlarge`: Used by JOCCH and TAP.
- `acmtog`: Used by TOG.

3 TABLES

The “`acmart`” document class includes the “`booktabs`” package — https://ctan.org/pkg/booktabs — for preparing high-quality tables.

Table 1. Frequency of Special Characters

Non-English or Math	Frequency	Comments
Ø	1 in 1,000	For Swedish names
π	1 in 5	Common in math
\$	4 in 5	Used in business
Ψ_1^2	1 in 40,000	Unexplained usage

Table 2. Some Typical Commands

Command	A Number	Comments
`\author`	100	Author
`\table`	300	For tables
`\table*`	400	For wider tables

Table captions are placed *above* the table.

Because tables cannot be split across pages, the best placement for them is typically the top of the page nearest their initial cite. To ensure this proper "floating" placement of tables, use the environment **table** to enclose the table's contents and the table caption. The contents of the table itself must go in the **tabular** environment, to be aligned properly in rows and columns, with the desired horizontal and vertical rules. Again, detailed instructions on **tabular** material are found in the *LaTeX User's Guide*.

Immediately following this sentence is the point at which Table 1 is included in the input file; compare the placement of the table here with the table in the printed output of this document.

To set a wider table, which takes up the whole width of the page's live area, use the environment **table*** to enclose the table's contents and the table caption. As with a single-column table, this wide table will "float" to a location deemed more desirable. Immediately following this sentence is the point at which Table 2 is included in the input file; again, it is instructive to compare the placement of the table here with the table in the printed output of this document.

4 MATH EQUATIONS

You may want to display math equations in three distinct styles: inline, numbered or non-numbered display. Each of the three are discussed in the next sections.

4.1 Inline (In-text) Equations

A formula that appears in the running text is called an inline or in-text formula. It is produced by the **math** environment, which can be invoked with the usual `\begin . . . \end` construction or with the short form `$ . . . $`. You can use any of the symbols and structures, from α to ω, available in LaTeX [?]; this section will simply show a few examples of in-text equations in context. Notice how this equation: $\lim_{n\rightarrow \infty} x = 0$, set here in in-line math style, looks slightly different when set in display style. (See next section).

4.2 Display Equations

A numbered display equation—one set off by vertical space from the text and centered horizontally—is produced by the **equation** environment. An unnumbered display equation is produced by the **displaymath** environment.

Again, in either environment, you can use any of the symbols and structures available in LaTeX; this section will just give a couple of examples of display equations in context. First, consider the equation, shown as an inline equation above:

$$\lim_{n\to\infty} x = 0 \tag{12}$$

Notice how it is formatted somewhat differently in the **displaymath** environment. Now, we'll enter an unnumbered equation:

$$\sum_{i=0}^{\infty} x + 1$$

and follow it with another numbered equation:

$$\sum_{i=0}^{\infty} x_i = \int_{0}^{\pi+2} f \tag{13}$$

just to demonstrate LaTeX's able handling of numbering.

5 FIGURES

The "`figure`" environment should be used for figures. One or more images can be placed within a figure. If your figure contains third-party material, you must clearly identify it as such, as shown in the example below.

Your figures should contain a caption which describes the figure to the reader. Figure captions go below the figure. Your figures should **also** include a description suitable for screen readers, to assist the visually-challenged to better understand your work.

Figure captions are placed *below* the figure.

6 CITATIONS AND BIBLIOGRAPHIES

The use of BibTeX for the preparation and formatting of one's references is strongly recommended. Authors' names should be complete — use full first names ("Donald E. Knuth") not initials ("D. E. Knuth") — and the salient identifying features of a reference should be included: title, year, volume, number, pages, article DOI, etc.

The bibliography is included in your source document with these two commands, placed just before the `\end{document}` command:

```
\bibliographystyle{ACM-Reference-Format}
\bibliography{bibfile}
```

where "`bibfile`" is the name, without the "`.bib`" suffix, of the BibTeX file.

Citations and references are numbered by default. A small number of ACM publications have citations and references formatted in the "author year" style; for these exceptions, please include this command in the **preamble** (before "`\begin{document}`") of your LaTeX source:

```
\citestyle{acmauthoryear}
```

Some examples. A paginated journal article [**?**], an enumerated journal article [**?**], a reference to an entire issue [**?**], a monograph (whole book) [**?**], a monograph/whole book in a series (see 2a in spec. document) [**?**], a divisible-book such as an anthology or compilation [**?**] followed by the same example, however we only output the series if the volume number is given [**?**] (so Editor00a's series should NOT be present since it has no vol. no.), a chapter in a divisible book [**?**], a chapter in a divisible book in a series [**?**], a multi-volume work as book [**?**], an article in a proceedings (of a conference, symposium, workshop for example) (paginated proceedings article) [**?**], a proceedings article with all possible elements [**?**], an example of an enumerated proceedings article [**?**], an informally published work [**?**], a doctoral dissertation [**?**], a master's thesis: [**?**], an online document / world wide web resource [**? ? ?**], a video game (Case 1) [**?**] and (Case 2) [**?**] and [**?**] and (Case 3) a patent [**?**], work

Fig. 1. 1907 Franklin Model D roadster. Photograph by Harris & Ewing, Inc. [Public domain], via Wikimedia Commons. (https://goo.gl/VLCRBB).

accepted for publication [?], 'YYYYb'-test for prolific author [?] and [?]. Other cites might contain 'duplicate' DOI and URLs (some SIAM articles) [?]. Boris / Barbara Beeton: multi-volume works as books [?] and [?]. A couple of citations with DOIs: [? ?]. Online citations: [? ? ?].

7 SIGCHI EXTENDED ABSTRACTS

The "`sigchi-a`" template style (available only in LaTeX and not in Word) produces a landscape-orientation formatted article, with a wide left margin. Three environments are available for use with the "`sigchi-a`" template style, and produce formatted output in the margin:

- `sidebar`: Place formatted text in the margin.
- `marginfigure`: Place a figure in the margin.
- `margintable`: Place a table in the margin.

ACKNOWLEDGMENTS

To Robert, for the bagels and explaining CMYK and color spaces.

A USEFUL WEB REFERENCES

A.1 Introduction to Deep Learning

https://markus-x-buchholz.medium.com/deep-reinforcement-learning-introduction-deep-q-network-dqn-algorithm-fb74bf4d6862
https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html
https://www.guru99.com/reinforcement-learning-tutorial.html
https://deepmind.com/blog/article/deep-reinforcement-learning
https://smartlabai.medium.com/reinforcement-learning-algorithms-an-intuitive-overview-904e2dff5bbc
https://en.wikipedia.org/wiki/Deep_reinforcement_learning#Off-policy_reinforcement_learning

A.2 DQN

https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html
https://unnatsingh.medium.com/deep-q-network-with-pytorch-d1ca6f40bfda
https://www.toptal.com/deep-learning/pytorch-reinforcement-learning-tutorial
https://towardsdatascience.com/deep-q-learning-tutorial-mindqn-2a4c855abffc
https://towardsdatascience.com/dqn-part-1-vanilla-deep-q-networks-6eb4a00febfb
https://en.wikipedia.org/wiki/Q-learning
https://openai.com/blog/openai-baselines-dqn/

A.3 General Algorithms and Repositories

https://github.com/p-christ/Deep-Reinforcement-Learning-Algorithms-with-PyTorch
https://github.com/openai/baselines

A.4 Reference Papers and Surveys

https://spinningup.openai.com/en/latest/spinningup/keypapers.html
https://arxiv.org/abs/1906.10025

A.5 Latex

https://www.latextemplates.com/template/acm-publications
http://math.mit.edu/~dspivak/files/symbols-all.pdf