**Reinforcement Learning for Games**

Author: Nancy Jemimah Packiyanathan

**Abstract:**

The main idea behind this project is to understand one of the booming fields of artificial intelligence which is Reinforcement Learning. First and foremost, we understand the basic terminologies and the concepts of Reinforcement learning. The problems of Reinforcement learning is formulated as a Markov Decision Process (MDP). Then, we move to intermediate level, to understand more about the value function and policy function. The topics that are main focused in this project are Q learning, Deep Q Learning using Deep Learning networks. With respect to this project, I have created the tutorial blog series to capture my understanding and help other beginners to kickstart in the field of Reinforcement Learning. The links to my articles are mentioned in the below section

**Tutorial Series Created**

1.

[https://medium.com/@nancyjemi/basics-reinforcement-learning-66aae5da4c85](https://medium.com/@nancyjemi/basics-reinforcement-learning-66aae5da4c85 )

2. <https://medium.com/@nancyjemi/level-up-understanding-q-learning-cf739867eb1d>

3. <https://medium.com/@nancyjemi/deep-q-learning-explained-65df980aac6f>

**Key Terminologies**

These are the most common terminologies that are used when we start talking about RL

State: The current situation of the agent.

Action: The decision or the move made by the agent to change its current state to future state

Environment: The place where the agent interacts and performs the action, the environment can be physical or virtual

Agent: The one that we train in our Reinforcement Learning problem to tackle the environment.

Reward: The feedback given to the agent after the move from the current state to the future state. The feedback signal is scalar.

Policy: Optimal policy which is formulated as a strategy, that an agent adopts from moving from one state to another.

Value function: The reward that an agent would get by taking an action in the current state to move to the next future state.

Markov Decision Process

All our problems in Reinforcement learning can be formulated as a Markov decision process. An MDP consists of a set of finite environment states S, a set of possible actions A(s) in each state, a real-valued reward function R(s), and a transition model P(s’, s | a). However, real-world environments are more likely to lack any prior knowledge of environment dynamics.

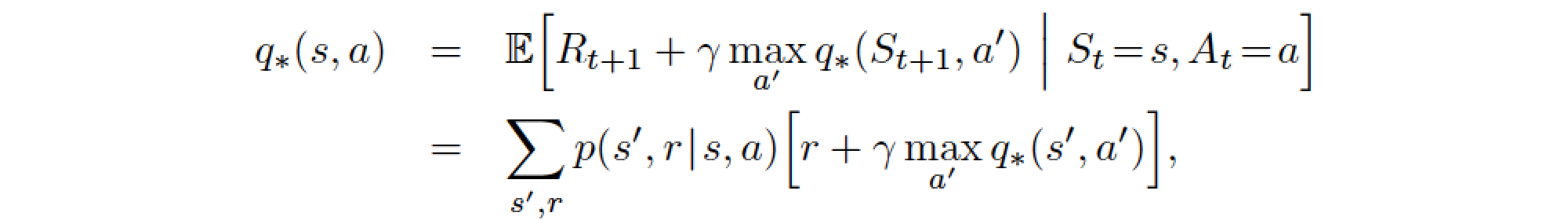
Therefore MDP and agent together give rise to a sequence or trajectory that begins like this:

S0, A0, R1, S1, A1, R2, S2, A2, R3,…..

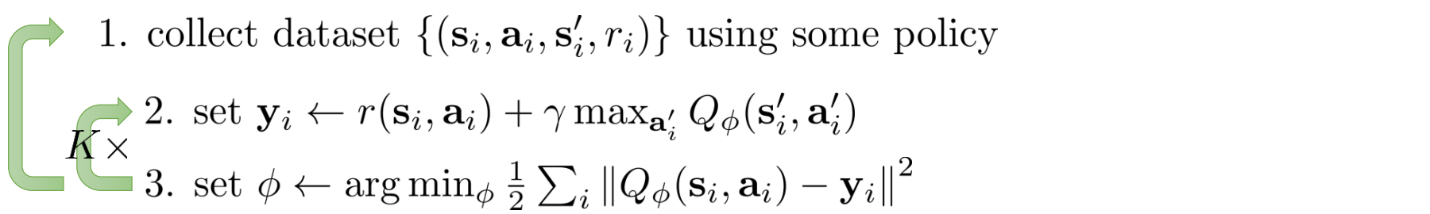
**Implementation**

The implementation of q-learning algorithm pseudocode is given below,

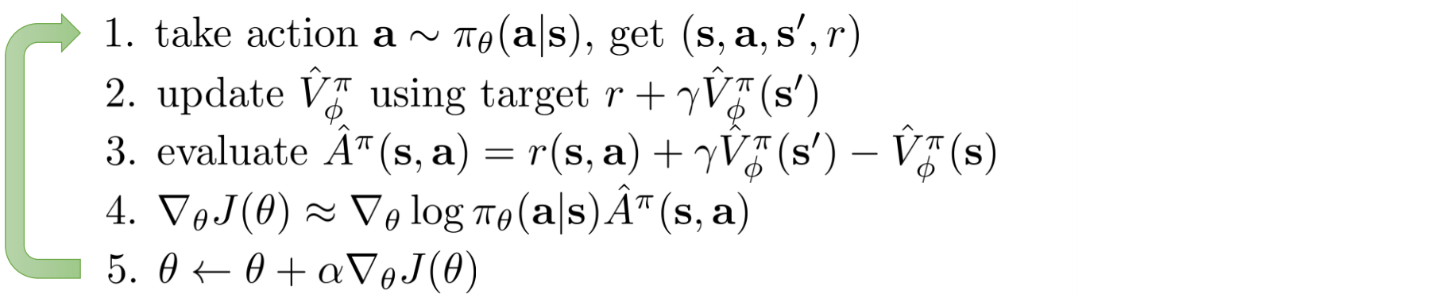
The Bellman equation for the action-value function is:



Estimate the action-value function ***Q*** with a model parameterized by *Φ*.



Full fitted Q-iteration [source](http://rail.eecs.berkeley.edu/deeprlcourse-fa17/index.html)



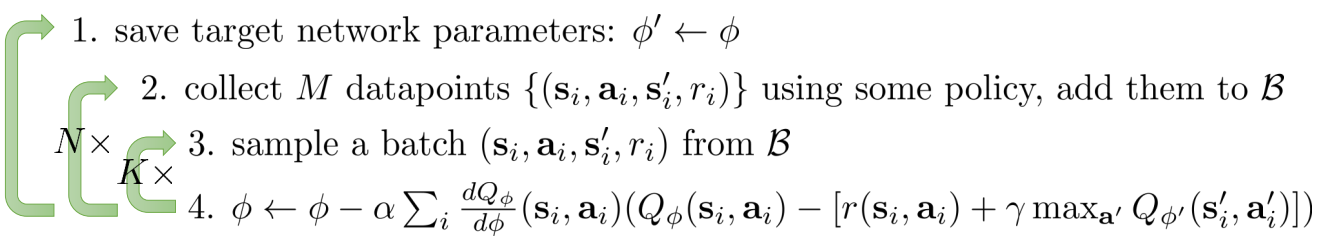
Online Q-iteration [source](http://rail.eecs.berkeley.edu/deeprlcourse-fa17/index.html)

The above-mentioned implementation is done in by jupyter notebook.

**Deep Q Learning algorithm Pseudocode**

**Q-learning with replay buffer and the target network**

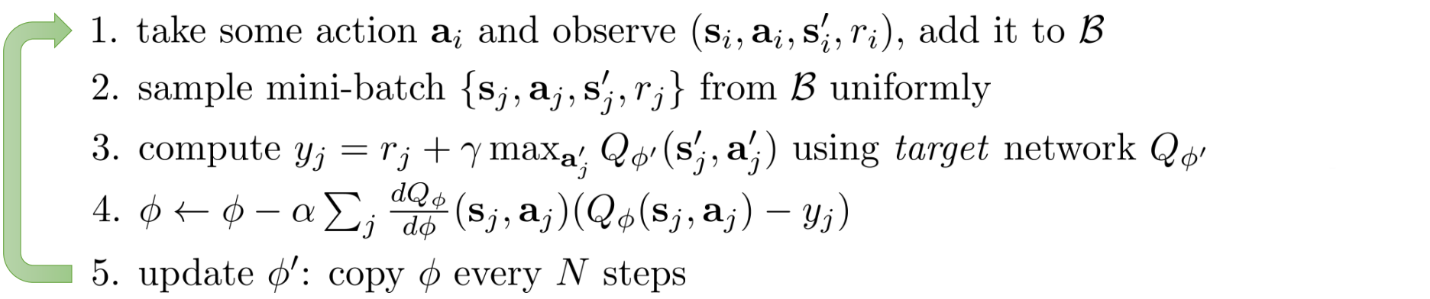
Add a replay buffer and a target network for training stability.



[Source](http://rail.eecs.berkeley.edu/deeprlcourse-fa17/index.html)

**DQN (Deep Q-learning)**

DQN is a Q-learning method with a deep network to estimate ***Q*** using a replay buffer and a target network.



[Source](http://rail.eecs.berkeley.edu/deeprlcourse-fa17/index.html)

**Important hyperparameters**

**Q- learning parameter optimization Strategy**

The values of **`alpha`, `gamma`, and `epsilon`** were mostly based on intuition and some “hit and trial”, but there are better ways to come up with good values.

Ideally, all three should decrease over time because as the agent continues to learn, it actually builds up more resilient priors

* **α:** (the learning rate) should decrease as you continue to gain a larger and larger knowledge base. The learning rate should be in the range of 0 -1. The higher the learning rate, it quickly replaces the new q value. We need to optimize it in a way so that our agent learns from the previous q values. A learning rate is a tool that can be used to find how much we keep our previous knowledge of our experience that needs to keep for our state-action pairs.
* **γ:** as you get closer and closer to the deadline, your preference for near-term reward should increase, as you won’t be around long enough to get the long-term reward, which means your gamma should decrease.
* **ϵ:** as we develop our strategy, we have less need for exploration and more exploitation to get more utility from our policy, so as trials increase, epsilon should decrease.

**Deep Q learning parameter optimization**

**state\_size** = [84, 84, 4] — Our input is a stack of 4 frames hence 84x84x4 (Width, height, channels)  
**action\_size** = env.action\_space.n — no of possible actions  
**learning\_rate** = 0.00025 — Alpha (aka learning rate)

**total\_episodes**= 100 — training episode count  
**max\_steps**= 50000 — maximum steps are taken at each episode  
**batch\_size**= 64 — training sample in the batch

**explore\_start**= 1.0 — exploration probability at start  
**explore\_stop**= 0.01 — minimum exploration probability  
**decay\_rate**= 0.00001 — exponential decay rate for exploration prob

**gamma**= 0.9 — Discounting rate

**pretrain\_length**= batch\_size — Number of experiences stored in the Memory when initialized for the first time  
**memory\_size**= 1000000 — Number of experiences the Memory can keep

**stack\_size**= 4- Number of frames stacked

**Conclusion**

This is the blog that inspired me to create my article with the concepts which I have understood from reading the source available. I have tried to recreate the first three tutorial of this link in my article. Please have a look at it and provide with the feedback and also assessments for each articles are created at the end of each blog.

<https://www.freecodecamp.org/news/diving-deeper-into-reinforcement-learning-with-q-learning-c18d0db58efe/>

The **terminologies and the taxonomy of the Reinforcement Learning** is covered in this report. The main areas that I have concentrated are

1. Q learning
2. Deep Q learning

**Environment used are from openai gym and retro** - Taxi, Mountain Car, CartPole

All the problem in the Reinforcement Learning is considered as a **Markov Decision Process .**

This process consists of set of states, actions and the rewards. **The goal of the Reinforcement Learning is to maximize the reward that the agent** receives in the future. The main idea to this is **the Reward Hypothesis**.

In the real-world, **we can define our Rewards to the environment** in which we want to train our agent

**Q-learning** – Can be used in the environment where the state space is discrete.

**Deep Q learning** – Can be used in the environment where the state space is continuous and for that we implement the deep learning model to calculate our Q value for every iteration of state-action pairs.

**Q learning is the model free, off policy algorithm,** The Q learning is improved by the trail and error method.

**Reference & Citations**

* <https://www.freecodecamp.org/news/diving-deeper-into-reinforcement-learning-with-q-learning-c18d0db58efe/>
* <https://towardsdatascience.com/introduction-to-various-reinforcement-learning-algorithms-i-q-learning-sarsa-dqn-ddpg-72a5e0cb6287>
* <https://deeplizard.com/learn/video/ewRw996uevM>

**License**

MIT License

Copyright (c) 2020 Nancy Jemimah

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.