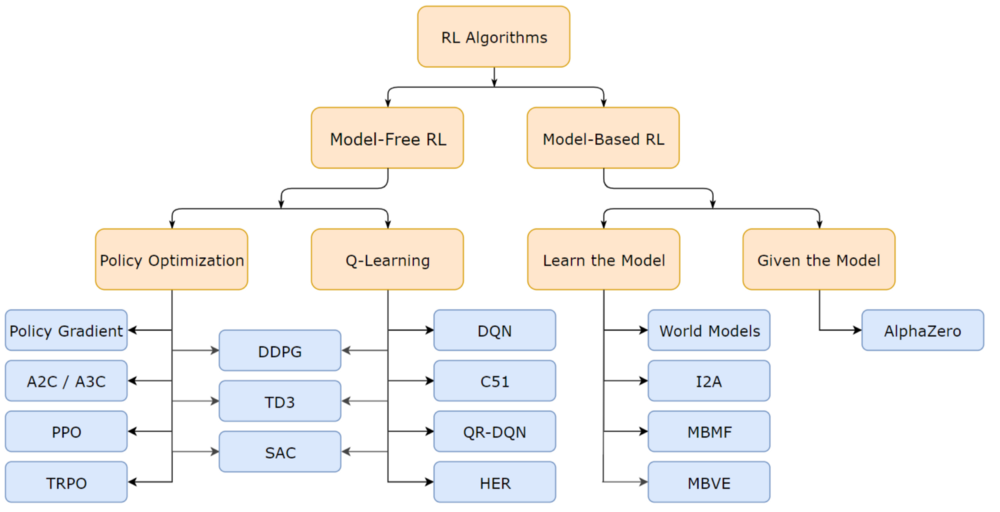
**Short Summary of RL ALgorithms**



# I. Model-free RL

Two main approaches to represent agents with model-free reinforcement learning are Policy optimization and Q-learning.

**I.1. Policy optimization or policy-iteration methods**

In policy optimization methods the agent learns directly the policy function that maps state to action. The policy is determined without using a value function.

Important to mention that there are two types of policies: deterministic and stochastic.

**Deterministic policy** maps state to action without uncertainty. It happens when you have a deterministic environment like a chess table.

**Stochastic policy** outputs a probability distribution over actions in a given state. This process is called Partially Observable Markov Decision Process (POMDP).

**I.1.1. Policy Gradient (PG)**

In this method, we have the policy π that has a parameter θ. This π outputs a probability distribution of actions.

https://miro.medium.com/max/1000/1*zS2nj2jyqPRzhegaBq5zdg.png

Probability of taking action a given state s with parameters theta.

Then we must find the best parameters (θ) to maximize (optimize) a score function J(θ), given the discount factor γ and the reward r.

https://miro.medium.com/max/1000/1*AooPzvc7qcjp7PyFhH5NSw.png - Policy score function

Main steps:

* Measure the quality of a policy with the policy score function.
* Use policy gradient ascent to find the best parameter that improves the policy.

A great and detailed explanation with all the math included about policy gradient can be found in [Jonathan Hui](https://medium.com/@jonathan_hui)’s [blog](https://medium.com/@jonathan_hui/rl-policy-gradients-explained-9b13b688b146) or in [Thomas Simonini](https://medium.freecodecamp.org/@thomassimonini)’s [introduction blog](https://medium.freecodecamp.org/an-introduction-to-policy-gradients-with-cartpole-and-doom-495b5ef2207f) to PG with examples in Tensorflow.

**I.1.2. Asynchronous Advantage Actor-Critic (A3C)**

This methods was published by Google’s DeepMind group and covers the following key concept embedded in it’s naming:

* **Asynchronous:** Several agents are trained in it’s own copy of the environment and the model form these agent’s are gathered in a master agent. The reason behind this idea, is that the experience of each agent is independent of the experience of the others. In this way the overall experience available for training becomes more diverse.
* **Advantage:** Similarly to PG where the update rule used the dicounted returns from a set of experiences in order to tell the agnet which acttions were “good” or “bad”.
* **Actor-critic:** combines the benefits of both approaches from policy-iteration method as PG and value-iteration method as Q-learning (See below). The network will estimate both a value function ***V(s)*** (how good a certain state is to be in) and a policy ***π(s).***

A simple but throughout explanation with code implemented in Tensorflow can be found in [Arthur Juliani blog.](https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2)

**I.1.3. Trust Region Policy Optimization (TRPO)**

A on-policy algorithm that can be used or environments with either discrete or continuous action spaces. TRPO updates policies by taking the largest step possible to improve performance, while satisfying a special constraint on how close the new and old policies are allowed to be.

A comprehensive introduction is provided on TRPO in [this](https://medium.com/@jonathan_hui/rl-trust-region-policy-optimization-trpo-explained-a6ee04eeeee9) and [this](https://medium.com/@jonathan_hui/rl-trust-region-policy-optimization-trpo-part-2-f51e3b2e373a) blog post and a great [repo](https://github.com/pat-coady/trpo) provides Tensorflow and OpenAI Gym based solutions.

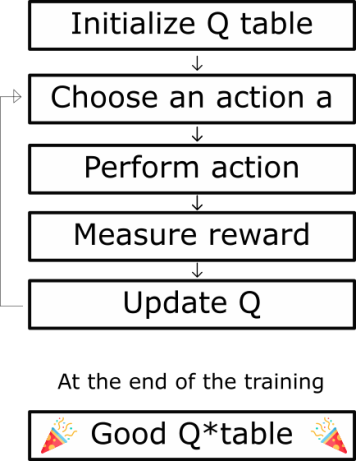
**I.1.4. Proximal Policy Optimization (PPO)**

Also an on-policy algorithm which similarly to TRPO can perform on discrete or continuous action spaces. PPO shares motivation with TRPO in the task of answering the question: how to increase policy improvement without the risk of performance collapse? The idea is that PPO improves the stability of the Actor training by limiting the policy update at each training step. PPO became popular when OpenAI made a breakthrough in Deep RL when they released an algorithm trained to play Dota2 and they won against some of the best players in the world. See description on [this](https://blog.openai.com/openai-five/) page.

For deep dive into PPO visit [this](https://towardsdatascience.com/proximal-policy-optimization-ppo-with-sonic-the-hedgehog-2-and-3-c9c21dbed5e) blog.

**I.2. Q-learning or value-iteration methods**

Q-learning learns the action-value function *Q(s, a)*:how good to take an action at a particular state*. Basically a scalar value is assigned* over an action a given the state s. The following chart provides a good representation of the algorithm.



Q-learning steps [[Source](https://medium.freecodecamp.org/diving-deeper-into-reinforcement-learning-with-q-learning-c18d0db58efe)]

**I.2.1 Deep Q Neural Network (DQN)**

DQN is Q-learning with Neural Networks . The motivation behind is simply related to big state space environments where defining a Q-table would be a very complex, challenging and time-consuming task. Instead of a Q-table Neural Networks approximate Q-values for each action based on the state.

For deep dive to DQN visit this [course](https://medium.freecodecamp.org/an-introduction-to-deep-q-learning-lets-play-doom-54d02d8017d8) and play Doom meanwhile.

**I.2.2 C51**

C51 is a feasible algorithm proposed by Bellemare et al. to perform iterative approximation of the value distribution Z using Distributional Bellman equation**.** The number 51 represents the use of 51 discrete values to parameterize the value distribution Z(s,a). See the original paper [here](https://arxiv.org/pdf/1707.06887.pdf) and for a deep dive follow this exploratory [tutorial](https://flyyufelix.github.io/2017/10/24/distributional-bellman.html) with implementation in Keras.

**I.2.3 Distributional Reinforcement Learning with Quantile Regression (QR-DQN)**

In QR-DQN for each state-action pair instead of estimating a single value a distribution of values values in learned. The distribution of the values, rather than just the average, can improve the policy.This means that quantiles are learned which threshold values attached to certain probabilities in the cumulative distribution function. See paper for the method [here](https://arxiv.org/pdf/1710.10044.pdf) and an easy implementation using Pytorch [here](https://github.com/senya-ashukha/quantile-regression-dqn-pytorch) .

**I.2.4 Hindsight Experience Replay (HER)**

In [Hindsight Experience Replay](https://arxiv.org/pdf/1707.01495.pdf) method, basically a DQN is suplied with a state and a desired end-state, or in other words goal. It allow to quickly learn when the rewards are sparse. In other words when the rewards are uniform for most of the time, with only a few rare reward-values that really stand out.

For a better understanding, beside the paper check out [this](https://becominghuman.ai/learning-from-mistakes-with-hindsight-experience-replay-547fce2b3305) blog post, fr coding [this](https://github.com/localminimum/hindsight-experience-replay) github repository

**I.3 Hybrid**

Simply as it sounds, these methods combine the strengths of Q-learning and policy gradients, thus the policy function that maps state to action and the action-value function that provides a value for each action is learned.

Some hybrid model-free algorithms are:

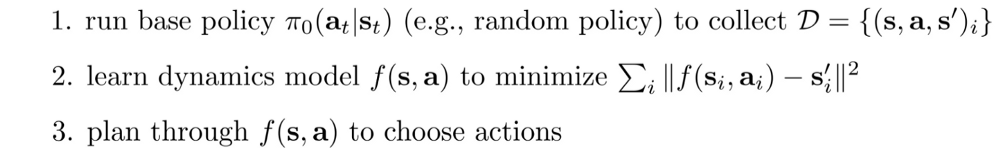
* Deep Deterministic Policy Gradients (DDPG): [paper](https://arxiv.org/pdf/1509.02971.pdf) and [code](https://pemami4911.github.io/blog/2016/08/21/ddpg-rl.html),
* Soft Actor -Critic (SAC): [paper](https://arxiv.org/abs/1801.01290) and [code](https://github.com/haarnoja/sac).
* Twin Delayed Deep Deterministic Policy Gradients (TD3) [paper](https://arxiv.org/pdf/1802.09477.pdf) and [code](https://github.com/sfujim/TD3)

**II. Model-based RL**

Model-based RL has a strong influence from control theory, and the goal is to plan through an *f(s,a)* control function to choose the optimal actions. Think of it as the RL field where the laws of physics are provided by the creator. The drawback of model-based methods is that although they have more assumptions and approximations on a given task, but may be limited only to these specific types of tasks. There are two main approaches: learning the model or learn given the model.

**II.1. Learn the Model**

To learn the model a base policy is ran, like a random or any educated policy, while the trajectory is observed. The model is fitted using the sampled data. Below steps describe the procedure:



Supervised learning is used to train a model to minimize the least square error from the sampled data for the control function. Optimal trajectory using the model and a cost function is used in step three. The cost function can measure how far we are from the target location and the amount of effort spent.

* World models: one of my favourite approaches in which the agent can learn from it’s own “dreams” due to the Variable Auto-encoders, See [paper](https://arxiv.org/pdf/1803.10122.pdf) and [code](https://github.com/hardmaru/WorldModelsExperiments).
* Imagination-Augmented Agents (I2A): learns to interpret predictions from a learned environment model to construct implicit plans in arbitrary ways, by using the predictions as additional context in deep policy networks. Basically it’s a hybrid learning method because it combines model-baes and model-free methods. [Paper](https://arxiv.org/pdf/1707.06203.pdf) and [implementation](https://github.com/gitlimlab/i2a-tf).
* Model-Based Priors for Model-Free Reinforcement Learning (MBMF): aims to bridge the gap between model-free and model-based reinforcement learning. See [paper](https://arxiv.org/abs/1709.03153) and [code](https://github.com/Jerryxiaoyu/MBMF).
* Model-Based Value Expansion (MBVE): Authors of the [paper](https://arxiv.org/pdf/1803.00101.pdf) state that this method controls for uncertainty in the model by only allowing imagination to fixed depth. By enabling wider use of learned dynamics models within a model-free reinforcement learning algorithm, we improve value estimation, which, in turn, reduces the sample complexity of learning.

**II.2. Given the Model**

* I would say this had the “hypest” hype in recent time when AlphaGo Zero defeated the best go player in the world. You can found anything you want on Deep Mind’s [website](https://deepmind.com/research/alphago/alphazero-resources/).

Content Reference:

https://smartlabai.medium.com/reinforcement-learning-algorithms-an-intuitive-overview-904e2dff5bbc