Vocabulary to "Reinforcement Learning: Multiarmed Bandit Problem, Finite MDPs", online: github.com/fewagner/ReinforcementLearning

ϵ-greedy Algorithm • Method to balance exploration and exploitation. In every step, with probability ϵ a random non-greedy action is taken. Otherwise the greedy action is taken. In the latter case, the value of the action state pair is updated according to some update rule.

Action • An action that the agent can take.

Action Space $\mathcal{A}$ • The set of all actions that the agent can take. Sometimes $\mathcal{A}_s$ referes to all actions the agent can take when in a certain state.

Action Value Funtion q_π • Assigns to every action state pair the expected return, when the policy π is followed thereafter. If π is an optimal policy π_*, it is called the Optimal Action Value Function q_*.

Agent • The individual in the reinforcement learning algorithm, that makes decisions based on its environment. Example: In a Tic-Tac-Toe game, the agent is the player who decides where to set it's stones, based on the position of all stones on the table.

Bandit Problem • A slot machine with one or multiple levers. Each lever gives a different probabilistic reward. In the bandit problem, the agent tries to learn which lever has the highest expected reward. The agent is always in the same state, so it only learns action values corresponding to the different levers.

Bellman Equation • Recursive Relation of the value function:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\big[r + \gamma v_\pi(s')\big]$$

The value function is the unique solution of the Bellman Equation. It can be calculated by solving the system of linear equations.

Bellman Optimality Equation • Bellman Equation of the optimal value function or optimal state value function:

$$v_*(s) = \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma v_*(s')\big]$$

$$q_*(s,a) = \sum_{s',r} p(s',r|s,a)\big[r + \gamma \max_{a'} q_*(s',a')\big]$$

Continuing Problem • A Problem that does not end after a finite number of steps.

Discounting Factor γ • For continuing problems the Return is as a simple sum of expected future rewards ill defined, as the sum would diverge. So $\gamma \in [0,1]$ is indroduced to define the discounted Return:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Episode • An episodic task is one that ends after finitely many steps. e.g. A chess play ends when one of the opponents wins. One game is then called an episode.

Environment • Everything that is not part of the agent. When the agent sets an action, the environment answers with a reward and a new state. When designing Machine Learning (ML) problems, one has to decide what is part of the agent and vice versa. Rule of thumb: "Everything that cannot be changed arbitrarily by the agent is part of the environment."

Exploitation • When the agent takes the greedy action. He might miss better options when only exploiting.

Exploration • When the agent takes non-greedy actions to get a better picture of its environment. Typically, after explorative actions the action state value is not updated.

Gradient Bandit Algorithm • Uses preferences instead of action values. Policy:

$$Pr\{A_t = a\} = \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} = \pi_t(a)$$

Preferences Update Rule:

$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \overline{R_t})(1 - \pi_t(A_t))$$

$$H_{t+1}(A_t) = H_t(a) + \alpha(R_t - \overline{R_t})\pi_t(a)$$

for all $a \neq A_t$.

Greedy Action • The action with the highest estimated Return.

Learning Rate α • The stepsize parameter in Q Learning.

Markov Chain • A MDP with only one possible action in every state, that is choosen automatically.

Markov Property • A state satisfies the Markov property when it contains all the relevant information for future development. e.g. All positions of all "X" and "O" in a tic tac toe game are together a state that satisfies the markov property. For the Travelling Salesman Problem, the current position alone does not satisfy the Markov property. Additionally it needs the information where the agent has already been.

Model of the Environment • A Priori Knowledge about the environment can be introduced to the Reinforcement Learning Problem by creating a model of the agents environment. e.g. When the agent tries to escape a labyrinth, the model could tell him that the walls won't change position over time.

MPD • A Markov Decision Process is a State-, an Action-, and a Reward Space, together with a function

$$p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$$

defined by

$$\begin{aligned} &p(s', r, s, a) = \dots \\ &Pr\{S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a\} \end{aligned}$$

and a function

$$r : \begin{cases} \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{R} \\ (s, a, s') \mapsto r(s, a, s') \end{cases}$$

that accounts for the expected rewards. Then all states satisfy the Markov Property.

Optimal Policy π_* • A policy π is better than π' if $v_\pi(s) \geqslant v_{\pi'}(s)$ for all s. An optimal policy π_* is better than or equal to all others. There exists always at least one such policy.

Optimistic Initial Values • ϵ-greedy method with unrealistically high initial values. It forces the agent to explore in the beginning.

Policy π • Mapping from states to probabilities of selecting each possible action: $\pi(a|s) = Pr(A_t = a|S_t = s)$

Q Learning • All action state values are contained in the so called Q table. Then an epsilon greedy method is applied with linearly decreasing epsilon for a certain number of episodes. An update rule, modelled after the Bellman Equation, is applied:

$$\begin{aligned} &Q(s_t, a_t) \leftarrow (1 - \alpha) Q(s_t, a_t) + \dots \\ &\alpha [R(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a)] \end{aligned}$$

Reinforcement Learning • The description of a ML problem by modelling it with an agent in an certain state. The agent takes actions in discrete time steps, getting answers from an environment in form of rewards and new states.

Return G • An estimation of the future rewards. In the simplest form: $G_t = R_{t+1} + R_{t+2} + \dots + R_T$. This simple form is only applicable to episodic tasks.

Reward • A scalar value that the agent gets from its environment as response to an action.

Reward Hypothesis • The hypothesis, that everything that an individuum wants to achieve can be described with a scalar value, the cummulative reward. In human decision making, problems are often more complex.

Reward Space $\mathcal{R}$ • The set of all possible rewards.

State • All information that the agent has about his current position in the task. e.g. In a game of chess, the position of all figures on the board is the state.

State Space $\mathcal{S}$ • The set of all possible states.

Supervised Learning • A task in which a ML model has to learn from labeled data. e.g. handwritten character recognition

Travelling Salesman Problem • The Agent has to visit certain places while driving the least distance possible.

UCB • Method that selects non-greedy actions according to their potential to be greedy. Action Selection Rule:

$$A_t = arg \max_a \big[Q_t(a) + c \sqrt{\frac{ln(t)}{N_t(a)}} \big]$$

$N_t(a)$ ... number of times a was selected prior to time step t, $c > 0$ ... controls the degree of exploration

Unsupervised Learning • A task in which a ML model has to find structure in unlabeled data.

Update Rule • A rule according to which the values of action state pairs are updated after usage. Usually its structure looks like: $NewEstimate \leftarrow OldEstimate + Stepsize \big[Target - OldEstimate \big]$

Value Function v_π • Assigns to every state its expected return, when following policy π. If π is an optimal policy π_*, it is called the Optimal Value Function v_*.