REINFORCEMENT LEARNING
for
CYBER-PHYSICAL SYSTEMS

Provide by

Yury Chernyshov

Alexey Sinadsky

Students:

Fadhil Fadhil Abbas

Hasan Mohamed Ali

Group: РИМ 201211

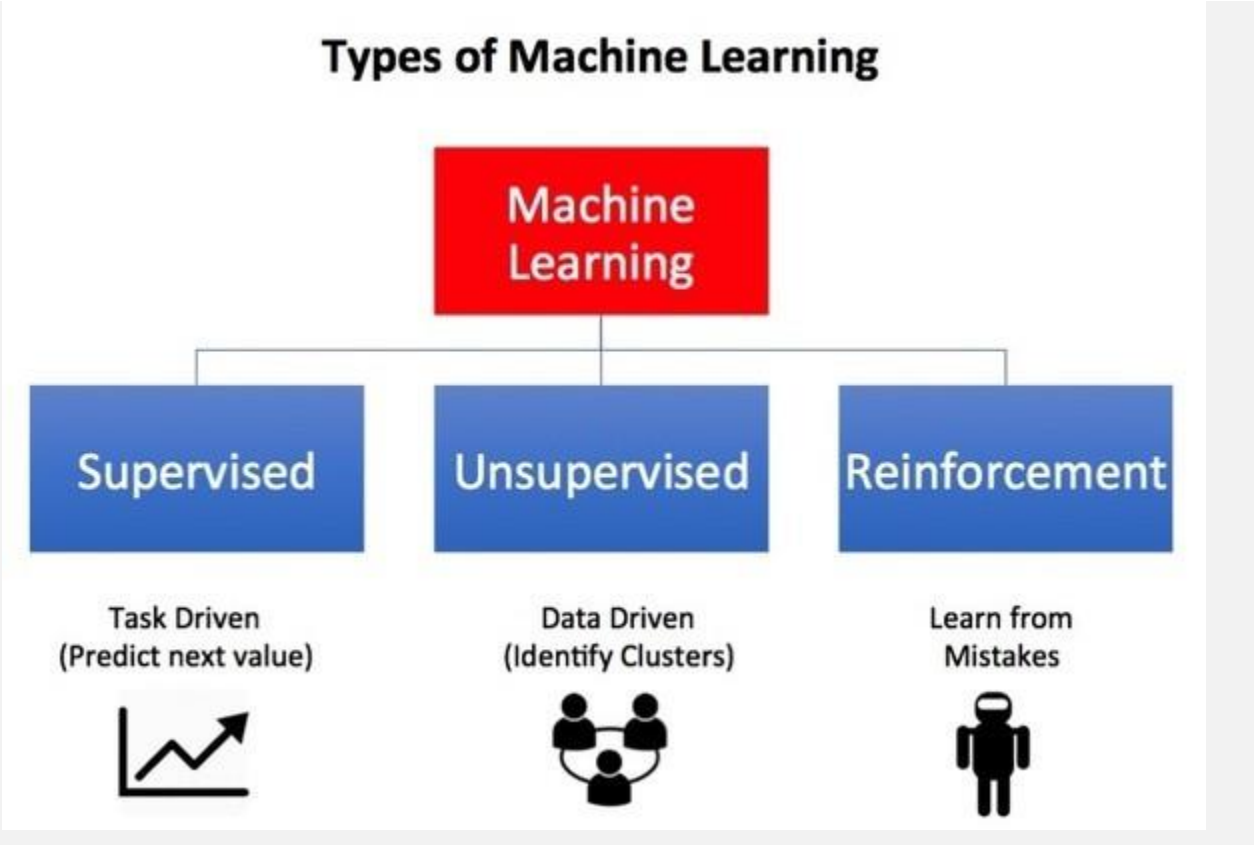
Yekaterinburg

2021

Contents

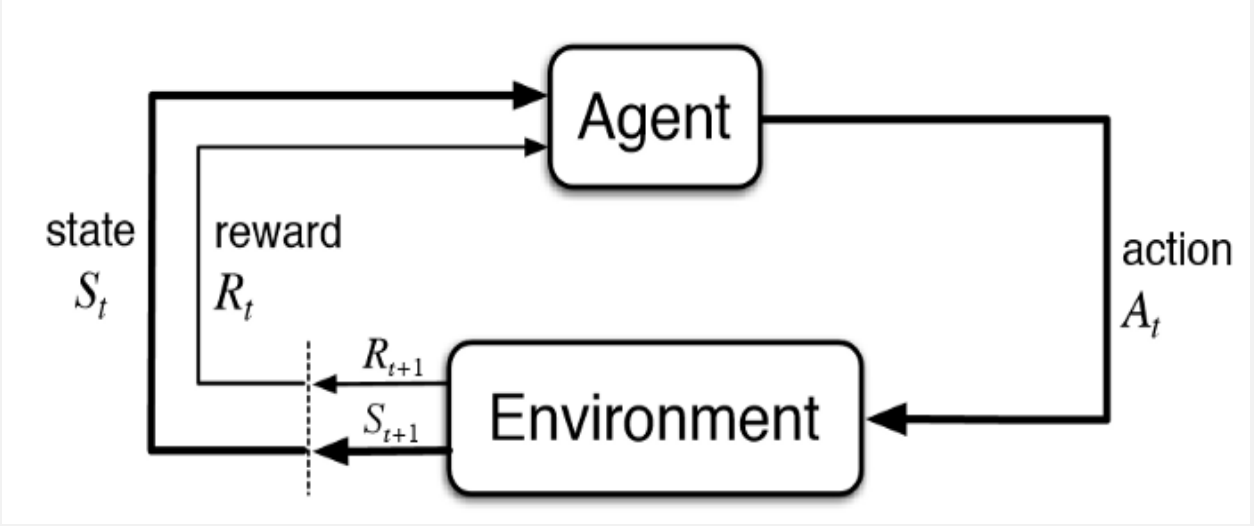
1. Reinforcement Learning (RL)

Reinforcement Learning (RL) is a type of machine learning technique that enables an agent to learn in an interactive environment by trial and error using feedback from its own actions and experiences.



Though both supervised and reinforcement learning use mapping between input and output, unlike supervised learning where the feedback provided to the agent is **correct set of actions** for performing a task, reinforcement learning uses **rewards and punishments** as signals for positive and negative behavior.

As compared to unsupervised learning, reinforcement learning is different in terms of goals. While the goal in unsupervised learning is to find similarities and differences between data points, in the case of reinforcement learning the goal is to find a suitable action model that would maximize the **total cumulative reward** of the agent. The figure below illustrates the **action-reward feedback loop** of a generic RL model.



2. Reinforcement Learning problem

Some key terms that describe the basic elements of an RL problem are:

1. Environment — Physical world in which the agent operates
2. State — Current situation of the agent
3. Reward — Feedback from the environment
4. Policy — Method to map agent's state to actions
5. Value — Future reward that an agent would receive by taking an action in a particular state

An RL problem can be best explained through games. Let's take the game of PacMan where the goal of the agent (PacMan) is to eat the food in the grid while avoiding the ghosts on its way. In this case, the grid world is the interactive environment for the agent where it acts. Agent receives a reward for eating food and punishment if it gets killed by the ghost (loses the game). The states are the location of the agent in the grid world and the total cumulative reward is the agent winning the game.

In order to build an optimal policy, the agent faces the dilemma of exploring new states while maximizing its overall reward at the same time. This is called Exploration vs Exploitation trade-off. To balance both, the best overall strategy may involve short term sacrifices. Therefore, the agent should collect enough information to make the best overall decision in the future.

Markov Decision Processes (MDPs) are mathematical frameworks to describe an environment in RL and almost all RL problems can be formulated using MDPs. 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. Model-free RL methods come handy in such cases.

Q-learning is a commonly used model-free approach which can be used for building a self-playing PacMan agent. It revolves around the notion of updating Q values which denotes value of performing action a in state s. The following value update rule is the core of the Q-learning algorithm.

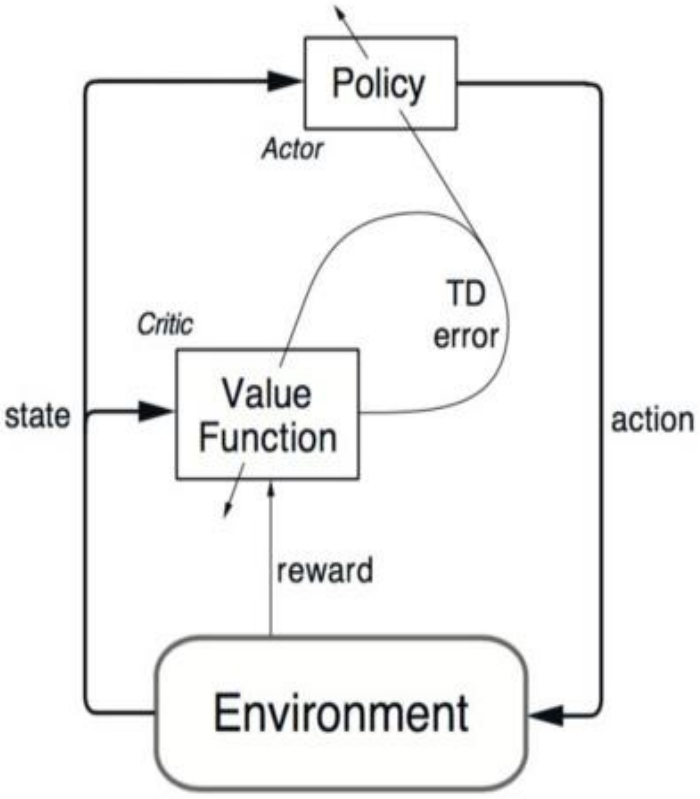
$$Q(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

3. RL algorithms usage

Q-learning and SARSA (State-Action-Reward-State-Action) are two commonly used model-free RL algorithms. They differ in terms of their exploration strategies while their exploitation strategies are similar. While Q-learning is an off-policy method in which the agent learns the value based on action a* derived from another policy, SARSA is an on-policy method where it learns the value based on its current action a derived from its current policy. These two methods are simple to implement but lack generality as they do not have the ability to estimates values for unseen states.

This can be overcome by more advanced algorithms such as Deep Q-Networks (DQNs) which use Neural Networks to estimate Q-values. But DQNs can only handle discrete, low-dimensional action spaces.

Deep Deterministic Policy Gradient (DDPG) is a model-free, off-policy, actor-critic algorithm that tackles this problem by learning policies in high dimensional, continuous action spaces. The figure below is a representation of actor-critic architecture.



4. Applications of Reinforcement Learning

Reinforcement learning has paved the way to autonomy of a number of fields. Examples include industrial robot, autonomous car, share market trading, blackout protection in power sector, and of course gaming agents like AlphaGo. In what follows, we highlight a few use cases from the above sectors.

1. Industrial robot: In a manufacturing setting, robots are trained with reinforcement learning algorithms. For example, a sheet has to be cut following a curved line. A robot is trained using Q-learning algorithm that can navigate along a curve to cut a metal sheet. Amazon has also started using industrial robots for inventory management.

2. Autonomous car: Self-driving car is much researched in the autonomous engineering community. The car can identify the lane lines, regulate its speed based on its surrounding vehicles, change lanes, and most importantly stop at the right signals. While autonomous cars exploit computer vision algorithms for identification and detection, reinforcement learning algorithms are used to regulate its telemetry and trajectory. For instance, the double deep-Q learning algorithm has been used to control a simulated car.

3. Share market trading: Financial giants like Goldman Sachs, JP Morgan Chase, and Morgan Stanley use deep reinforcement learning algorithms to regulate stock prices. A typical problem is when to sell stocks in the market so as to maximize stakeholder's profit. For this real-life challenge, reinforcement learning algorithms can calculate in real time the right price to bid with, and the optimal quantity of shares to present for sale in order to achieve near optimal profit for a given time horizon. For instance, a fusion of the popular Q learning algorithm and dynamic programming has been used to solve the above problem.

4. Blackout protection in the power sector: Blackout refers to a power outage when all generators in a connected grid shut down at the right time. It happens due to a chain of events when power demand peaks abruptly. For the generators to start and run in a power system, a spin-reserve is to be maintained. However, if a simultaneously large amount of power is drained out from different distribution centers around the country, sufficient spin-reserve cannot be maintained. As a consequence, generators shut down one by one and there is no power left in the grid. Such a blackout is catastrophic because it results in power failure for essential needs such as cellular communication, internet connection, and heat or air conditioning. One such blackout happened in the northeastern region of the United States where 55 million people were affected. Nowadays, a reinforcement learning agent is delegated with the responsibility to monitor millions of distribution points to properly regulate power draining in order to prevent the blackout.

5. Games: Google DeepMind created an intelligent computer that can beat a professional human player in the Go game in 2015. The computer is trained with a reinforcement learning algorithm such that after sufficient training, it can perform at a human level. The reinforcement learning era caught significant attention recently when it was able to beat a human in a game. In March 2016, it beat professional Go player Lee Sedol in a five-game match. The detailed methodology used in the intelligent computer AlphaGo.

6. Computer vision: After the development and advancement of deep neural networks, given an image of a cow, car, or baby, the neural network can recognize the object correctly. However, the problem that is not yet adequately addressed is the real-time detection of the object. Take our previous shark sighting problem.

7. Natural language processing: Natural language processing refers to understanding words from speech, parsing texts to produce a summary, or answering questions asked by a human. Nowadays, when we call the customer care center of any service, they allow us to say in a few words why we are calling them. Their automated system is capable of intelligently interacting with a human caller and serve his or her needs. Deep learning and reinforcement learning algorithms are widely used for this kind of speech-to-text conversation. A good survey is given that covers the development of reinforcement learning systems trained with maximum likelihood word prediction from dialogue and nonrepetitive summary production from texts.

5. SIMULATION TOOLKITS FOR REINFORCEMENT LEARNING

Reinforcement Learning is essentially a computational approach in which an agent interacts with an environment by taking actions to maximize its accumulated rewards. Therefore, to evaluate a reinforcement learning algorithm in simulations, one has to create an environment and the agent environment interface. If the environment is complicated, this job could be very non-trivial and time costly. As motivated by this fact, several reinforcement learning toolkits have been developed as a collection of environments designed for testing, developing and comparing reinforcement learning algorithms.

6. OpenAI Gym

OpenAI Gym is a toolkit for reinforcement learning research. It includes a growing collection of benchmark problems that expose a common interface, and a website where people can share their results and compare the performance of algorithms.

For example, of the open-source OpenAI Gym which supports training agents in everything from walking to playing the "Frozen Lake" game, which we will be explained in the practical part.