Reinforcement learning

Yulia Newton, Ph.D.

CS156, Introduction to Artificial Intelligence

San Jose State University

Spring 2021

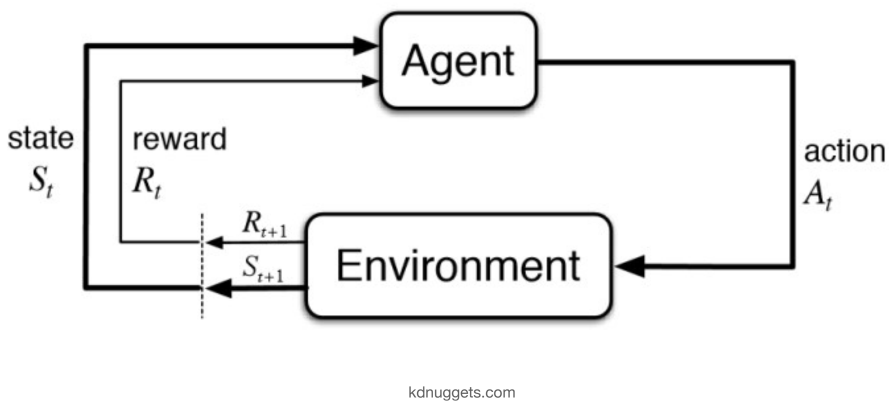
What is reinforcement learning?

- Reinforcement learning is a subfield of ML that studies decision making
  - How does an agent choose an action, given a space of possible actions and the environment, in order to maximize the reward
- Falls between supervised and unsupervised learning
  - Does not strictly rely only on a set of labeled training examples
  - Is not unsupervised as relies on a reward, which we try to maximize

Typical reinforcement learning setting

- **Agent** - the ML program/model that is being trained
- **Environment** - the setting and outside influence in the context of which the agent performs actions
- **State** - current situation in the agent's environment
- **Action** - an action taken by the agent; an action changes the status in the environment
- **Reward** - evaluation of the action taken by the agent; can be positive or negative (we can call it penalty)
- **Policy** - a method of mapping the state to an action (strategy for making a decision)
- **Value** - the expected long-term cumulative reward of the current state

Schematic view of reinforcement learning framework



Example of RL: movie recommender

- **Agent** - the program that decides what to present in the suggested movies section
- **Environment** - movie streaming system
- **Action** - suggest movie
- **Reward** - positive if user chooses to play the movie, negative if the user does not choose to play the movie

Example of RL: walking robot

- **Agent** - the program that controls the robot
- **Environment** - terrain in which the robot is walking
- **Action** - taking a step in a given direction (forward, backward, left, or right)
- **Reward** - positive when the robot approaches the target destination without falling or running into an obstacle, negative otherwise

Simple ways to define reinforcement learning

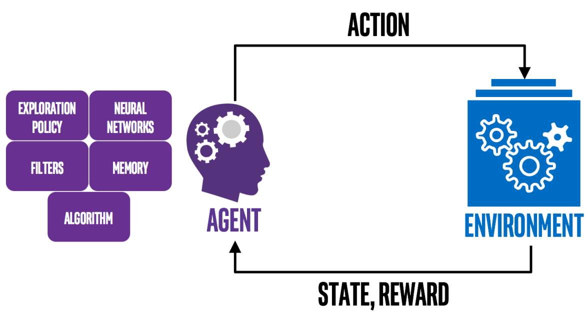
- Reinforcement learning is learning from interactions with the environment through evaluating each action

- Reinforcement learning is the science of decision making by mapping situations to actions

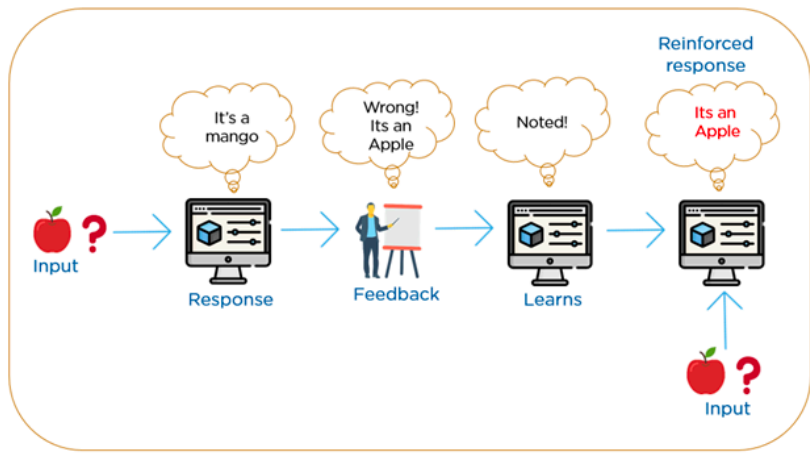
- Reinforcement learning framework provides a computational approach to learning from interactions

- Reinforcement learning is similar to how humans learn

Feedback from the environment determines the action the agent takes



https://medium.com



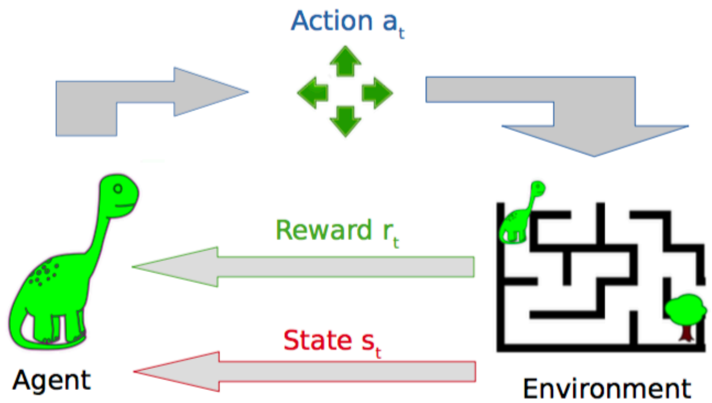
https://towardsdatascience.com

Simple motivation example

- Let’s say we want to teach a computer to play a game of Tetris
- We can have a very skilled player play this game for hours and record every frame presented to the player and every action this player takes
- We can then train a neural network to play this game based on the training data
- Our model will never be a better player than the human player because, by definition, the model cannot perform better than the training examples it is provided to learn from
- Can the agent learn to play the game on its own without supervised training examples?

Learning through the feedback from the environment instead of from labeled outcomes

- The model can learn from "observing" how the predictions change the environment rather than comparing to predetermined training labels



https://www.inteliment.com/blog/our-thinking/deep-reinforcement-learning-and-its-applications

Common applications of reinforcement learning

- Robotics
- Resource management in a data center
- Traffic lights controller
- Computational chemistry (optimizing chemical reactions)
- Recommender systems
- Advertising and ad placement
- Gaming
- Text mining
- Stock trading
- Natural language processing
- etc.

So how do we approach the problem then?

- In the traditional supervised learning setting we have:
  - The input data
  - The model (e.g. deep neural network)
  - The output labels
  - We train the model by comparing the predicted output to the actual output labels

So how do we approach the problem then? (cont’d)

- In the RL setting we do not have output labels
  - The model is called the “policy model” (e.g. “policy network”)
    - Usually a deep neural network
  - Reward is a scalar feedback value, indicating how well the agent performed at step t
  - The goal is to select actions to maximize total future reward
    - Reward hypothesis (next slide)
  - Value function computes the mapping between the action and the state
  - Utilize “policy gradients” to learn the model parameters

Reward hypothesis

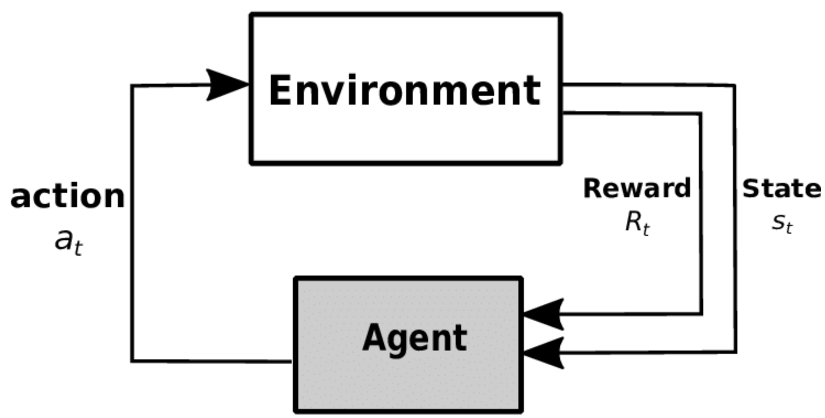
- A goal for any RL problem can be described by maximization of expected cumulative reward
  - Optimization problem of maximizing total reward
- The total future reward at any time point t can be presented as:
  - $G_t = R_{t+1} + R_{t+2} + R_{t+3} + \ldots$
- The rewards can be discounted at a discount rate, depending on how much the agent cares about short/long term rewards:
  - $G_t = R_{t+1} + \boldsymbol{\gamma} R_{t+2} + \boldsymbol{\gamma}^2 R_{t+3} + \ldots$

RL is sequential decision making

- Sequence of making decision, taking action, and calculating reward
- Early-on decisions can affect long-term consequences
- Sometimes it's better to sacrifice short-term reward to gain greater long-term reward
  - E.g. in chess we gladly sacrifice a pawn to capture the bishop later
- Reward hypothesis accounts for long-term consequences

Setting up an RL problem

- At a time point t the agent makes a decision to proceed with action a_t
- This decision is made based on reward R_t received from observing environment state s_t
  - The decision is made using a policy π
- The environment receives the action a_t and changes its state accordingly to new s_{t+1}



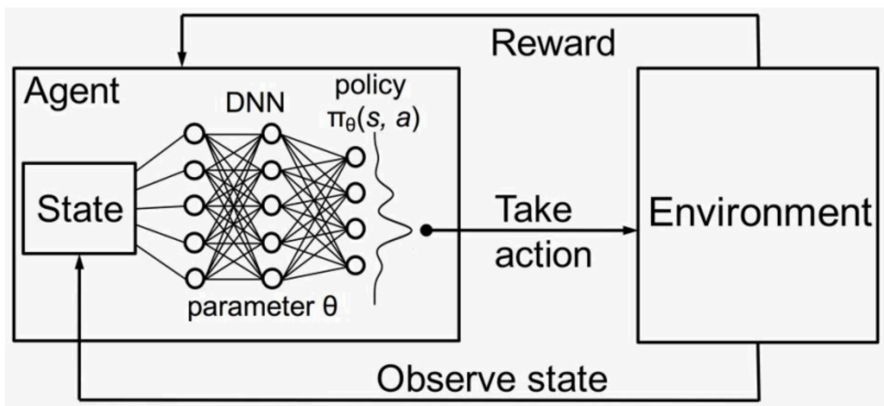
"A Machine Learning Approach for Power Allocation in HetNets Considering QoS", Amiri et al. 2018

Positive vs. negative learning

- Based on the value of the reward we can think of RL as positive or negative
  - Positive if the reward is positive
  - Negative if the reward is negative (penalty)
  - Hybrid (both reward and penalty are used)

Deep reinforcement learning

- When an RL model uses an artificial neural network it is called “deep reinforcement learning”



https://medium.com

Supervised vs. RL learning

	Supervised learning	Reinforcement learning
Definition	Works on existing or given sample data or examples	Works on interacting with the environment
Preference	Assets are depreciable	Liabilities are non-depreciable
Tasks	Classification and regression	Exploitation and exploration
Mapping between input and output	Both input and output will be available for decision making where the learner will be trained on many examples or sample data are given	Sequential decision making happens and the next input depends on the decision of the learner or system
Platform	Operated with interactive software or applications	Supports and works better in AI where human interaction is prevalent
Algorithms	Many algorithms exist in using this learning	Neither supervised nor unsupervised algorithms are used
Integration	Runs on any platform or with any applications	Runs with any hardware or software devices

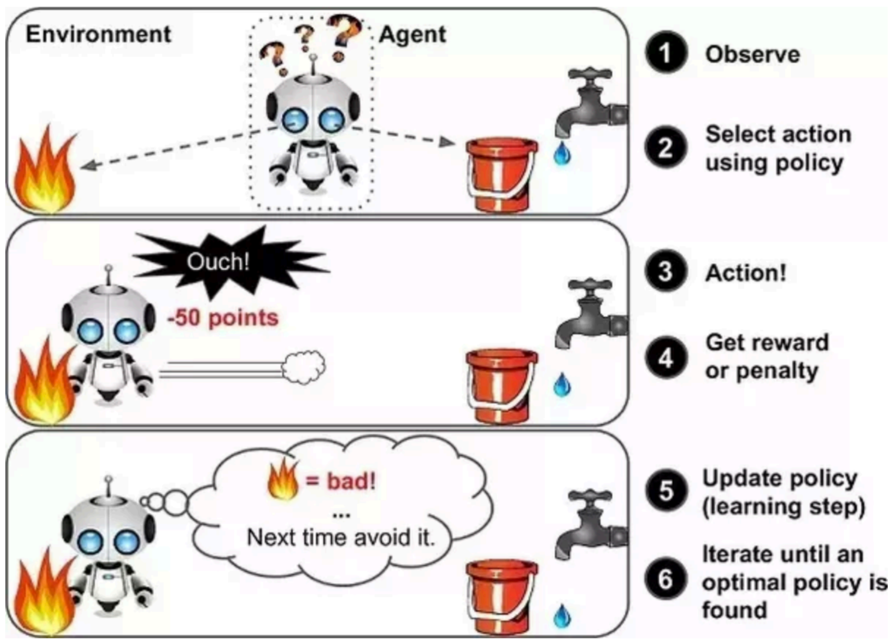
https://towardsdatascience.com

Unsupervised vs. RL learning

	Unsupervised learning	**Reinforcement learning**
Definition	No external teacher or pre-trained data	Works on interacting with the environment
Preference	Assets are depreciable	Liabilities are non-depreciable
Tasks	Clustering and association	Exploitation and exploration
Mapping between input and output	To find the underlying patterns rather than the mapping.	Will get constant feedback from the user by suggesting few news articles and then build a "knowledge graph"
Platform	Operated with interactive software or applications	Supports and works better in AI where human interaction is prevalent
Algorithms	Many algorithms exist in using this learning	Neither supervised nor unsupervised algorithms are used
Integration	Runs on any platform or with any applications	Runs with any hardware or software devices

https://towardsdatascience.com

Learning parameters of the model (RL optimization)



https://towardsdatascience.com

RL optimization approaches

- Different approaches have different optimization goals

- **Value based**
  - The goal is to optimize the value function
    - Policy is implicit in this case
- **Policy based**
  - The goal is to optimize the policy
- **Model based**
  - The goal is to model the environment and its characteristics and behaviors

Bellman Equation

- Equation often utilized in optimizations that use dynamic programming
  - Used in RL all the time
- Decomposes the value function into two parts:
  - The reward from the current state, current action
  - The discounted future value of the next state
- Allows to break up a dynamic optimization problem into a sequence of subproblems

- s = given state
- a = an action the agent will take
- s′ = state to which the system will move
- γ = discount factor
- R(s, a) = reward function for a given state
- V(s) = value of being in a particular state

$$V(s) = \max_a \left(R(s,a) + \gamma V\left(s'\right) \right)$$

$$v(s) = \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right]$$

https://towardsdatascience.com

Different ways of presenting Bellman Equation

Markov decision processes

- Markov decision process (MDP) “is a random process in which future is independent of the past, given the present” - Kyle Siegrist

$$(X_t|X_{t-1}, X_{t-2}, X_{t-3}, ...) = (X_t|X_{t-1})$$

Markov decision processes in RL (cont'd)

- MDP can be used when transitioning from one state to the next
  - The agent moves from one state to the next with some transition probability (different next states will have different transition probabilities)
  - We can write the probability for the next state as:

$$P[S_{t+1} \mid S_t] = P[S_{t+1} \mid S_1, \ldots\ldots, S_t]$$ https://towardsdatascience.com

  - And we can write transition probability to the next state as:

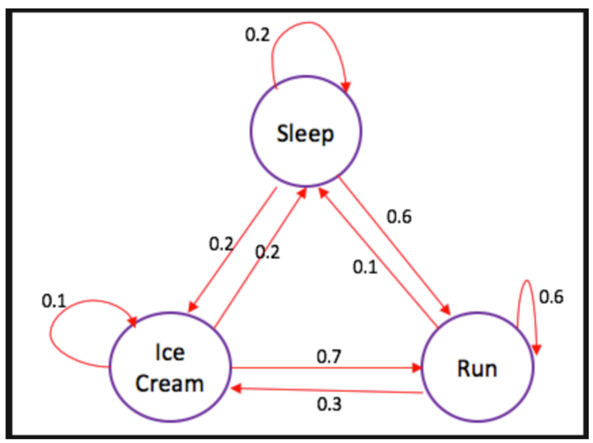
$$\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$$ https://towardsdatascience.com

  - And we can drive a state transition probability matrix: $P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ & \cdots & \cdots & \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{bmatrix}$

https://towardsdatascience.com

Markov decision processes example

- Some possible sequences from the following Markov decision process are:
  - Sleep - run - ice cream - run
  - Sleep - ice cream - ice cream - sleep



https://towardsdatascience.com

Markov reward process

- When combining an MDP with the reward hypothesis we get Markov Reward Process
  - There are reward values associated with each state
- We can think about the expected reward at each state as follows:
  - Reward value at state S is the expected reward value at the next state given the current state

$$R_s = E[R_{t+1} \mid S_t]$$

https://towardsdatascience.com

Policy vs. value functions in the context of Markov reward processes

- Policy function is the probability distribution function over all possible actions $(a \in A)$
- The value function of state S, given a policy, is the expected return (cumulative value of rewards) from future states, starting at the current state S, if the agent follows this policy

Different types of policies

- Deterministic policy
  - $a_{t+1} = \pi(s_t)$
- Stochastic policy (when we utilize Markov decision processes)
  - $\pi(a_{t+1} | s_t) = P(a_{t+1} | s_t)$

Modified Bellman’s equation

- To introduce randomness into the original formulation of Bellman’s equation we can modify it to perform an action with some amount of stochasticity

$$V(s) = \max_a(R(s,a) + \gamma \sum_{s'} P(s,a,s')V(s'))$$

Probability of moving from state *s* to state *s’* by performing action *a*

Advantages of RL

- Good at solving complex problems in a way similar to how humans learn
  - Has shown a track record of producing high-accuracy solutions in many applications
- Can utilize neural networks and therefore take advantage of their ability to solve complex non-linear problems
- Constant learning and improving
  - A mistake that occurs early on is unlikely to occur later on
- Versatility of the models possible to build with RL
- RL requires exploration so

Disadvantages of RL

- Can take a long time to learn a good solution
  - Can be resource-heavy
- Some problems require a lot of training data in order to learn an accurate solution
- RL is not appropriate for simple problems as it is geared to produce complex solutions
- High maintenance cost for RL models
- Possibility of excessive training

RL algorithms: Q-learning

- Q-learning is a value-based RL algorithm
- Model-free
  - Uses dynamic programming
  - Builds Q-table to store accumulated rewards
- Policy-free
  - Algorithm learns from examining all possible actions and picking the one with the highest expected reward
  - Learns from taking random actions sometimes (actions are not based on a policy)
- “Q” stands for “quality”
  - Quality of the action that maximizes the reward

RL algorithms: Q-learning (cont'd)

- *Q(s, a)* is a value function

- Steps:
  1. Initialize Q*(s, a)* to an arbitrary initial value
  2. Repeat until done
     i. Choose action *a* based on the state *s* by estimating Q*(s, .)*
     ii. Perform action *a* and observe the outcome state s_{new} and reward *r*
     iii. Update Q as follows:

learning rate

$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha(update)$$

value for the previous observation in the current state

$$R(s,a) + \gamma \max_{a'} Q(s',a') - Q_{t-1}(s,a)$$

maximize over all possible actions

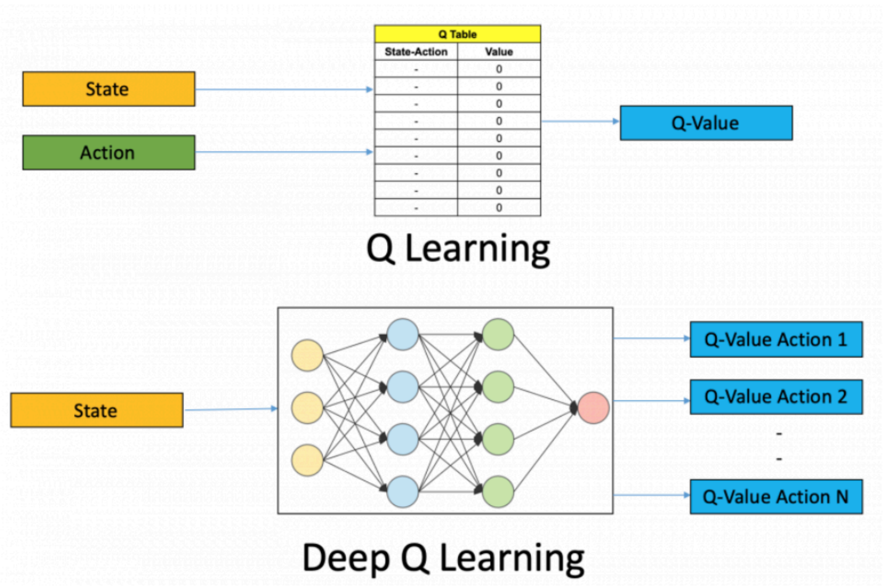
Recognize this equation? This is essentially Bellman's equation.

RL algorithms: Q-learning (cont’d)

- Q-table
  - A table for computing expected future reward for each possible action
  - The action with the highest expected reward is selected after constructing the table

Deep Q-networks

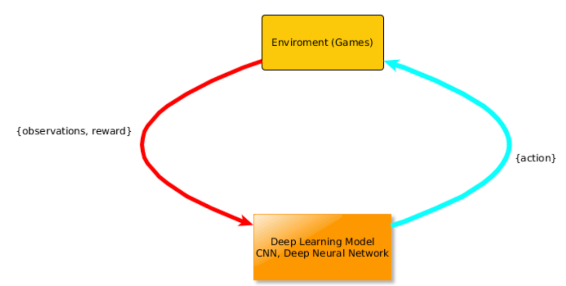
- In deep RL we use ANNs to approximate value function



analyticsvidhya.com

Policy gradient

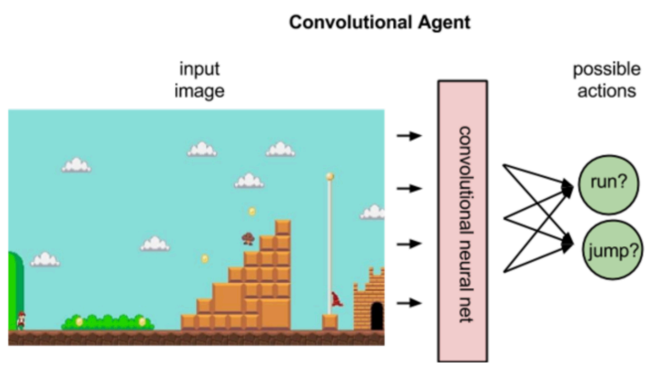
- Objective: optimize the policy to maximize the expected reward when using that policy
  - This means to optimize the parameters of the policy
  - Remember the definition of the policy and the value function in the context of MDP?
    - Policy is the probability distribution function over all possible actions
- Maximize the agent's performance over time
- There are number of policy gradient algorithms



https://leonardoaraujosantos.gitbook.io

Policy gradient (cont'd)

- The deep neural network takes observations as input and outputs the action
- Training this neural network learns the policy



https://leonardoaraujosantos.gitbook.io

Nice introduction to policy gradient

- https://leonardoaraujosantos.gitbook.io/artificial-inteligence/artificial_intelligence/reinforcement_learning/deep_reinforcement_learning

Episode

- In reinforcement learning an episode is a sequence of states, actions, and rewards between the starting state and a terminal state
- Normally the reward is observed only at the terminal state

Reward shaping

- Often the reward is not observed until a sequence of actions have taken place
  - A reward might only be observed at the end of the episode
  - Becomes difficult to separate which actions were “good” and which ones lead to a low reward/penalty
- Reward shaping refers to customizing the reward function to guide the agent to learn desired policy
- This makes the problem easier to learn
- Has been shown to work well for some problems

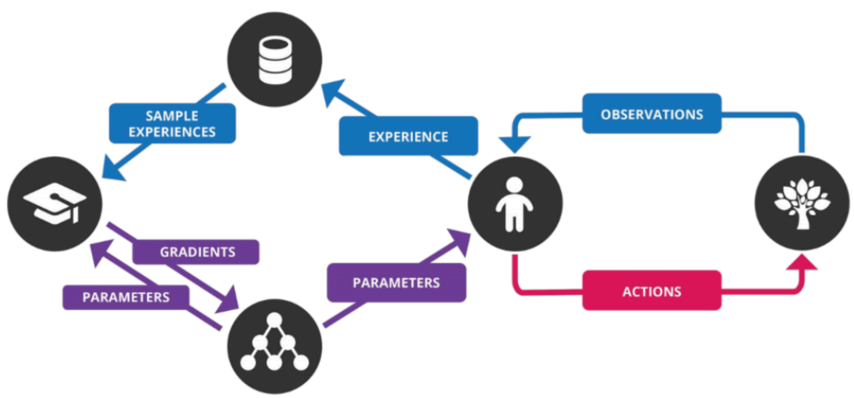
OpenAI gym toolkit

- Reinforcement learning toolkit
- https://gym.openai.com/docs/

- pip install gym

Let’s look at some code examples

- Q-learning example with taxi cab
  - *RL.Q-learning.taxi_game.ipynb*



analyticsvidhya.com