Reinforcement Learning: Overview

Sargur N. Srihari
srihari@cedar.buffalo.edu

Topics in Reinforcement Learning

1. RL as a topic in Machine Learning
2. Tasks performed by reinforcement learning
3. Policies with exploration and exploitation
4. RL connected to a deep neural net

Task of Reinforcement learning

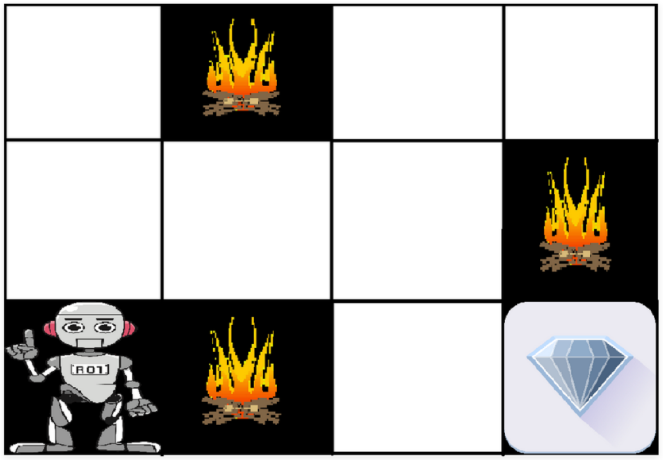
- Autonomous agent must learn to perform a task by trial and error without any guidance from the human operator
- Reinforcement learning is the problem of getting an agent to act in the world so as to maximize its rewards

Analogy of teaching a dog

- Consider teaching a dog a new trick:
  - You cannot tell it what to do, but you can reward/punish it if it does the right/wrong thing
  - It has to figure out what it did that made it get the reward/punishment, which is known as the credit assignment problem

Example of agent and environment

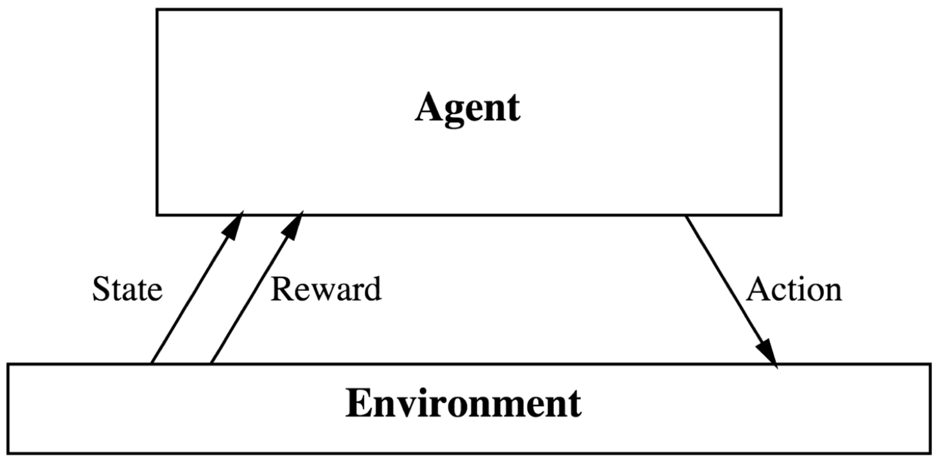
- Goal of agent: get reward of diamond and avoid the hurdles (fire)
- Robot learns by trying all
- possible paths and choosing path
- which gives reward with the least
- hurdles



- Each right step will give robot a reward and each wrong step will subtract the reward
- Total reward is calculated when it reaches the final reward that is the diamond

Reinforcement Learning Terminology

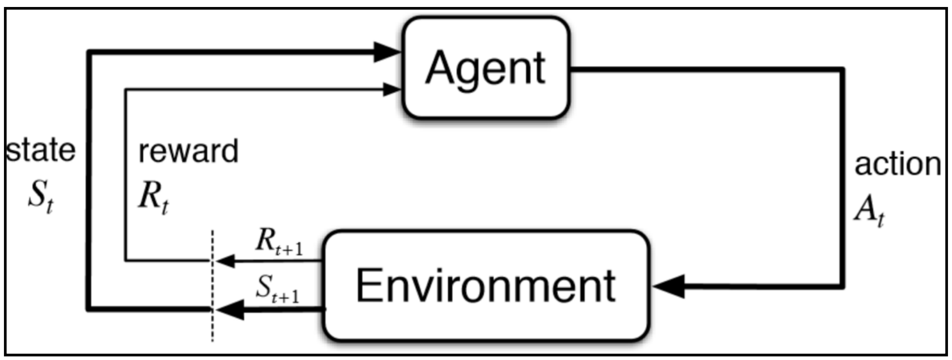
- *Agent* (algorithm) interacts with its environment
- A feedback loop between agent (a system) and its experience (in the environment)



  – A mobile robot has *actions* (move forward, turn).
    - Its task is to learn a control strategy or *policy* for choosing actions that achieve its goals
      – E.g., goal of docking onto battery charger when battery is low

The Learning Task

- Agent exists in environment with set of states S
    - It can perform any of a set of actions A
        - Performing action A_t in state S_t receives reward R_t
- Agent's task is to learn control policy $\pi : S \rightarrow A$
    - That maximizes expected sum of rewards
        - with future rewards discounted exponentially



$$S_0 \xrightarrow[R_0]{A_0} S_1 \xrightarrow[R_1]{A_1} S_2 \xrightarrow[R_2]{A_2} \ldots\ldots$$

Goal: Learn to choose actions that maximize

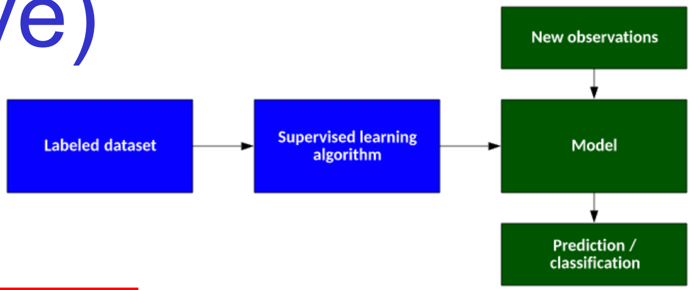
$$R_0 + \gamma R_1 + \gamma^2 R_2 + \ldots.., \quad \text{where } 0 \leq \gamma < 1$$

Summary of Terminology

- Action (A): possible moves that agent can take
- State (S): Current situation returned by environment
- Reward (R): Immediate return sent back from the environment to evaluate the last action
- Policy (π): Strategy that agent employs to determine next action based on current state
- Value (V): Expected long-term return with discount, as opposed to short-term reward

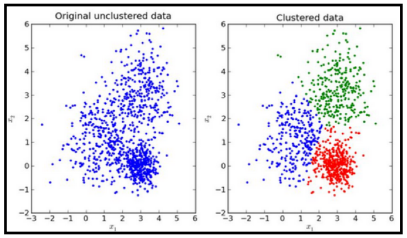
Three Types of Machine Learning Tasks

1. Supervised Learning (Predictive)



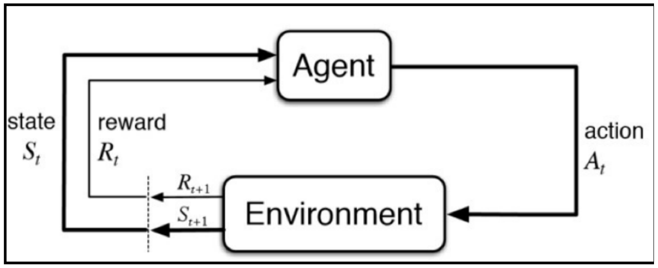
- Learn $y(\boldsymbol{x})$ given $D = \{(\boldsymbol{x}_n, t_n)\}$
  - E.g., MNIST classification
  - Minimize log loss: $\arg\min_{\boldsymbol{w}} E(\boldsymbol{w}) = -\sum_{n=1}^{N} \{ t_n \ln y_n + (1-t_n)\ln(1-y_n) \}$ where y_n denotes $y(\boldsymbol{x}_n, \boldsymbol{w})$

2. Unsupervised Learning (Descriptive)



- Learn distributions from inputs $D = \{\boldsymbol{x}_i\}$
  - E.g., Determine k clusters
  - Maximize likelihood with latent variables $\boldsymbol{z}$: $\arg\max_{\theta} \ln p(X \mid \theta) = \ln \left\{ \sum_{Z} p(X, Z \mid \theta) \right\}$

3. Reinforcement Learning



- How to act given reward signals
  - E.g., robot learns to walk
  - Optimize policy $s \rightarrow a$: $\pi^*(s) = \arg\max_{\pi} [r(s,a) + \gamma V^*(\delta(s,a))]$

Defining an environment

- In Gym, an openAI toolkit for RL, we define:
  1. action_space: possible actions of agent
  2. observation_space: possible states based on action
  3. state: current state of the environment
- We also define the following methods:
  1. _init_: initialise environment with default values
  2. step: accepts an Action, calculates and returns {new state, reward and done_state} after taking this action
  3. reset: clear all the variables in the environment and reset it to its initial state
  4. render: provide output for better debugging or showcasing

https://medium.com/grandcentrix/take-the-iot-world-by-reinforcement-learning-4bd458d639b0

Learning a Control Policy

- Target function to be learned is a control policy, π: $S \rightarrow A$, that outputs action a given state $s \in S$
- Determine as to what action to take in a particular situation, so as to maximize cumulative reward
- Problem is one of learning to control a sequential process
  - In manufacturing optimization
    - What sequence of manufacturing actions must be chosen
      - Reward to be maximized is value of goods produced minus cost

How RL differs from other ML

1. Delayed Reward
   - In other types of ML, training example is $<s, a=\pi(s)>$
   - In RL, trainer provides immediate reward for a
     - Which actions are to be credited for outcome
2. Exploration
   - Agent influences distribution: by action sequence chosen. So which experimentation produces best learning?
     - Exploration of unknown states and actions?
     - Or exploitation of states and actions already learned
3. Partially observable states
   - Entire state may not be observable
     - Need to combine previous observations with current sensor data when choosing actions

Exploration and Exploitation

- Reinforcement learning requires choosing between exploration and exploitation
- Exploitation
  - Refers to taking actions that come from the current best version of the learned policy
    - Actions that we know will achieve a high reward
- Exploration
  - Refers to taking actions specifically to obtain more training data

Policy with exploration/exploitation

- Given context $\boldsymbol{x}$, action a gives us a reward of 1
- We do not know if it is the best possible reward
- We may want to exploit our current policy and continue taking action a to be sure of obtaining reward of 1
- We may also want to explore by trying action a'
- We do not know what will happen if we try a'
- We hope to get a reward of 2, but we run risk of getting a reward of 0
- Either way we get some knowledge

Implementation of Exploration

- Implemented in many ways
  - Occasionally taking random actions intended to cover the entire range of possible actions
  - Model-based approaches that compute a choice of action based on its expected reward and the model's uncertainty about that reward

Preference for exploration or exploitation

- Factors for preference
- If agent has only a short amount of time to accrue reward then we prefer exploitation
- If agent has a long time to accrue reward we begin with more exploraion
  - So that future actions can be planned more effectively with knowledge
  - As time progresses we move towards more exploitation

Intuitions from other disciplines

- RL has a very close relationship with psychology, biology and neuroscience.
- What a RL agent does is just trial-and-error:
  - it learns how good or bad its actions are based on the rewards it receives from the environment
  - This how a human learns
- Besides, exploration/exploitation and credit assignment, attempts to model the environment are also something we face in our everyday life.

Applications of Reinforcement Learning

- Use similar method to train computers for
  - Game playing (backgammon, chess, GO)
  - Scheduling jobs
  - Robots (in a maze, controlling robot limbs)

https://www.youtube.com/watch?v=gn4nRCC9TwQ

  - Multiple agents, Partial observability

- RL system based on deep learning
  - Play Atari video games (Deep Mind)
  - Robotics
  - Reaching human level performance on many tasks

Data Sets

- Unlike supervised and unsupervised learning, reinforcement learning does not just experience a fixed data set
- Reinforcement learning algorithms interact with an environment
  - Q-learning generates data exclusively from experience, without incorporation of the prior knowledge.
  - If we put all our history data into a table with *state, action, reward, next state* and then sample from it, it should be possible to train our agent that way, without the dataset

Application of RL: Resource Management

1. Resources management in computer clusters
   - To allocate/schedule resources to waiting jobs, with objective to minimize average job slowdown
   - State space formulated as current resources allocation and resources profile of jobs.
   - Action space, they used a trick to allow the agent to choose more than one action at each time step
   - Reward was the sum of (-1/duration of the job) over all the jobs in the system

https://people.csail.mit.edu/alizadeh/papers/deeprm-hotnets16.pdf

Application of RL: Robotics

- Robot learns policies to map raw video images to robot's actions.
- The RGB images were fed to a CNN and outputs were the motor torques
- The RL component was the guided policy search to generate training data that came from its own state distribution

1. https://www.ias.informatik.tu-darmstadt.de/uploads/Publications/Kober_IJRR_2013.pdf

Multi-tasking in RL

- Robot learning may involve learning several related tasks
- Mobile robot may need to:
  - Dock on its battery charger
  - Navigate through narrow corridors
  - How to pick up output from a laser printer

RL connected to deep neural net

- Task: Learning to navigate in complex environments without prior knowledge.
- RL agent infers from complex environments by punishment-reward system. It can model decision making process.
- Example Applications:
  - AlphaGo Zero beat the world champion (December 2017)
  - OpenAI bot won in Dota2 world championship (Aug 2018)

Deep Reinforcement Learning for Atari

Paper: "Playing Atari with Deep Reinforcement Learning" by V. Mnih, et. al. NIPS 2013, Atari Breakout

Dataset: Q-learning generates data exclusively from experience, without incorporation of the prior knowledge. If we put all our history data into a table with *state, action, reward, next state* and then sample from it, it should be possible to train our agent that way, without the dataset.

Backend: Python3, Keras, Tensorflow

Core libraries: OpenAI Gym, Keras - RL

Code: https://github.com/nathanmargaglio/DQN

Atari strategy

- Strategy: (1) estimate discounted sum of rewards of taking action a in state s - $Q(s, a)$ function, (2) choose the action with the maximum Q-value in any given state

$$\boxed{Q_{i+1}(s,a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q_i(s',a') \middle| s,a\right]}$$

r - reward; γ - discounting factor.

An agent learns by getting positive or negative rewards

- Loss: Huber Loss (modified MSE/MAE)

$$\boxed{Huber(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \leq 1, \\ (|a| - \frac{1}{2}), & \text{otherwise.} \end{cases}}$$

- Evaluation metrics: Maximizing the cumulative reward. Comparing to other implementations and human players.
- Stopping criterion: Once agent cannot increase total reward

Reinforcement Learning: ATARI

Environment: BreakoutDeterministic-v4

Backend: Keras, Python3

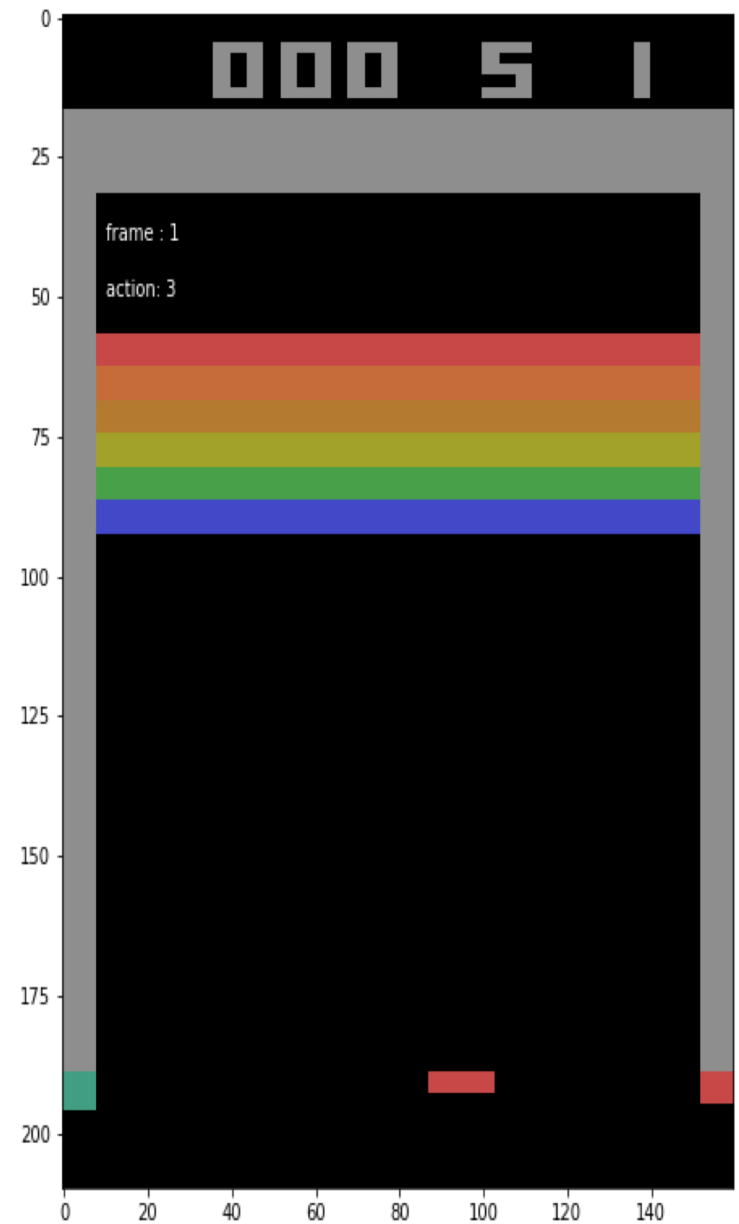
Libraries: OpenAI Gym, Keras-RL

Reward: max score - 208 (the benchmark in the paper 225)

Preprocessing: original image was downsampled from 210×160 pixel images to 105×80 and converted from RGB to gray-scale to decrease the computation

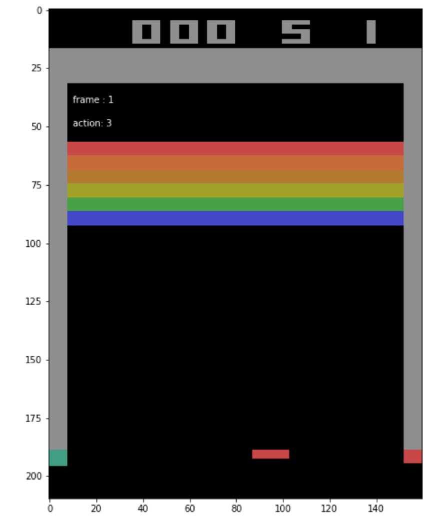
Training time: 15 hours including simulation time on a GTX 650 with 1 GB of RAM

Notations:
Frame - a snapshot of the environment state at every point
Action (a) - a set of actions, that agent can take {0, 1, 2, 3}
Upper left corner - score (our evaluation metric)
Upper middle - number of "lives" for each game (initially 5)
Upper right corner - might be version



RL: Learning to play ATARI

- Action(a)={left,right}
- Observation(s)=[image frame]
- Reward(r)= -100 if lose, -1 if win
- Policy(π)=$P_\pi(a|s)$
  - 10,000 states, 2 actions
- $Q(s,a)$= value (action,state)



$$Q_{i+1}(s,a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q_i(s',a')|s,a]$$

- Loss = γ+E[$\max_{a'} Q(s',a')$-$Q_i(s',a')$)