An Introduction to Reinforcement Learning

by John Sigmon

What is Reinforcement Learning (RL) ?

Supervised Learning	A classification or regression problem where we have labeled data we can use to learn the model.

What is Reinforcement Learning (RL) ?

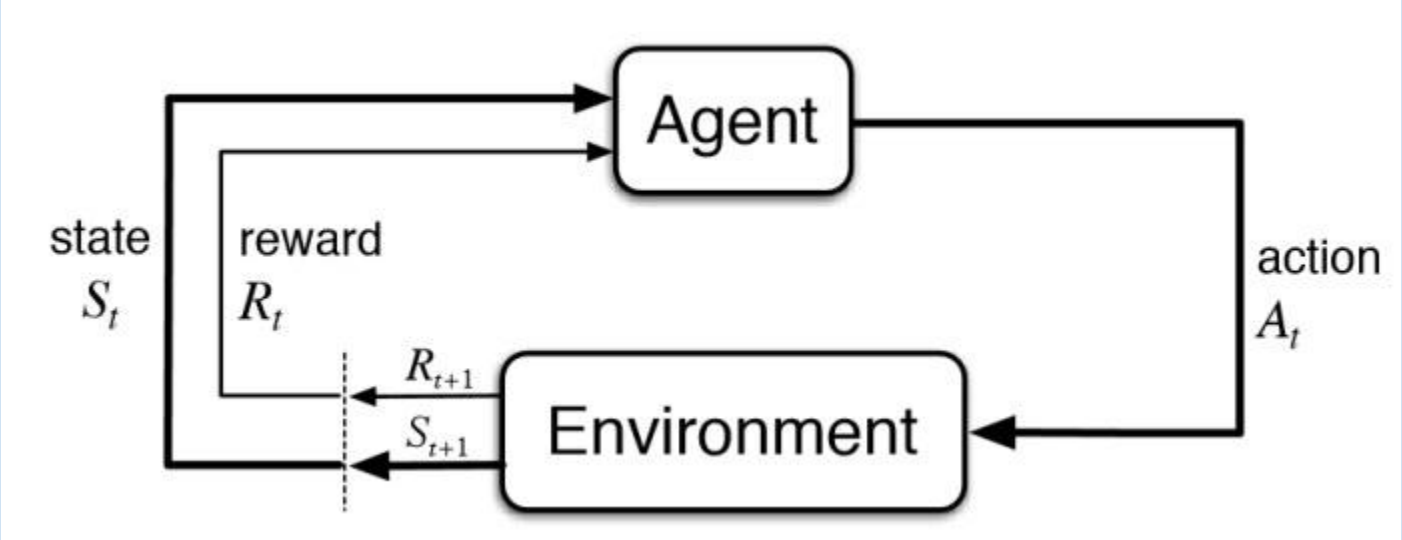
Supervised Learning	A classification or regression problem where we have labeled data we can use to learn the model.
Unsupervised Learning	A problem where there are no labels and the goal is to find hidden structure in the data.

What is Reinforcement Learning (RL) ?

Supervised Learning	A classification or regression problem where we have labeled data we can use to learn the model.
Unsupervised Learning	A problem where there are no labels and the goal is to find hidden structure in the data.
Reinforcement Learning	A time dependent problem where the model must learn based on rewards.

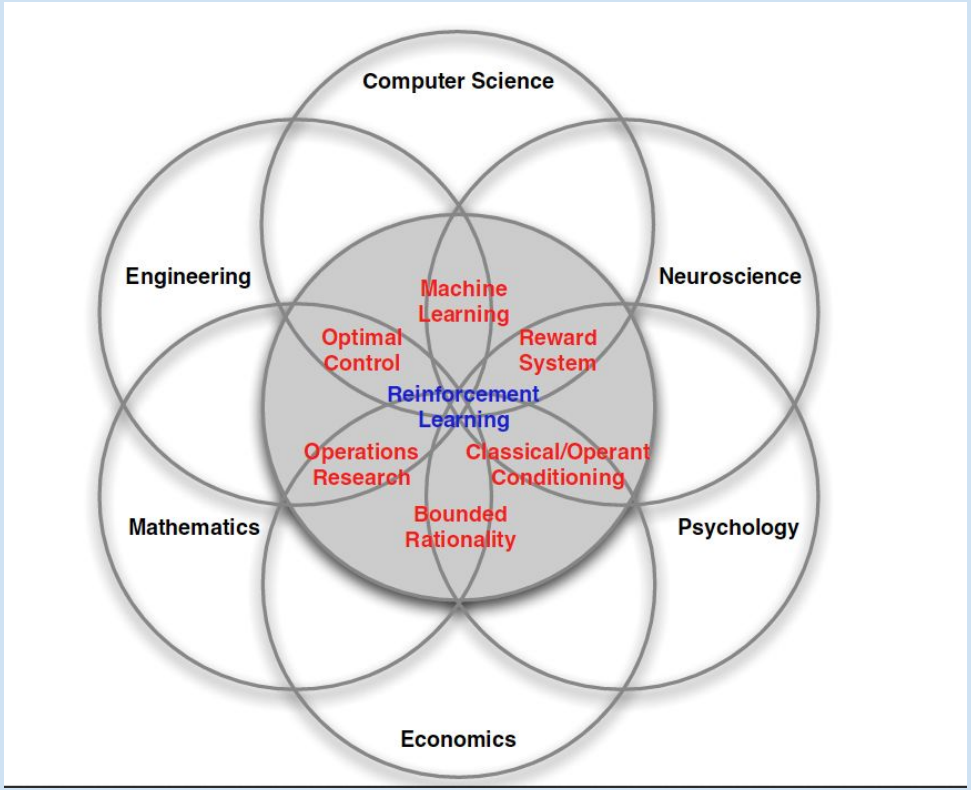
What is Reinforcement Learning (RL) ?

Typically means solving any problem that fits into this diagram



What is Reinforcement Learning (RL) ?

How people (in RL) see the field



Where is RL used?

Self driving cars

- State is recorded camera data and other sensor input
- Actions are acceleration, deceleration, turning, and others
- Rewards could be obeying traffic laws

Where is RL used?

Playing video games

- State is the pixels on the computer screen
- Actions are defined by the game (e.g. moving up and down in Pong)
- Rewards could be the game score

Where is RL used?

Grasping objects in robotics

- State is recorded camera data
- Actions are movement of the robot (usually a robotic arm in object grasping)
- Rewards could be positive for successful grasping, and penalty for no success

Where is RL used?

Simulated robot soccer

- State is the set of robot joint angles along with location of ball and other players
- Actions are defined by robot joint angles directly to form complex movement
- Rewards are defined differently for different skills, e.g. kick distance when learning a long kick

Where is RL used?

Chatbots (work in progress)

- State is the state of the conversation encoded somehow
- Actions are chatbot responses to human input
- Rewards could be given when correct information is provided to the human (depends on what type of chatbot)

The Details

First, difference in notation

RL Terminology	RL Notation	Control Terminology	Control Notation
Action	a	Control Signal	u
State	s (or o)	State / Environment	x
Reward	r	Cost / Payoff	J or c

The Details

The Markov Decision Process:

$$< S, A, P_a, R_a, \gamma >$$

The Details

The Markov Decision Process:

$$< S, A, P_a, R_a, \gamma >$$

State Set of all possible states (usually infinite) $S = \{s_1, s_2, \ldots, s_n\}$

The Details

The Markov Decision Process:

$$< S, A, P_a, R_a, \gamma >$$

State Set of all possible states (usually infinite) $S = \{s_1, s_2, ..., s_n\}$

Action Set of all possible actions (usually infinite) $A = \{a_1, a_2, ..., a_m\}$

The Details

The Markov Decision Process:

$$< S, A, P_a, R_a, \gamma >$$

State	Set of all possible states (usually infinite)	$S = \{s_1, s_2, ..., s_n\}$
Action	Set of all possible actions (usually infinite)	$A = \{a_1, a_2, ..., a_m\}$
Dynamics	Probability of a state given previous state and action	$Pr(s_{t+1} \mid s_t, a_t)$

The Details

The Markov Decision Process:

$$< S, A, P_a, R_a, \gamma >$$

State	Set of all possible states (usually infinite)	$S = \{s_1, s_2, ..., s_n\}$
Action	Set of all possible actions (usually infinite)	$A = \{a_1, a_2, ..., a_m\}$
Dynamics	Probability of a state given previous state and action	$Pr(s_{t+1} \mid s_t, a_t)$
Reward	Reward for transitioning to a state, given a previous state, action pair	$r(s_t, a_t, s_{t+1}) : S \times A \times S \to \mathbb{R}$

The Details

The Markov Decision Process:

$$< S, A, P_a, R_a, \gamma >$$

State	Set of all possible states (usually infinite)	$S = \{s_1, s_2, ..., s_n\}$
Action	Set of all possible actions (usually infinite)	$A = \{a_1, a_2, ..., a_m\}$
Dynamics	Probability of a state given previous state and action	$Pr(s_{t+1} \mid s_t, a_t)$
Reward	Reward for transitioning to a state, given a previous state, action pair	$r(s_t, a_t, s_{t+1}) : S \times A \times S \to \mathbb{R}$
Discounting Factor	Discounting factor for future rewards	$\gamma \in [0, 1]$

The Details

Policy:

A function from the state space to the action space.

$$\pi(s) : S \to A$$

The Details

Policy:

A function from the state space to the action space.

$$\pi(s) : S \rightarrow A$$

Optimal Policy:

The policy that maximizes the sum of all future rewards.

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \cdot r_t(s_t, a_t, s_{t+1}) \,|\, \pi]$$

Finding an Optimal Policy

Value Iteration
Policy Iteration
Dynamic Programming
Policy Gradients
Temporal Difference Learning
Sarsa Learning
Q Learning
Monte Carlo Methods

How to Get Started

OpenAI

Founded December 2015

Released OpenAI Gym April 2016

https://gym.openai.com/

How to Get Started

OpenAI

Founded December 2015

Released OpenAI Gym April 2016

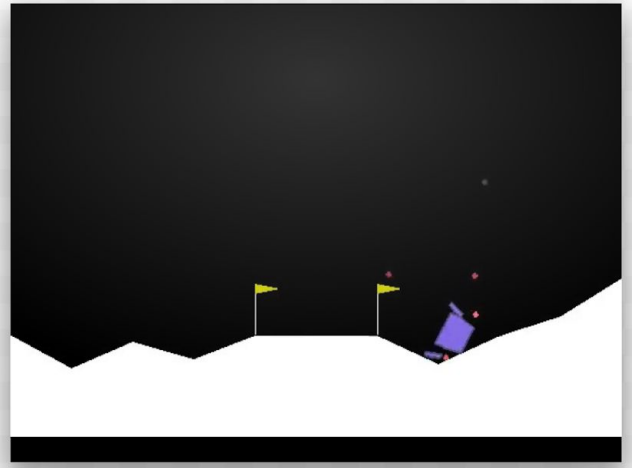
https://gym.openai.com/

Gym

Gym is a toolkit for developing and comparing reinforcement learning algorithms. It supports teaching agents everything from walking to playing games like Pong or Pinball.

View documentation ›

View on GitHub ›



RandomAgent on LunarLander-v2

Code Demo

```
1   import gym
2   env = gym.make('CartPole-v0')
3   env.reset()
4
5
6
7
8
9
10
11
12
13
14
```

Code Demo

```
1   import gym
2   env = gym.make('CartPole-v0')
3   env.reset()
4   for _ in range(20):
5       observation = env.reset()
6       for t in range(100):
7
8
9
10
11
12
13
14
```

Code Demo

```
import gym
env = gym.make('CartPole-v0')
env.reset()
for _ in range(20):
    observation = env.reset()
    for t in range(100):
        env.render()
        print(observation)
        action = env.action_space.sample()
        observation, reward, done, info = env.step(action)
```

Code Demo

```
import gym
env = gym.make('CartPole-v0')
env.reset()
for _ in range(20):
    observation = env.reset()
    for t in range(100):
        env.render()
        print(observation)
        action = env.action_space.sample()
        observation, reward, done, info = env.step(action)
        if done:
            print("Episode finished after {} timesteps".format(t+1))
            break

```

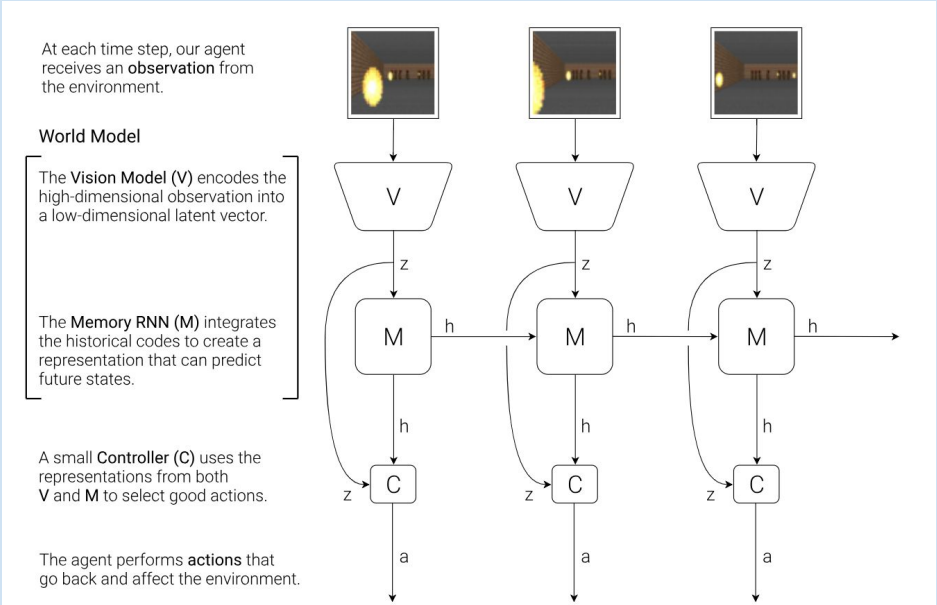
Code Demo

Current Research

World Models by David Ha, Jurgen Schmidhuber
https://arxiv.org/abs/1803.10122 and https://worldmodels.github.io/

Current Research

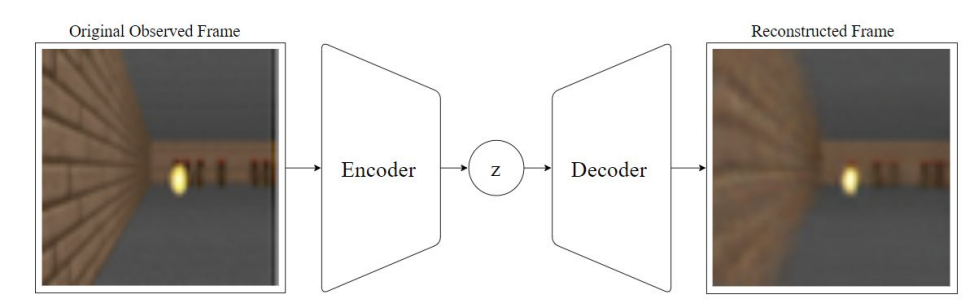
World Models by David Ha, Jurgen Schmidhuber
https://arxiv.org/abs/1803.10122 and https://worldmodels.github.io/



Our agent consists of three components that work closely together: Vision (V), Memory (M), and Controller (C).

World Models

Vision Model
A variational autoencoder

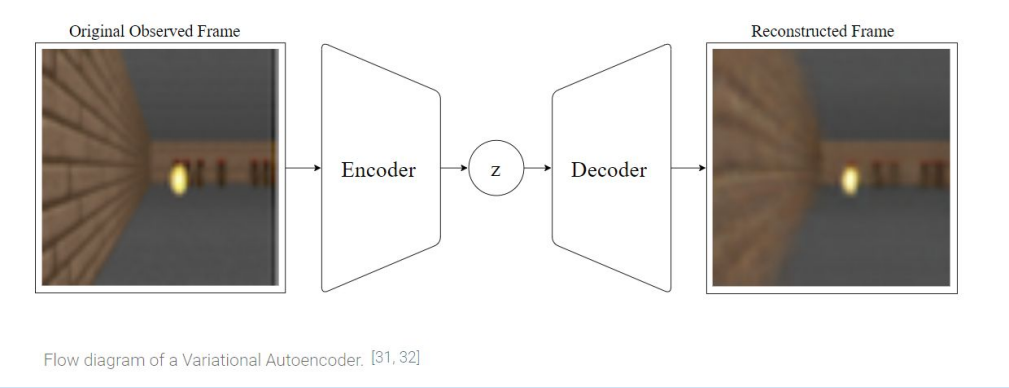


Flow diagram of a Variational Autoencoder. [31, 32]

https://worldmodels.github.io/

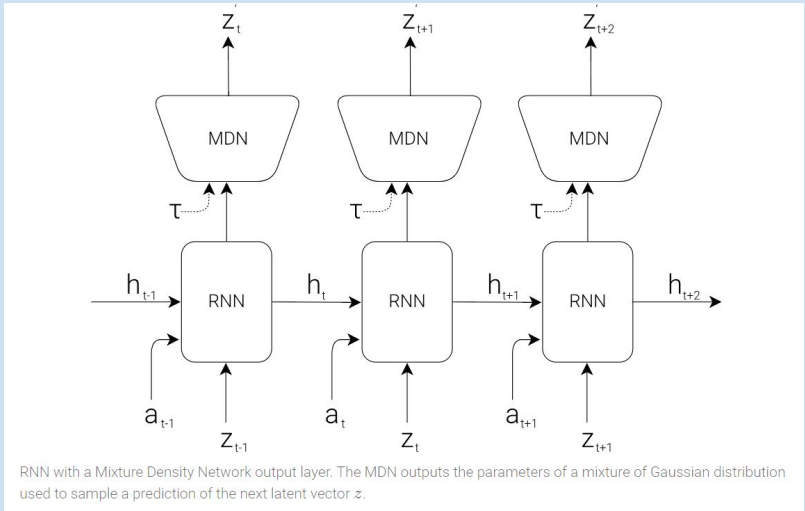
World Models

Vision Model
A variational autoencoder



Flow diagram of a Variational Autoencoder. [31, 32]

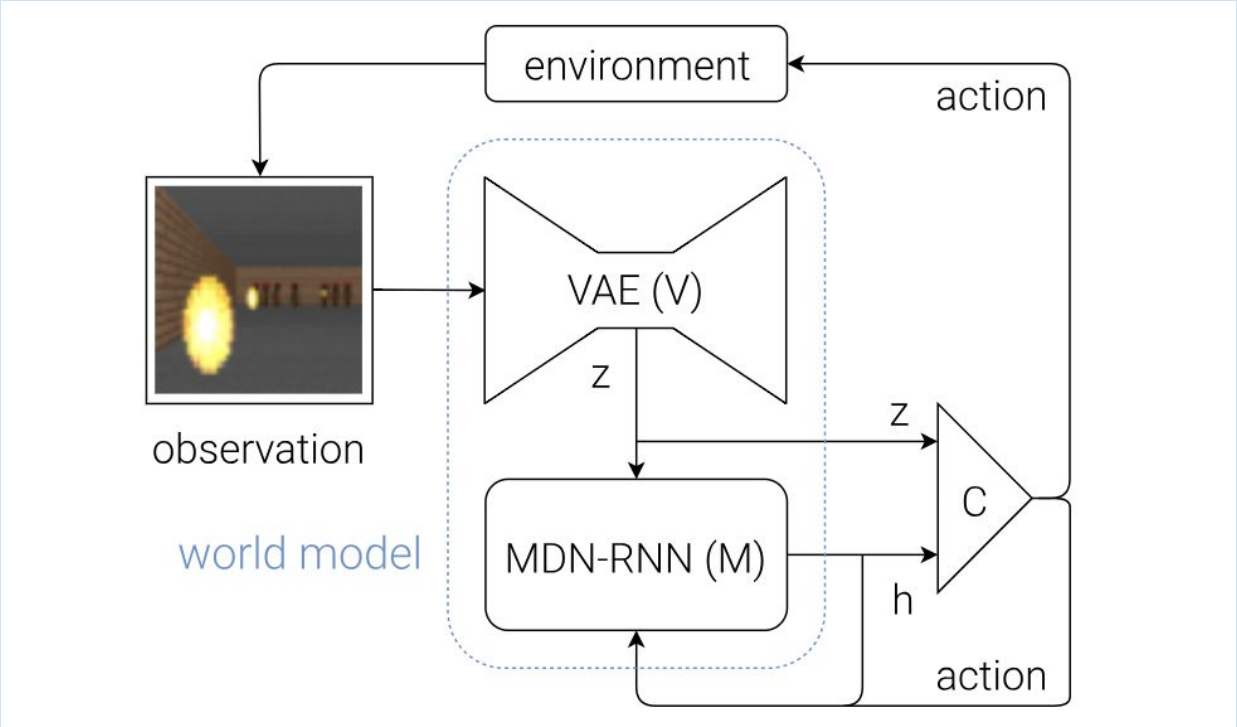
Memory Model
RNN feeding into a Mixture Density Network



RNN with a Mixture Density Network output layer. The MDN outputs the parameters of a mixture of Gaussian distribution used to sample a prediction of the next latent vector z.

https://worldmodels.github.io/

World Models



https://worldmodels.github.io/

Current Research

World Models by David Ha, Jurgen Schmidhuber
https://arxiv.org/abs/1803.10122 and https://worldmodels.github.io/

Deep Neuroevolution by Uber AI Labs
https://arxiv.org/abs/1712.06567 and https://eng.uber.com/deep-neuroevolution/

Learning Dexterous In-Hand Manipulation by Open AI
https://arxiv.org/abs/1808.00177 and https://blog.openai.com/learning-dexterity/

One Shot Imitation Learning by Open AI
https://arxiv.org/abs/1703.07326 and https://blog.openai.com/robots-that-learn/

Reinforcement Learning with Deep Energy Based Policies
https://arxiv.org/abs/1702.08165

Educational Resources

Reinforcement Learning by Richard Sutton, Andrew Barto
available online: http://incompleteideas.net/book/bookdraft2017nov5.pdf

Richard Sutton's home page
http://incompleteideas.net/

Peter Stone's course on Reinforcement Learning
http://www.cs.utexas.edu/~pstone/Courses/394Rfall16/resources/index.html

Levin, Schulman, and Finn's course on Deep Reinforcement Learning
http://rll.berkeley.edu/deeprlcoursesp17/

David Silver's course on Reinforcement Learning
http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

Some Other RL Environments

VizDoom for playing the 1993 video game Doom
http://vizdoom.cs.put.edu.pl/

TorchCraft a library to connect Torch and RTS games like StarCraft
https://github.com/TorchCraft/TorchCraft

DeepMind Lab a 3d environment based on Quake III
https://github.com/deepmind/lab

Dopamine a Google RL framework
https://ai.googleblog.com/2018/08/introducing-new-framework-for-flexible.html
https://github.com/google/dopamine

Thank You

Twitter
@JohnSigmon

LinkedIn
https://www.linkedin.com/in/john-sigmon/

Email
johnsigmon@gmail.com

GitHub Page for this talk
https://github.com/jsigee87/rl_talk