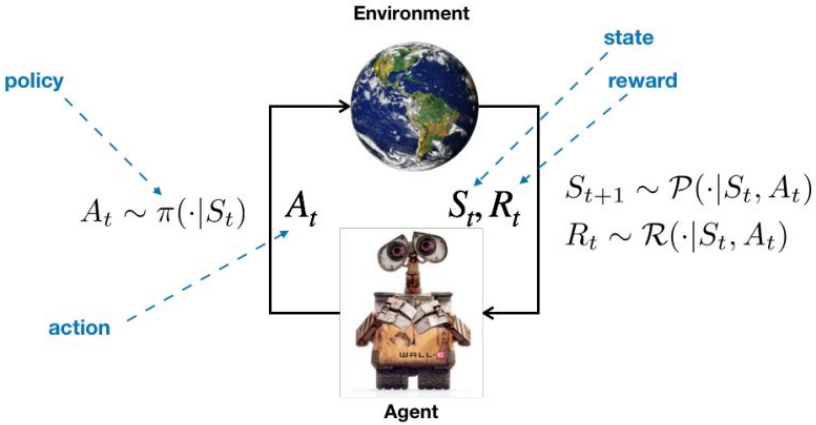
Model-Based RL

Xuchan (Jenny) Bao

CSC2515 Tutorial 10, Nov 20/21, 2019

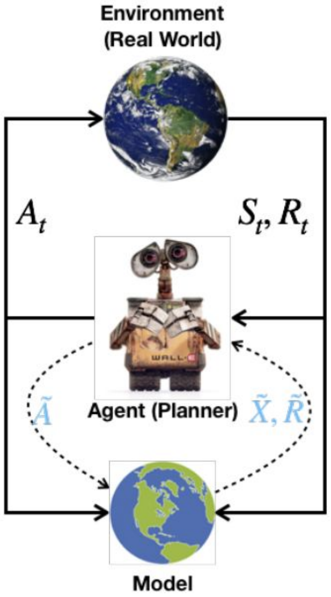
So far, we’ve seen model-free RL algorithms

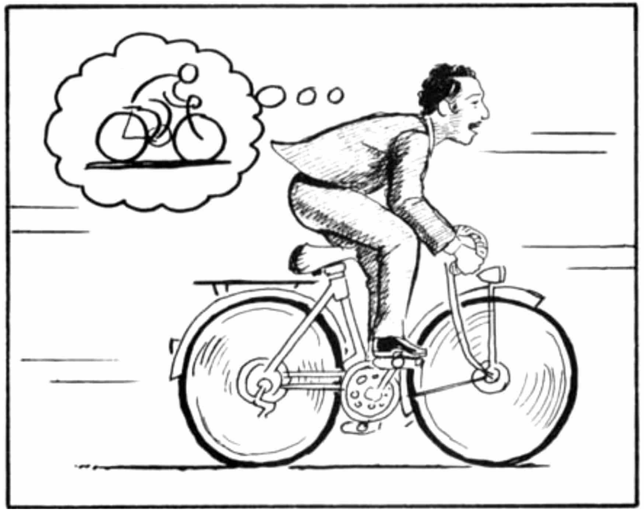
- Learn policy directly by interacting with the environment
- Agent does NOT attempt to model the transition P(s’ | s, a)


Environment
policy
state
reward
$A_t \sim \pi(\cdot|S_t)$
A_t
S_t, R_t
$S_{t+1} \sim \mathcal{P}(\cdot|S_t, A_t)$
$R_t \sim \mathcal{R}(\cdot|S_t, A_t)$
action
Agent

Model-Based RL

- Learn a representation (model) of the world, and use the model for policy learning / planning





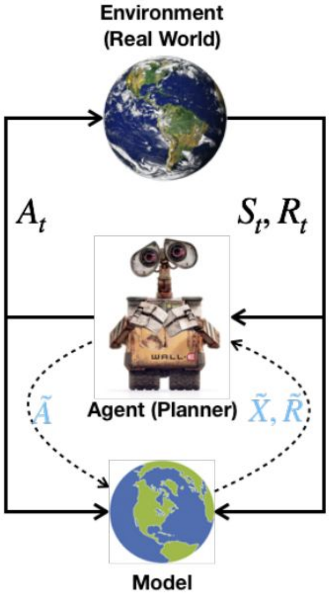
Images from:

Scott McCloud *Understanding Comics* https://en.wikipedia.org/wiki/Understanding_Comics https://memeengine.tumblr.com/post/28333277260/more-thoughts-from-understanding-comics-by-scott
CSC2515 Fall 2019 slides http://www.cs.toronto.edu/~rgrosse/courses/csc2515_2019/slides/lec10-slides.pdf

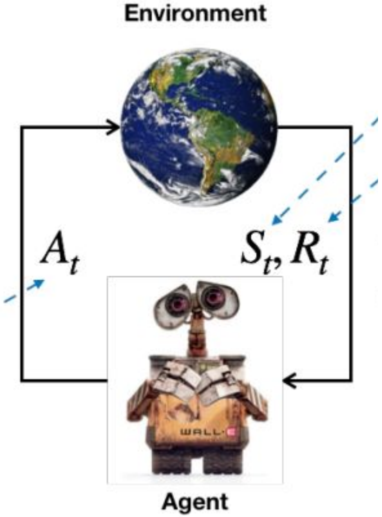
“*The image of the world around us, which we carry in our head, is just a model. Nobody in his head imagines all the world, government or country. He has only selected concepts, and relationships between them, and uses those to represent the real system.*”

– Jay Wright Forrester (Technology Review. 1971)

Model-Based vs. Model-Free RL



Model-Based RL	Model-Free RL
Learns to represent the world, and uses it to decide actions	Learns how to act directly via interaction with the real world
● Sample-efficient ● Model can be used for transfer learning	● Needs large amount of samples from the real world (expensive) ● Knowledge not easily transferable
● Often quite fragile, tricky to get to work ● Difficult to model high-dimensional environments with complex dynamics ● Policy can overfit to model ● Can be stuck at bad local minima → lower asymptotic performance	● More consistent & robust performance ● Higher asymptotic performance



Example: World Models (Ha et al, 2018)

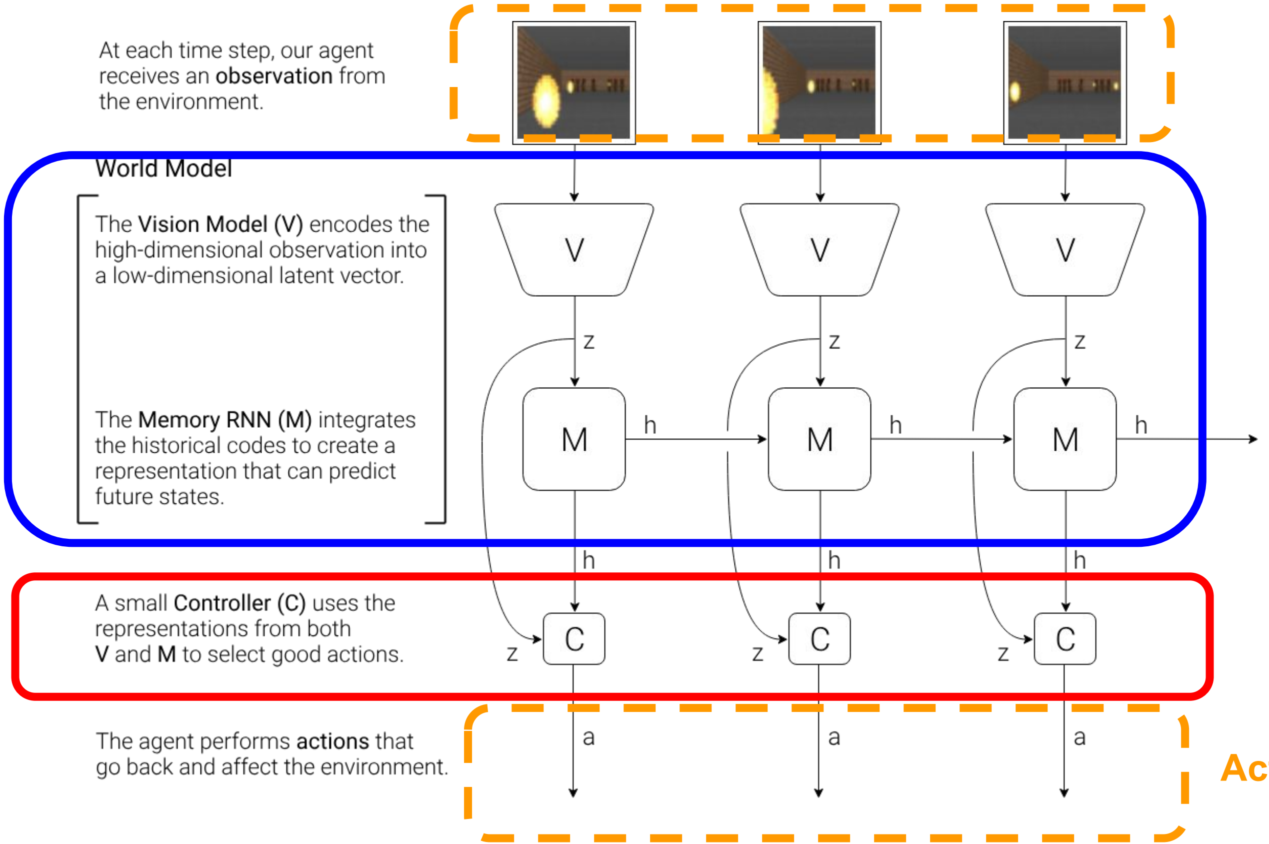
- Model-based RL from pixel input

World Model Architecture

Model
Vision model (V)
(VAE)
+
Memory model (M)
(RNN)

Controller
Simple Linear
Controller (C)

Sequential observations



At each time step, our agent receives an **observation** from the environment.

World Model

The **Vision Model (V)** encodes the high-dimensional observation into a low-dimensional latent vector.

The **Memory RNN (M)** integrates the historical codes to create a representation that can predict future states.

A small **Controller (C)** uses the representations from both **V** and **M** to select good actions.

The agent performs **actions** that go back and affect the environment.

Actions

Training World Models

All components are trained separately

- **Vision model (V):** a simple variational autoencoder, trained to reconstruct each observation
- **Memory model (M):** an RNN with mixture of Gaussian output, trained to model the transition in the encoded space
- **Controller (C):** a linear model, trained using Covariance-Matrix Adaptation Evolution Strategy (CMA-ES)

Design Decisions of World Model

- Use a very simple controller (just a linear model), so that most of the model’s complexity resides in the “world model” part (i.e. V and M models).
    - Can efficiently train the V and M models (backpropagation)
    - C is in general harder to train (e.g. RL), but C model is very simple by design
- Train all models separately
    - Easier to implement, and requires less hyperparameter tuning
    - Achieves satisfactory results
    - Limitation: the VAE model can encode irrelevant information (such as brick tile patterns on the walls)
    - Training V and M together to predict reward can help learn task-relevant information only, but might hurt generalization to other tasks

World Models: Demo

https://worldmodels.github.io/