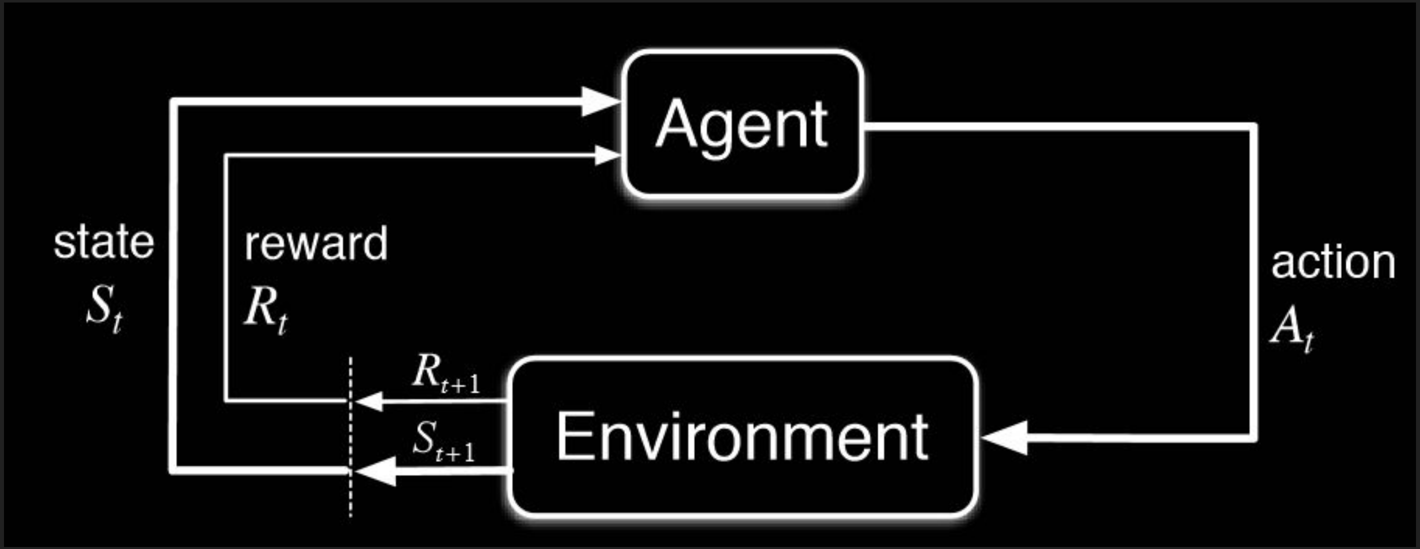
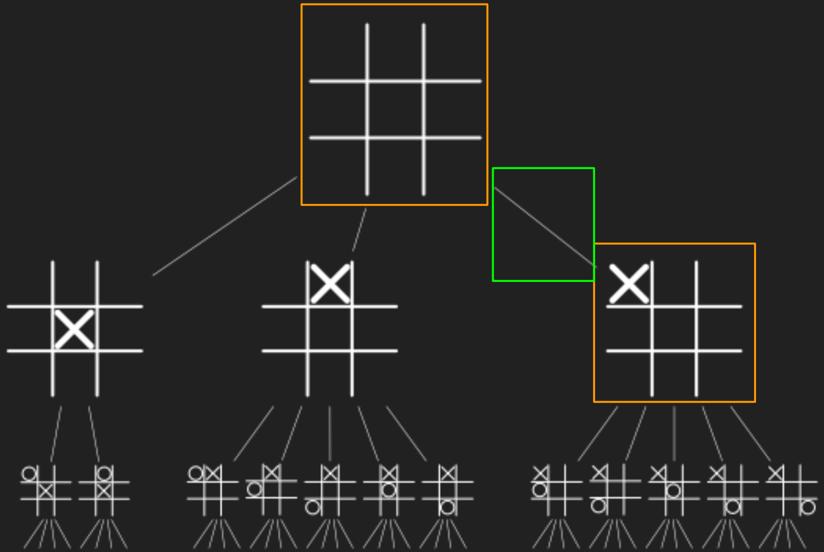
Model-based Reinforcement Learning

Arman Raayatsanati

Recap: Reinforcement Learning



Example: Tic Tac Toe



Agent: (AI) Player

State: Current board configuration

Action: Placing X (or O)

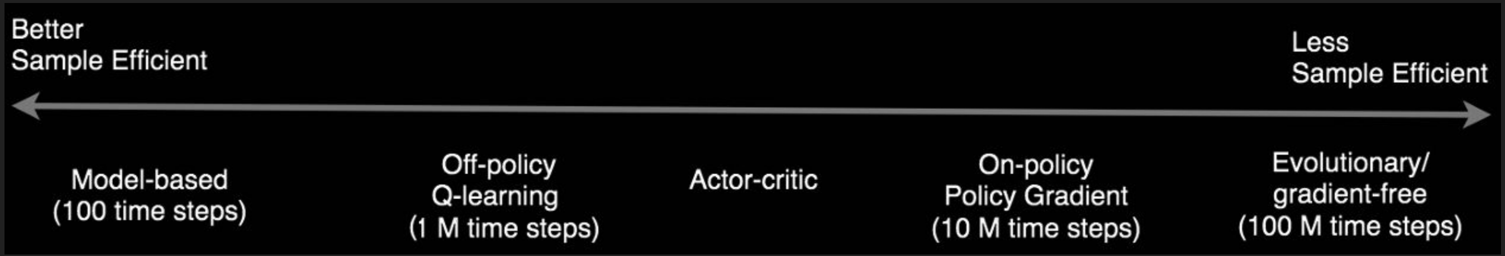
Reward: Points for winning / losing

Model-based Reinforcement Learning

Different dynamic states of an environment and how these states lead to a reward

Why?

- Better sampling efficiency



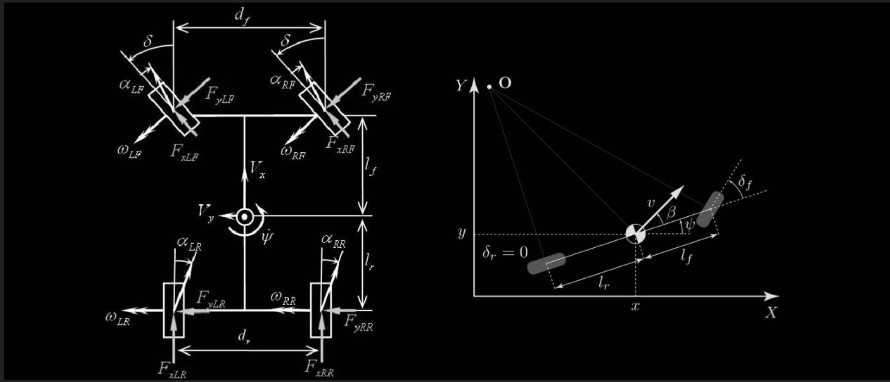
- Models can be reused for different tasks

Simplified Algorithm

1. Create dynamics model
2. Use model to improve policy and choose actions

Known Models

AlphaGo



Physical models

Known Models

$$s' = f(s, a)$$

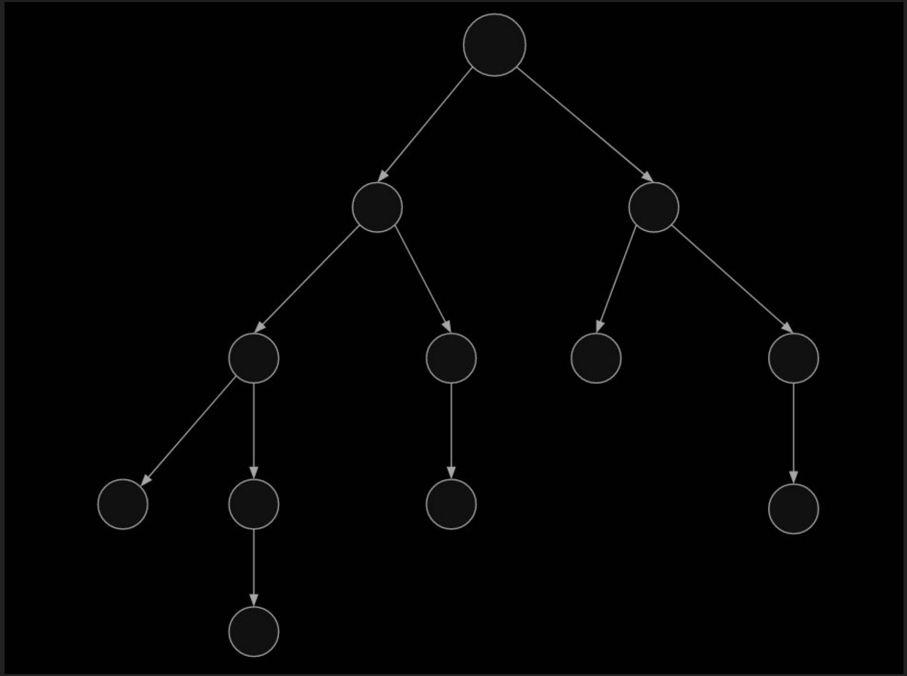
We have a mathematical equation that allows us to calculate and select the best next state using the current state and the current action.

This action is called planning.

Planning

For discrete actions: search algorithms that create decision trees

For continuous actions: trajectory optimization techniques such as model predictive control



Types of Models

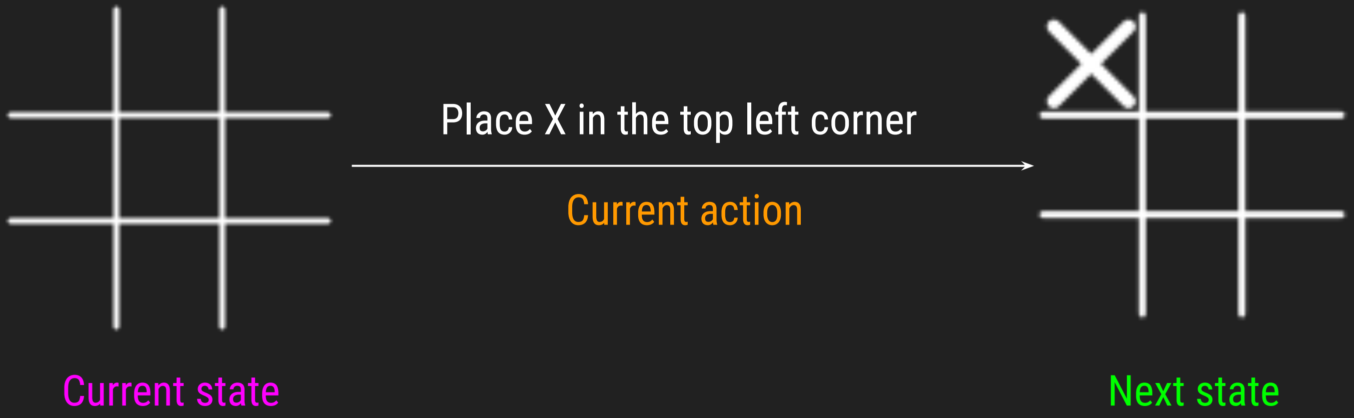
In general, we can think of different approaches to modeling the environment.

- Forward Model
- Backward/Reverse Model
- Inverse Model

So far, we have only looked at forward models!

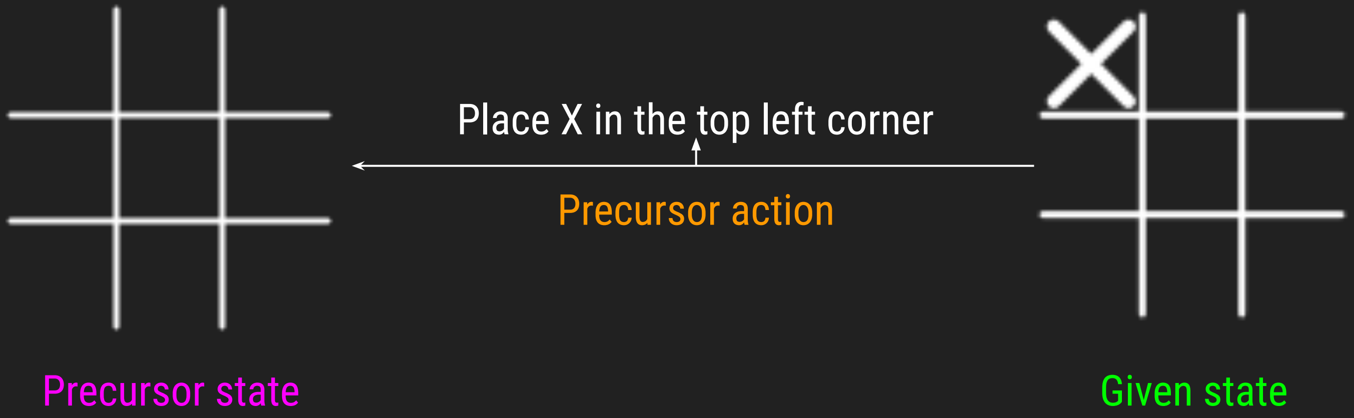
Forward Model

$$s' = f(s, a)$$



Backward Model

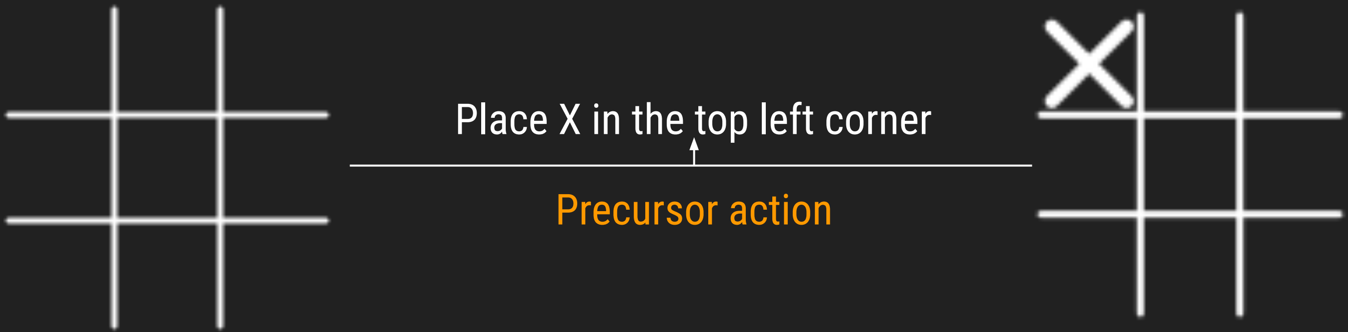
$$(s, a) = f(s')$$



Can you think of a case where the precursor state and the precursor action are not unique?

Inverse Model

$$a = f(s, s')$$

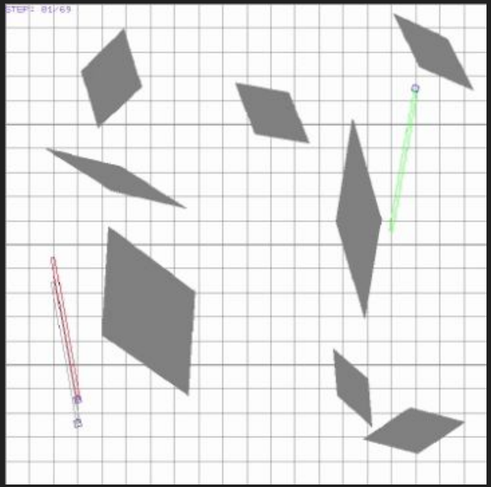


Precursor state

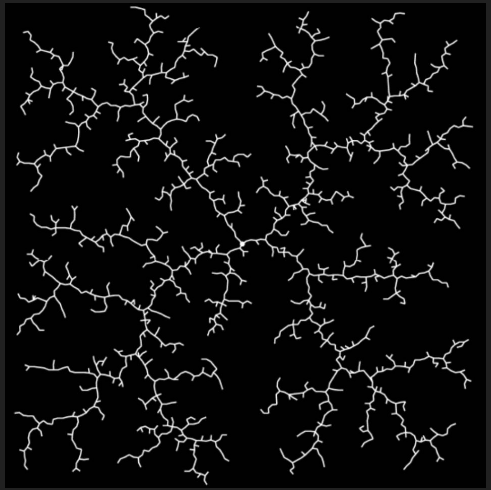
Given state

Working Examples

Despite the added challenges, backward and inverse models can be useful in practice.



Prioritized Sweeping (Backward Model)



Rapidly-Exploring Random Tree (Inverse Model)

Modified Algorithm

1. Create dynamics model (choose the appropriate type)

2. Use model to improve policy and choose actions

What about unknown models?

Here's where our machine learning models come in ☺️

Model-based Deep RL with a neural network

Learning the Model

Estimation of the model of the dynamics is a supervised learning problem.

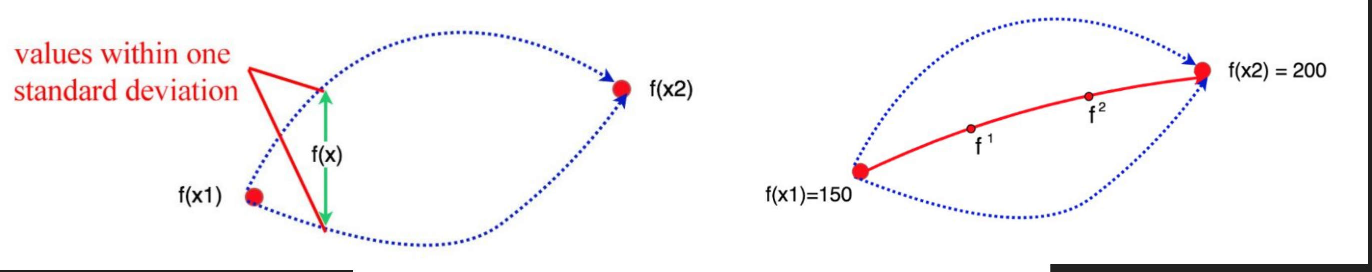
$$p(s'|s, a)$$

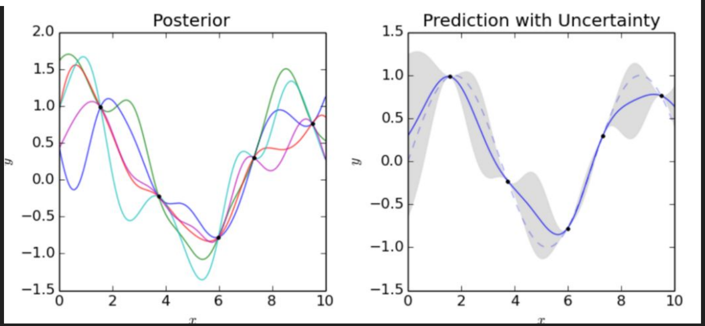
Maximize the likelihood of the next state given the current state and the current action.

However, we now also need data that we generally generate from a base policy.

Learning the Model

Just like with any supervised learning task, we can use a deterministic or a probabilistic model.

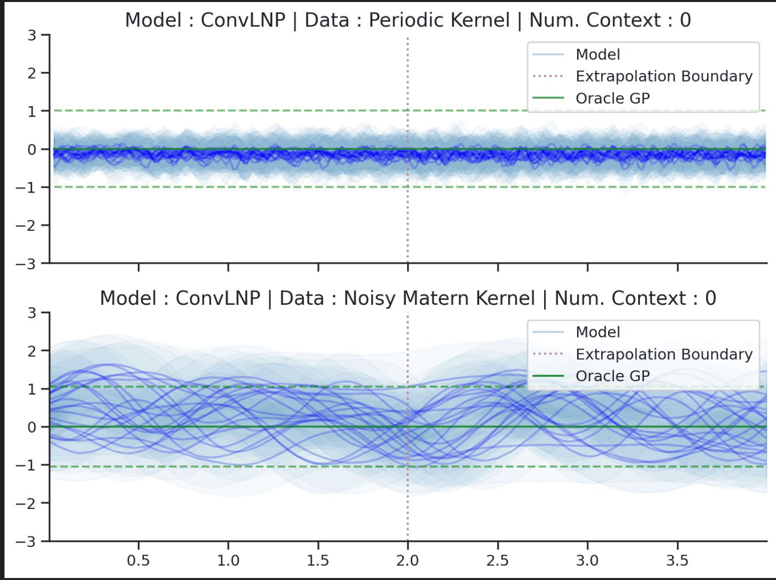


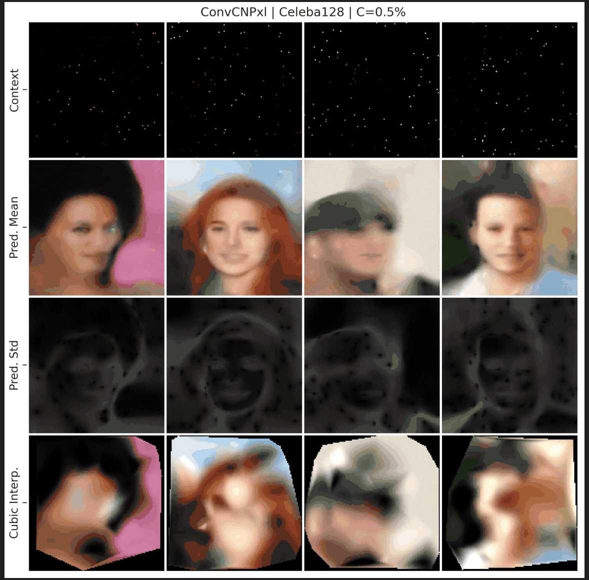


Gaussian Processes

Learning the Model

We might even be able to combine the two approaches using neural processes!





Modified Algorithm

1. Collect data under current (base) policy
2. Create dynamics model (choose the appropriate type)
3. Learn from collected data
4. Use model to improve policy and choose actions

Example: World Models

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

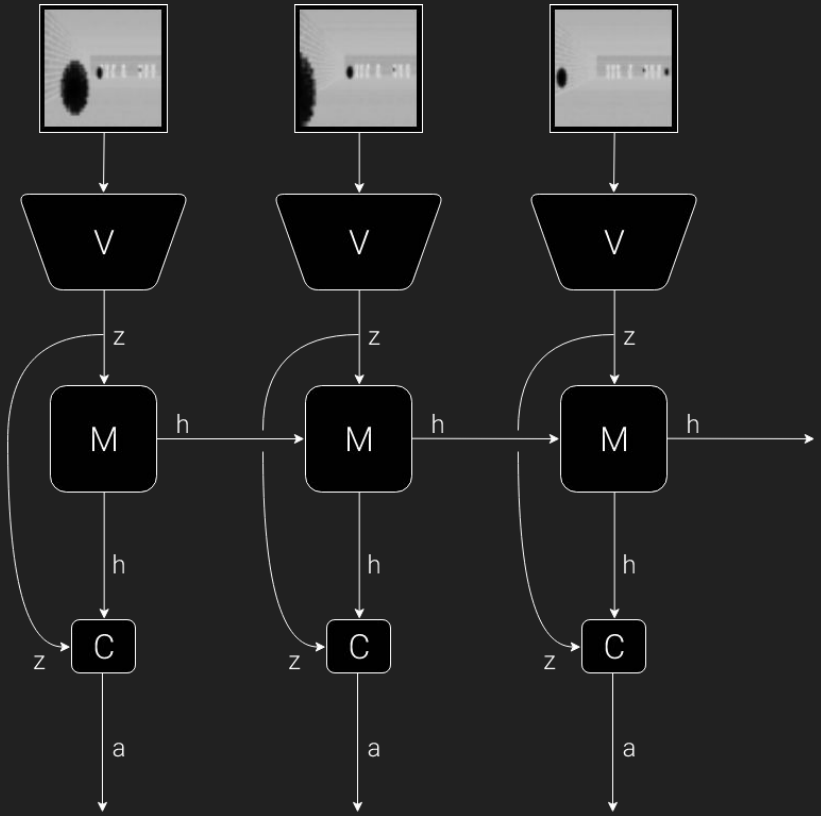
World Model

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

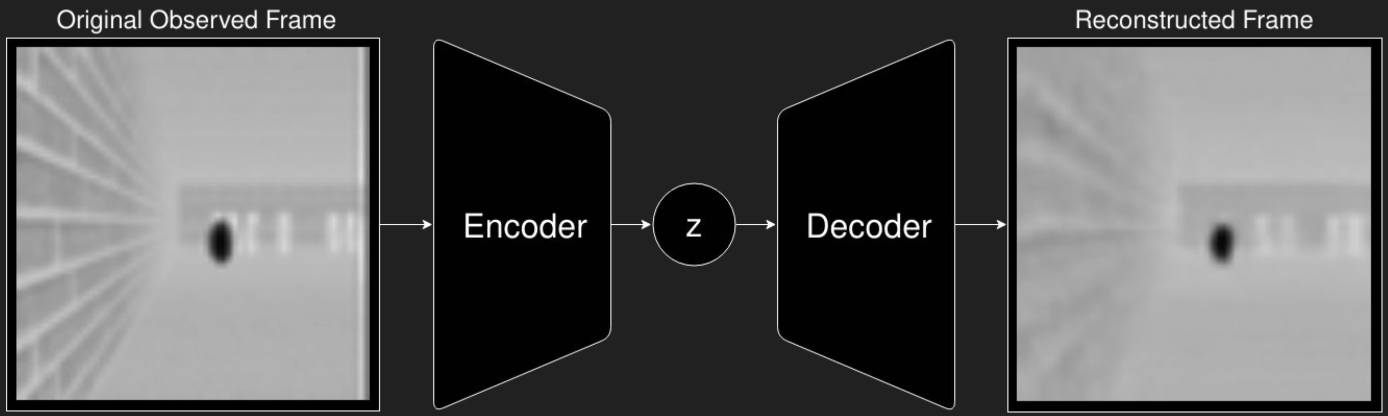
The Memory RNN (M) integrates the historical codes to create a representation that can predict future states.

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.

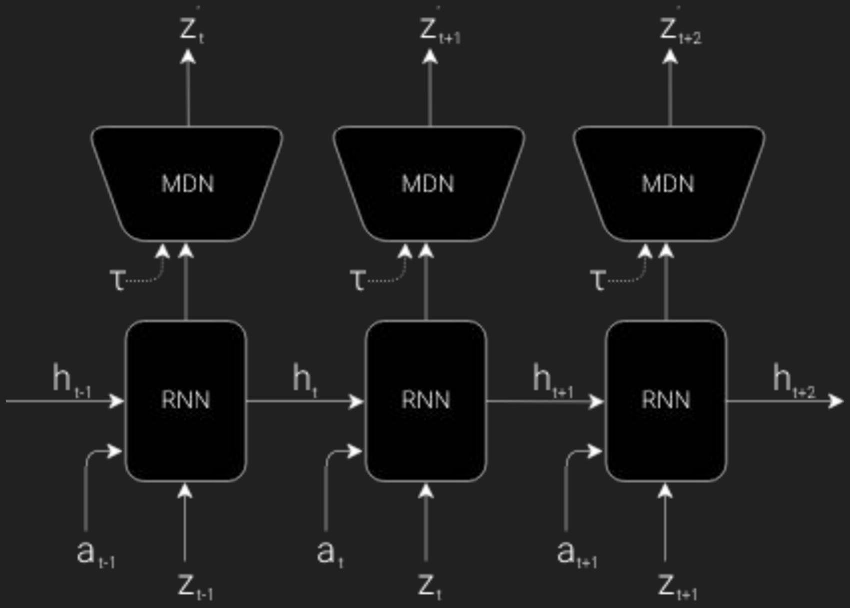


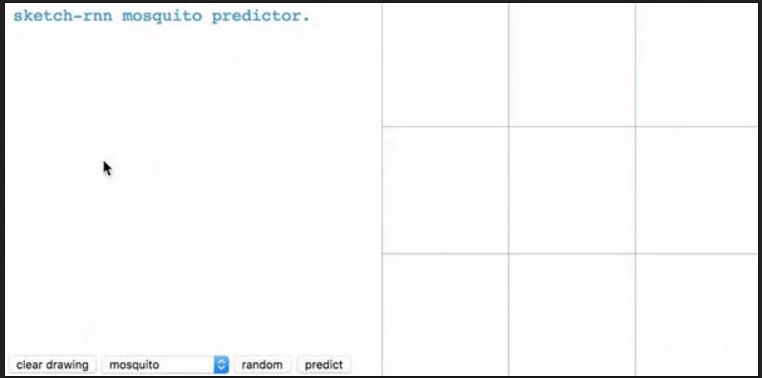
World Models: Vision Model

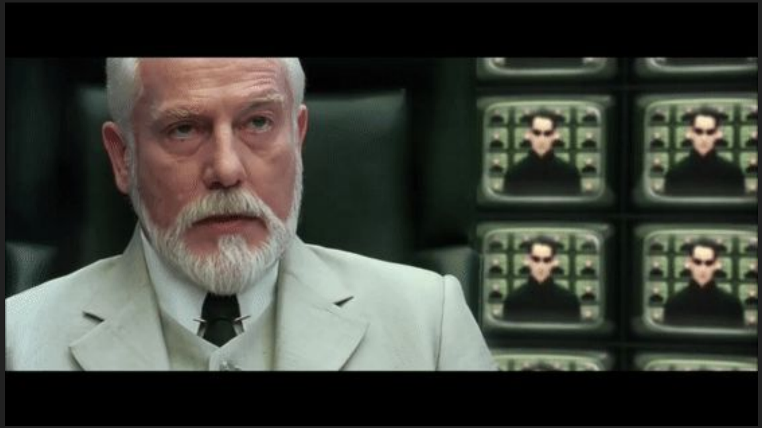


We can use a variational autoencoder for the latent representation.

World Models: Memory RNN

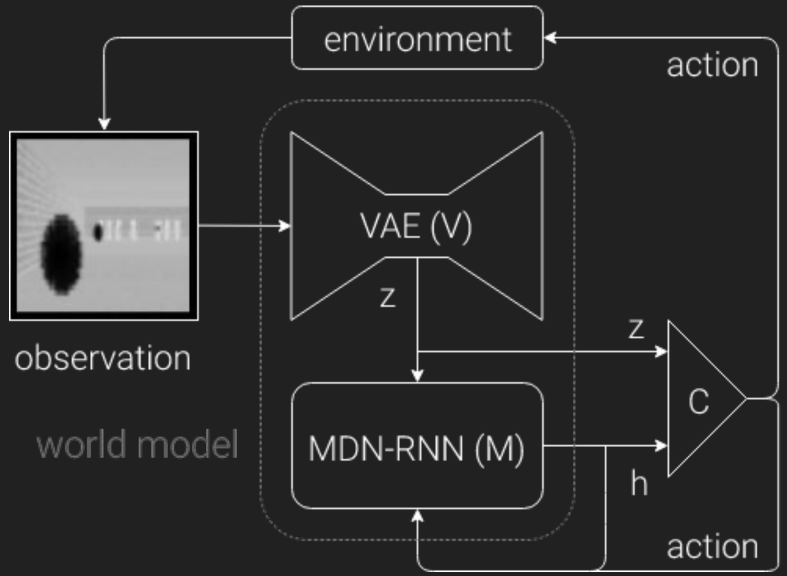


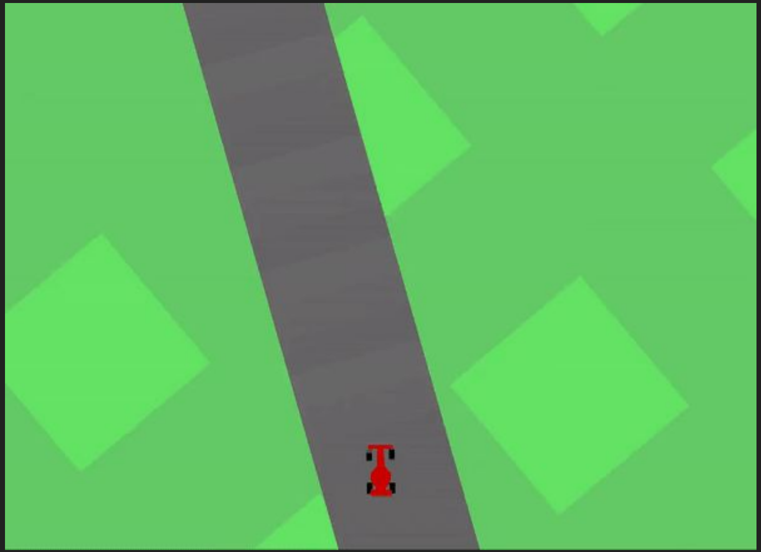




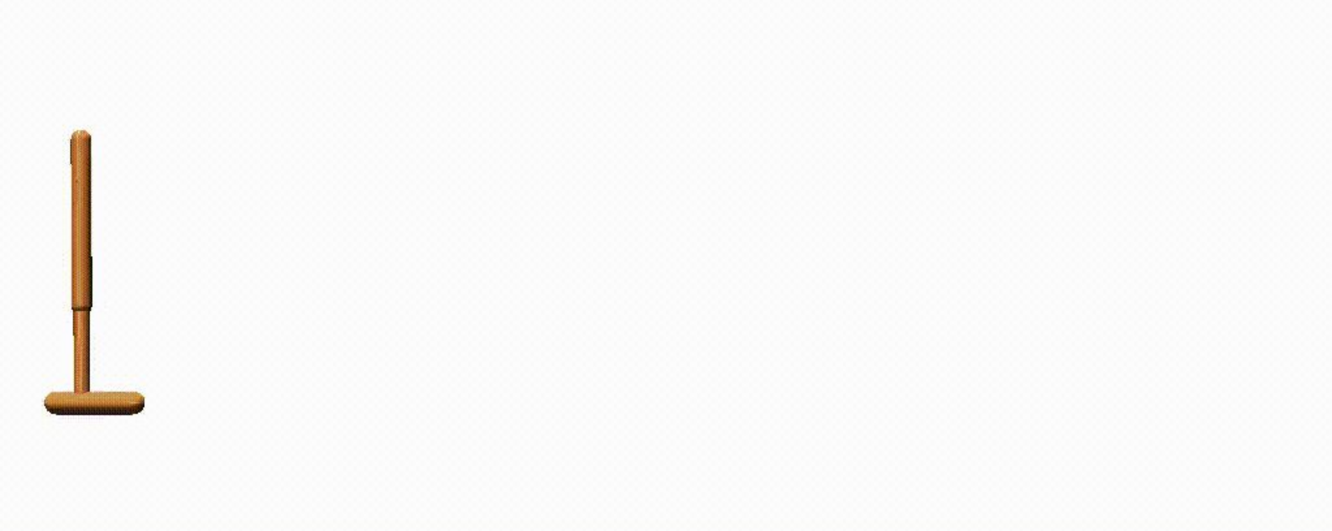
And an RNN with a Mixture Density Network output layer as a predictive model.

World Models: Final Architecture



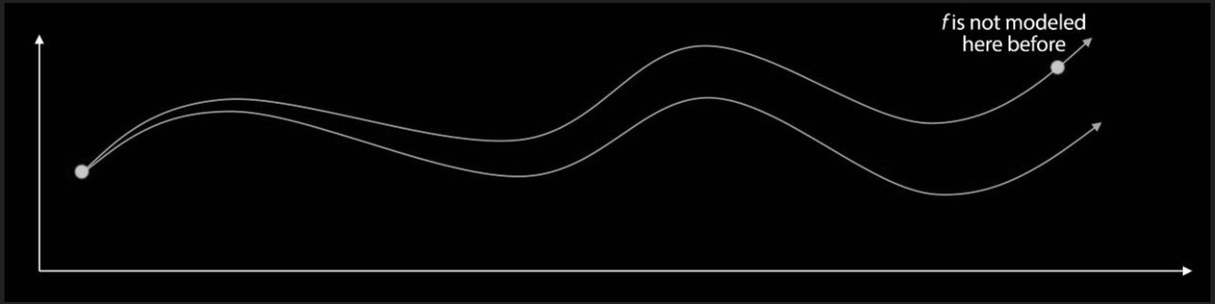


The bad news



Despite all our efforts, small errors still compound over actions.

Iterative Learning



Model is prone to drifting, hence we need to continue to fit it.

Modified Algorithm

1. Collect data under current (base) policy

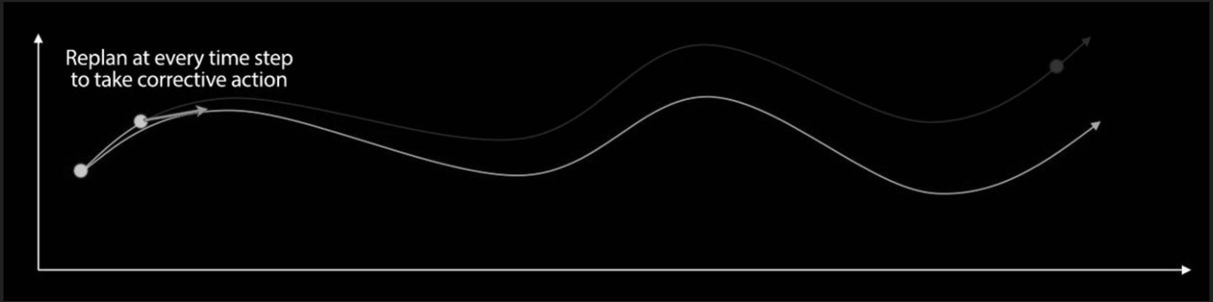
2. Create dynamics model (choose the appropriate type)

3. Learn from collected data

4. Use model to improve policy and choose actions

5. Add the resulting data to the collected data

Executing Actions



Agent executes all planned actions before fitting the model again. We may already be off-course.

Modified Algorithm

1. Collect data under current (base) policy

2. Create dynamics model (choose the appropriate type)

3. Learn from collected data

4. Use model to improve policy and choose the first planned action

5. Add the resulting data to the collected data

Overfitting in Model-based RL

Standard overfitting:

Neural network performs well on training data, but poorly on test data

In our case: Predicting s' from (s, a)

Additional overfitting challenge in Model-based RL: Model bias

Policy optimization tends to exploit regions with insufficient data.

The Takeaway Message

Model-based reinforcement learning is great

If you have a good model!

The Takeaway Message

Resulting policy from model-based architectures good in simulations but not the real world!

However, with some slight adjustments, we can improve the weaknesses.

→ Active research area ☺️

Final Algorithm?

1. Collect data under current (base) policy
2. Create dynamics model (choose the appropriate type) → Improvements needed?
3. Learn from collected data
4. Use model to improve policy and choose the first planned action
5. Add the resulting data to the collected data

Sources

Books and Papers:

- “Reinforcement Learning: An Introduction” (1998)
- “Model-based Reinforcement Learning: A Survey” (2021)
- “Learning to Paint With Model-based Deep Reinforcement Learning” (2019)
- “Algorithmic Framework for Model-based Deep Reinforcement Learning with Theoretical Guarantees” (2021)
- “Reinforcement Learning via Gaussian Processes with Neural Network Dual Kernels” (2020)
- “A saturation-balancing control method for enhancing dynamic vehicle stability” (2013)

Additional References:

- https://jonathan-hui.medium.com/rl-model-based-reinforcement-learning-3c2b6f0aa323
- https://medium.com/analytics-vidhya/introduction-to-model-based-reinforcement-learning-6db0573160da
- https://worldmodels.github.io/
- https://yanndubs.github.io/Neural-Process-Family/text/Intro.html
- https://github.com/xenomeno/RodManeuvering
- https://towardsdatascience.com/too-many-terms-ruins-the-regression-7cf533a0c612
- https://towardsdatascience.com/a-hitchhikers-guide-to-mixture-density-networks-76b435826cca
- https://www.cs.cmu.edu/~motionplanning/lecture/lec20.pdf
- https://bair.berkeley.edu/blog/2019/12/12/mbpo/
- https://towardsdatascience.com/reinforcement-learning-q-learning-with-decision-trees-ecb1215d9131
- https://medium.com/the-official-integrate-ai-blog/understanding-reinforcement-learning-93d4e34e5698

Additional Media:

- http://www.norsemathology.org/wiki/images/f/f9/Tic-tac-toe-game-tree.png
- https://thumbs.gfycat.com/AnimatedAmazingCirriped-size_restricted.gif
- https://y.yarn.co/295cb135-2a7a-4019-9130-c0ab1aef6e57_screenshot.jpg