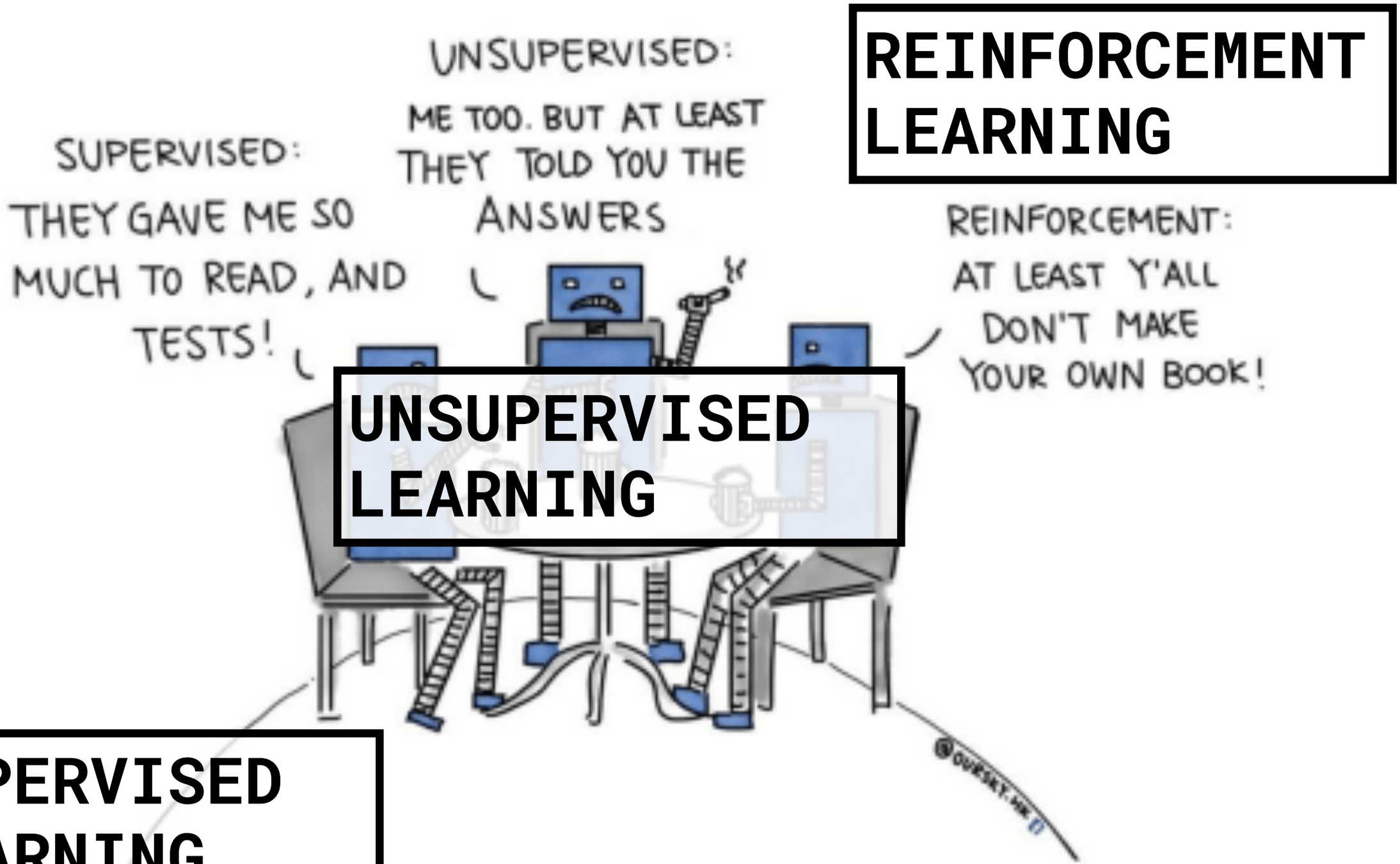
Reinforcement Learning

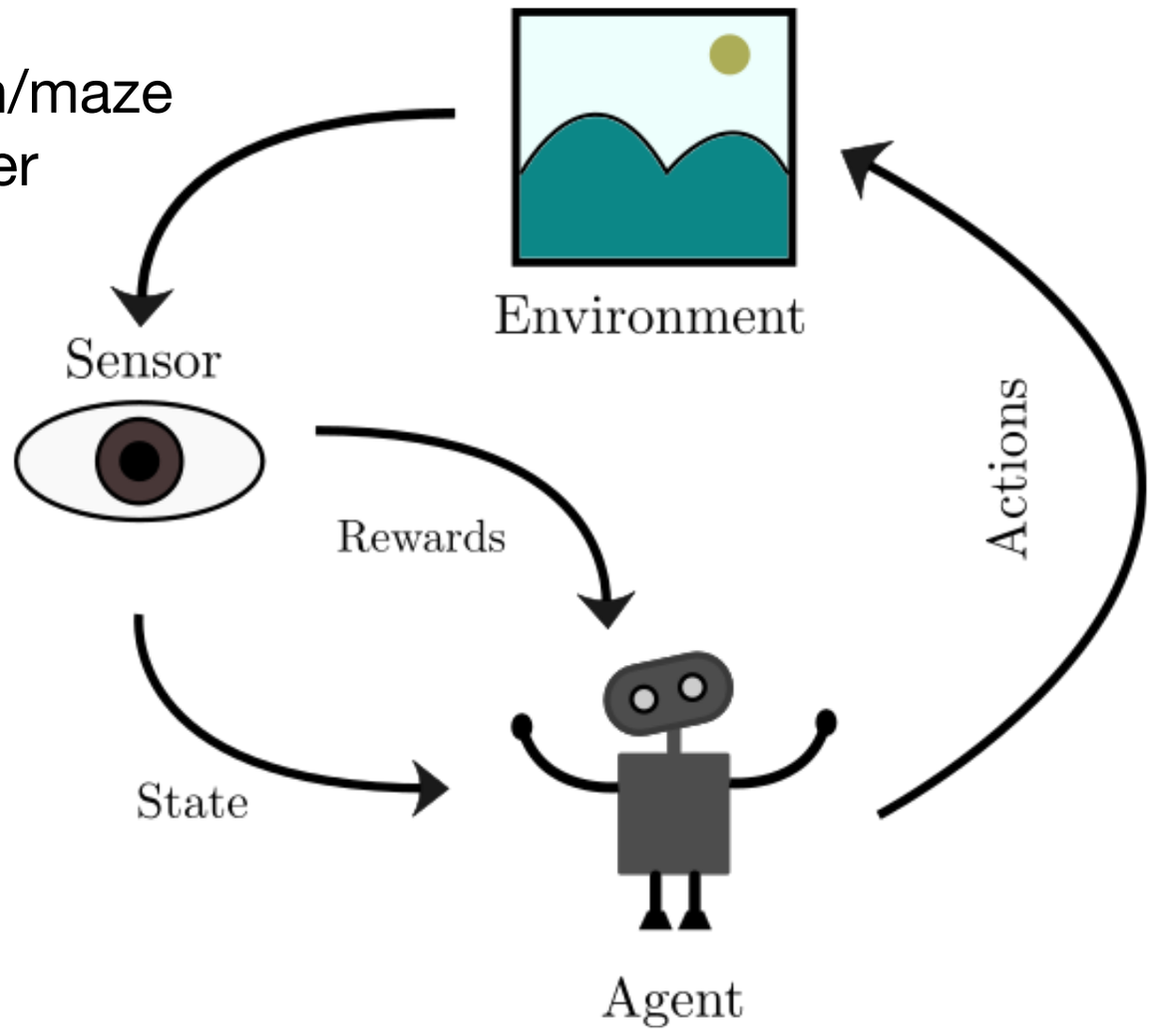
Give us help


REINFORCEMENT LEARNING
UNSUPERVISED:
ME TOO. BUT AT LEAST THEY TOLD YOU THE ANSWERS
SUPERVISED:
THEY GAVE ME SO MUCH TO READ, AND TESTS!
REINFORCEMENT:
AT LEAST Y'ALL DON'T MAKE YOUR OWN BOOK!
UNSUPERVISED LEARNING
SUPERVISED LEARNING
QUESTY.COM

Agent VS Environment

Possible environments:

1. real physical environment
2. 2D/3D grid/graph/maze
3. chess or any other game
4. text creation
5. etc



Source: https://technopremium.com/blog/intuition-behind-reinforcement-learning/

Possible actions:

1. control of the mechanical arm
2. steps in a grid/maze
3. chess move
4. word/letter
5. anything else

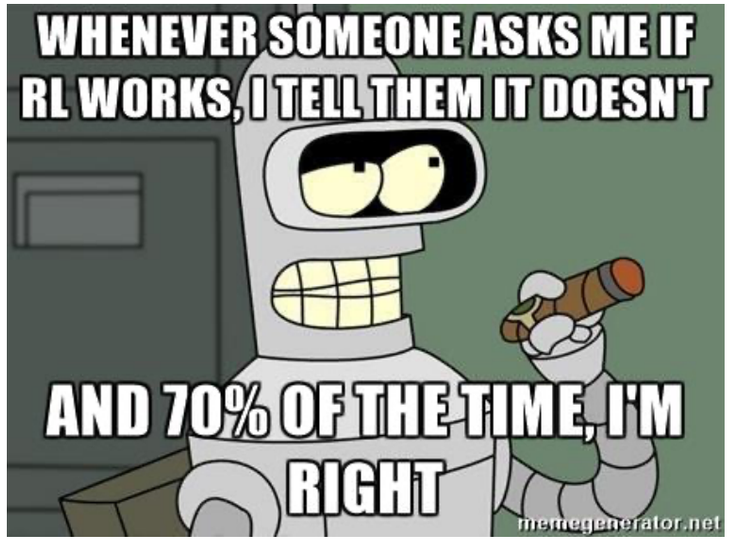
RL Disclaimer

SOUNDS LIKE IT CAN SOLVE A LOT OF PROBLEMS

True but usually unnecessary

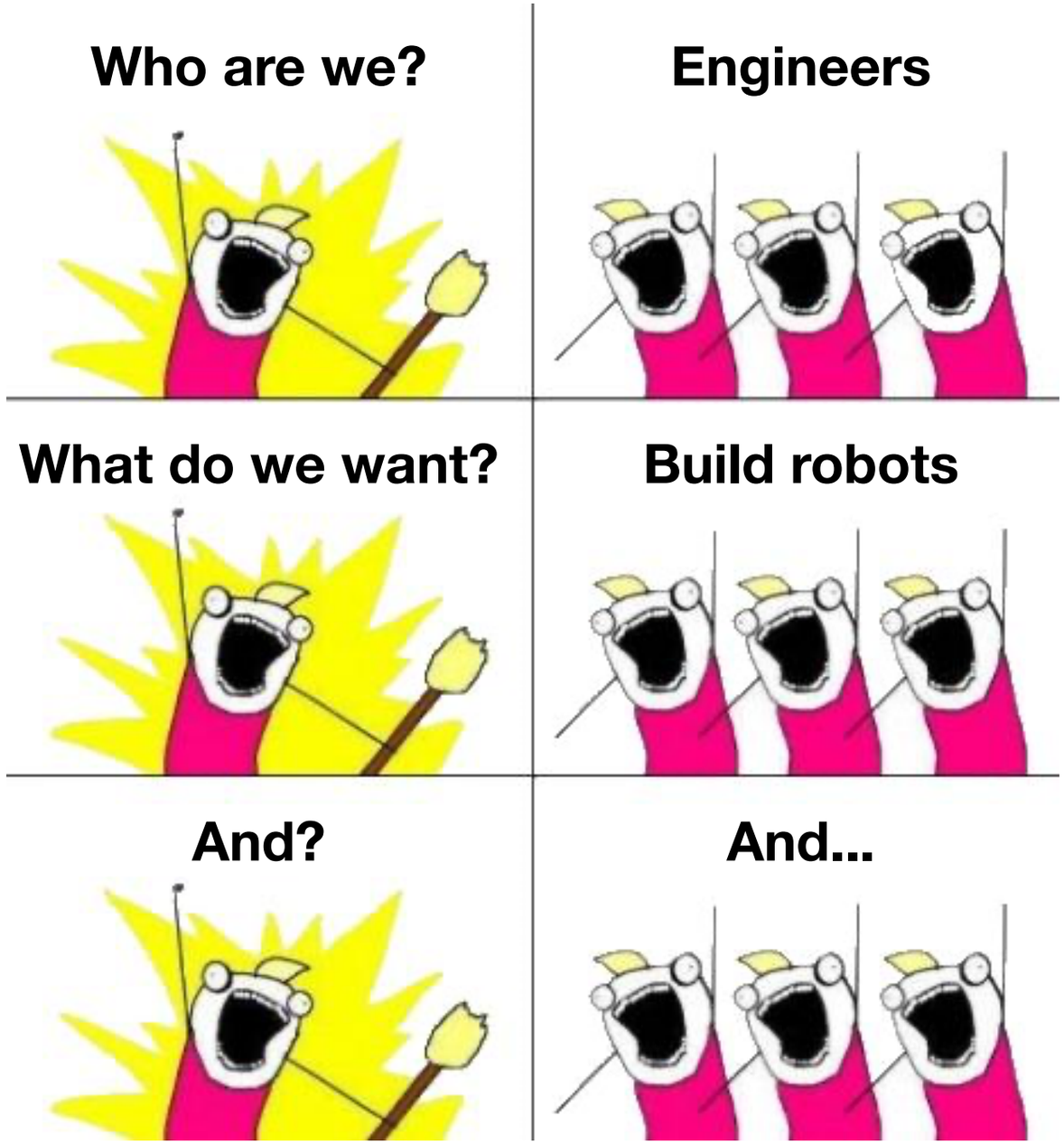
LESS MATH

MORE MATH



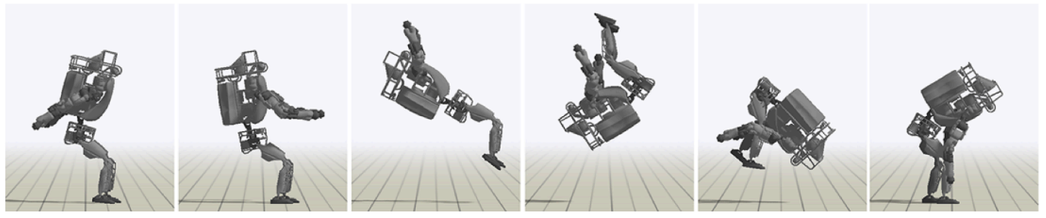
It works…
Just don’t get too excited

How do we do it?



We have:

1. Really cool robot
2. Lots of space
3. And we are willing to fix the robot everyday

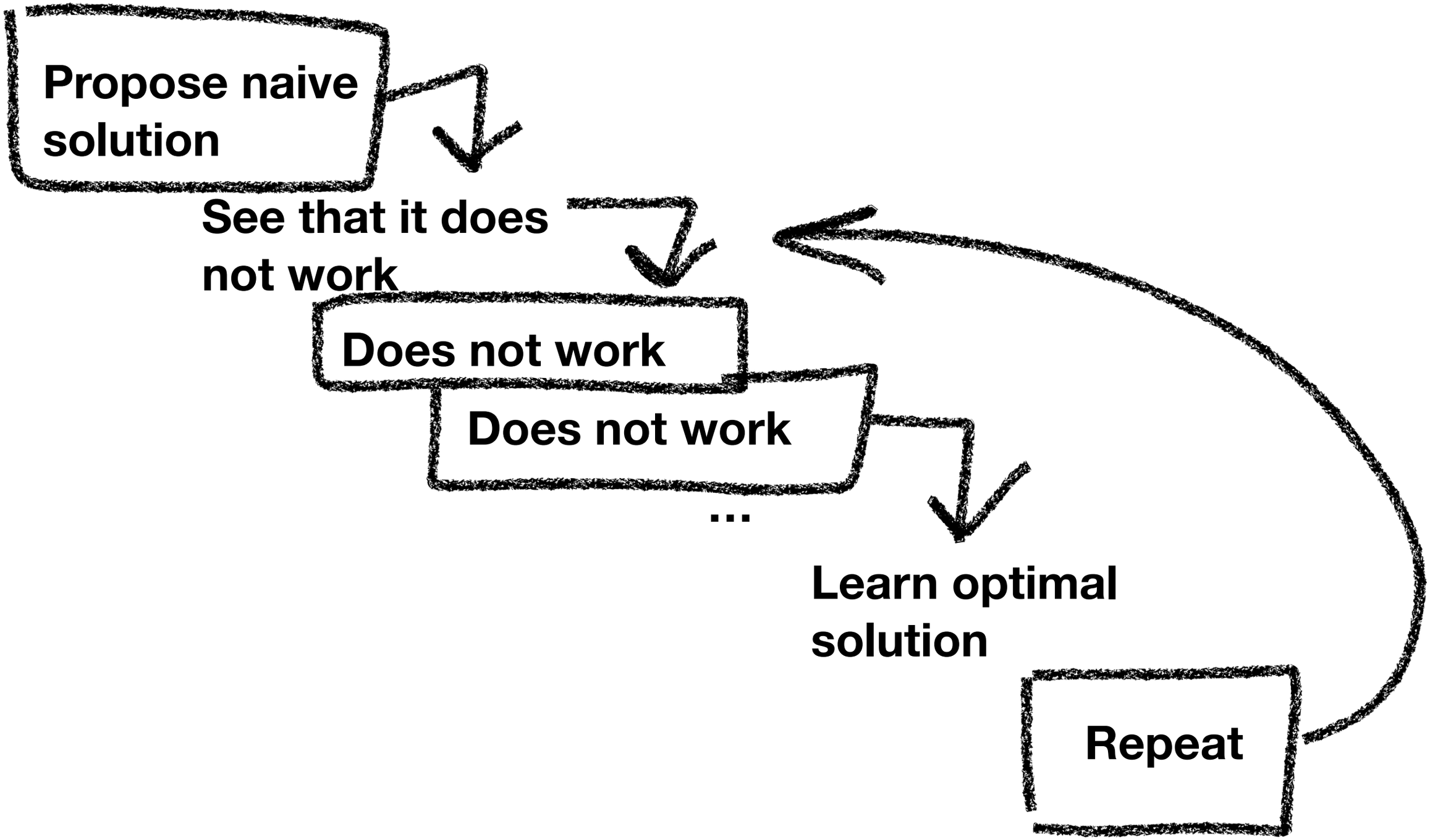


Source: https://arxiv.org/pdf/1804.02717.pdf

We want:

1. Learn to do a backflip
2. ~~Enslave the humanity~~

Duct tape approach

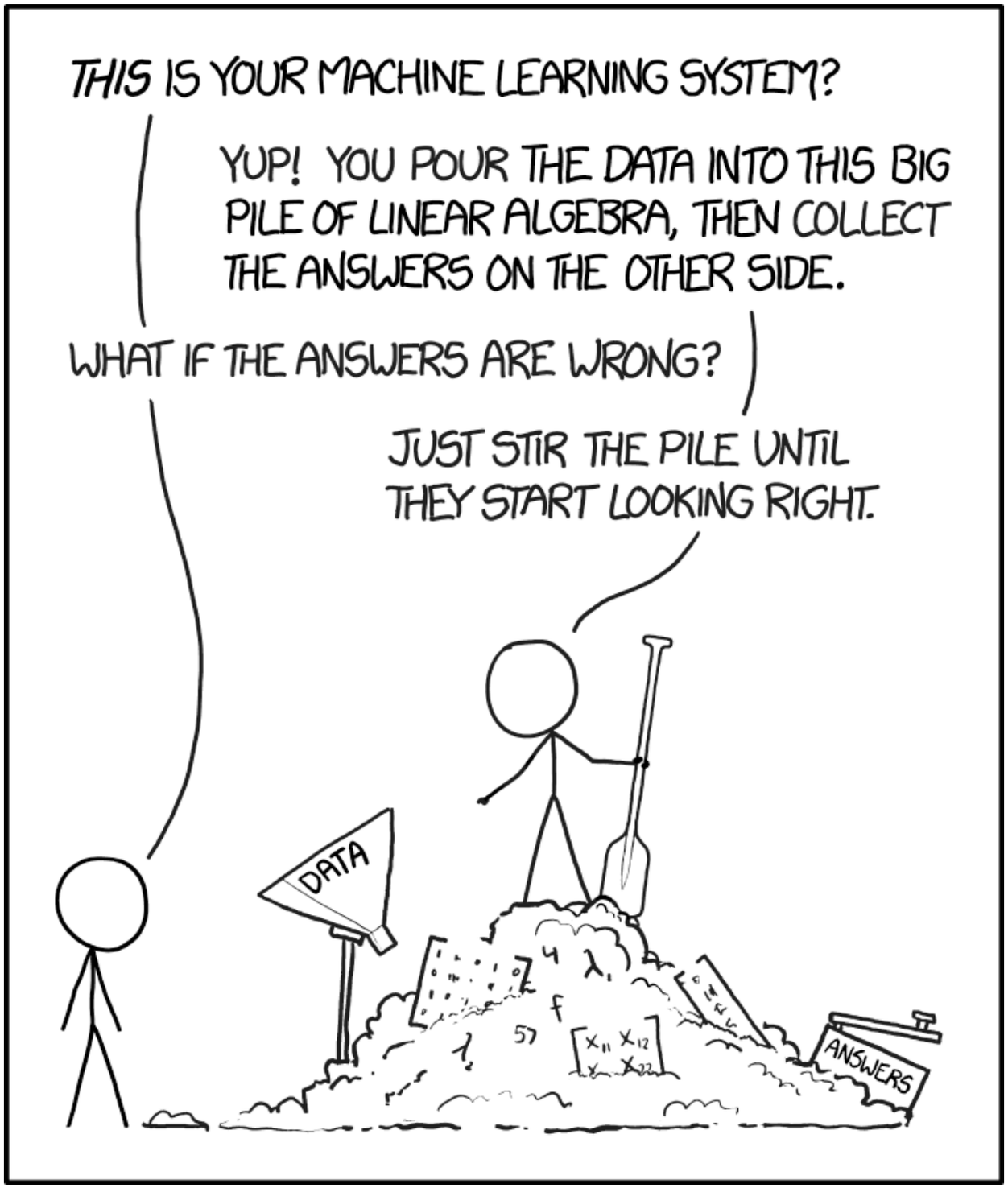


Some of the problems

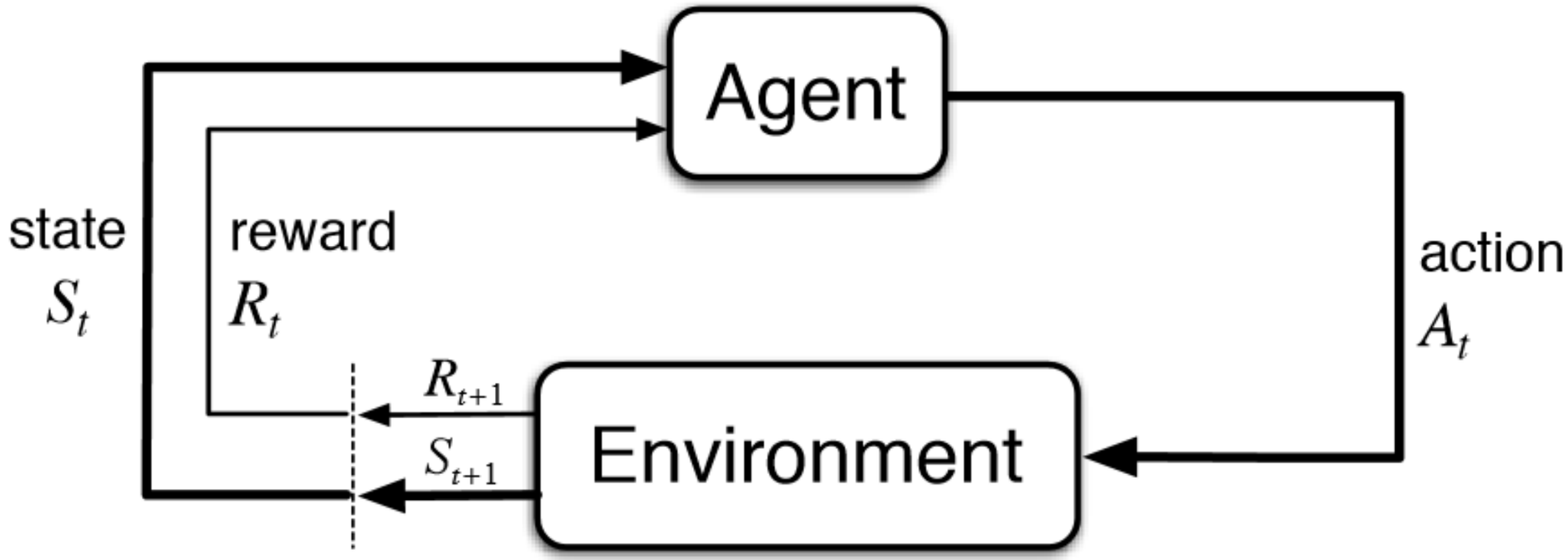
Extract as much money as you can right now

Make user happy so that he would visit you again

Getting into math



Let’s Define RL Task



Source: https://towardsdatascience.com/reinforcement-learning-demystified-36c39c11ec14

- **Environment —** Physical world in which the agent operates
- **State —** Current situation of the agent
- **Reward —** Feedback from the environment
- **Policy —** Method to map agent’s state to actions
- **Value —** Future reward that an agent would receive by taking an action in a particular state

Markov Decision Process (MDP)

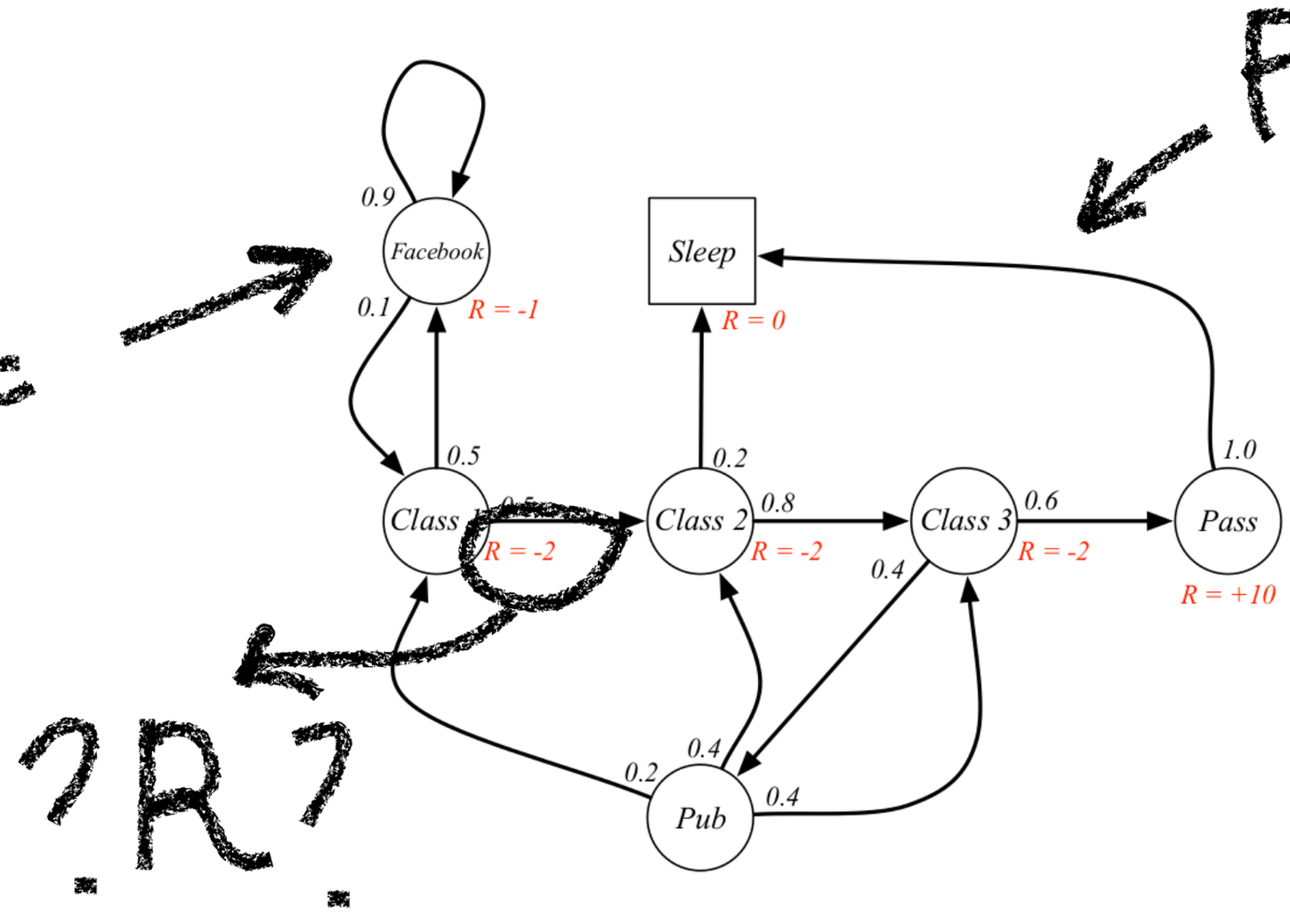
Markov property

" The future is independent of the past given the present."

Markov Process

A *Markov* process is a memory-less random process, i.e. a sequence of random states ***S1***, ***S2***, ….. with the *Markov* property.

A *Markov* process or *Markov* chain is a tuple (***S***, ***P***) on state space ***S***, and transition function ***P***.



Markov Decision Process

Markov Reward Process

In **Reinforcement learning**, we care about **maximizing** the cumulative reward (all the rewards agent receives from the environment) instead of, the reward agent receives from the current state(also called immediate reward). This **total sum of reward** the agent receives from the environment is called **returns**.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

R_{t+1} immediate reward we expect to get from state *S* performing action *A*

$\gamma \in [0,1]$ Discount Factor, trade-off between i**mmediate reward** and **future rewards**

The state-value function of an MRP is the expected return starting from state ***s***,

$$\boxed{V(s) = \mathbb{E}[G_t \mid S_t = s]}$$

Markov Decision Process

Policy Function and Value Function

A policy is a simple function, that defines a **probability distribution** over Actions (a∈ A) for each state (s ∈ S).

$$\pi(a \mid s) = \mathbb{P}\left[A_t = a \mid S_t = s\right] \quad \text{— policy function}$$

$$V_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] \quad \text{— state-value function}$$

Bellman Equation

Bellman Equation states that value function can be **decomposed** into two parts:

- Immediate Reward, R[t+1]
- Discounted value of successor states,

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

Dynamic Programming (Value iteration and Policy iteration),
Monte-Claro methods and **TD-Learning**.

Markov Decision Process

What is a Markov Decision Process?

It is a tuple of (***S***, ***A***, ***P***, ***R***, γ) where:

- S is a set of states,
- A is the set of actions agent can choose to take,
- P is the transition Probability Matrix,
- R is the Reward accumulated by the actions of the agent,
- γ is the discount factor.

Transition Probability Matrix $$P^a_{ss'} = \mathbb{P}\left[S_{t+1} = s' \,|\, S_t = s, A_t = a\right]$$

Reward Function $$R^a_s = \mathbb{E}\left[R_{t+1} \,|\, S_t = s, A_t = a\right]$$

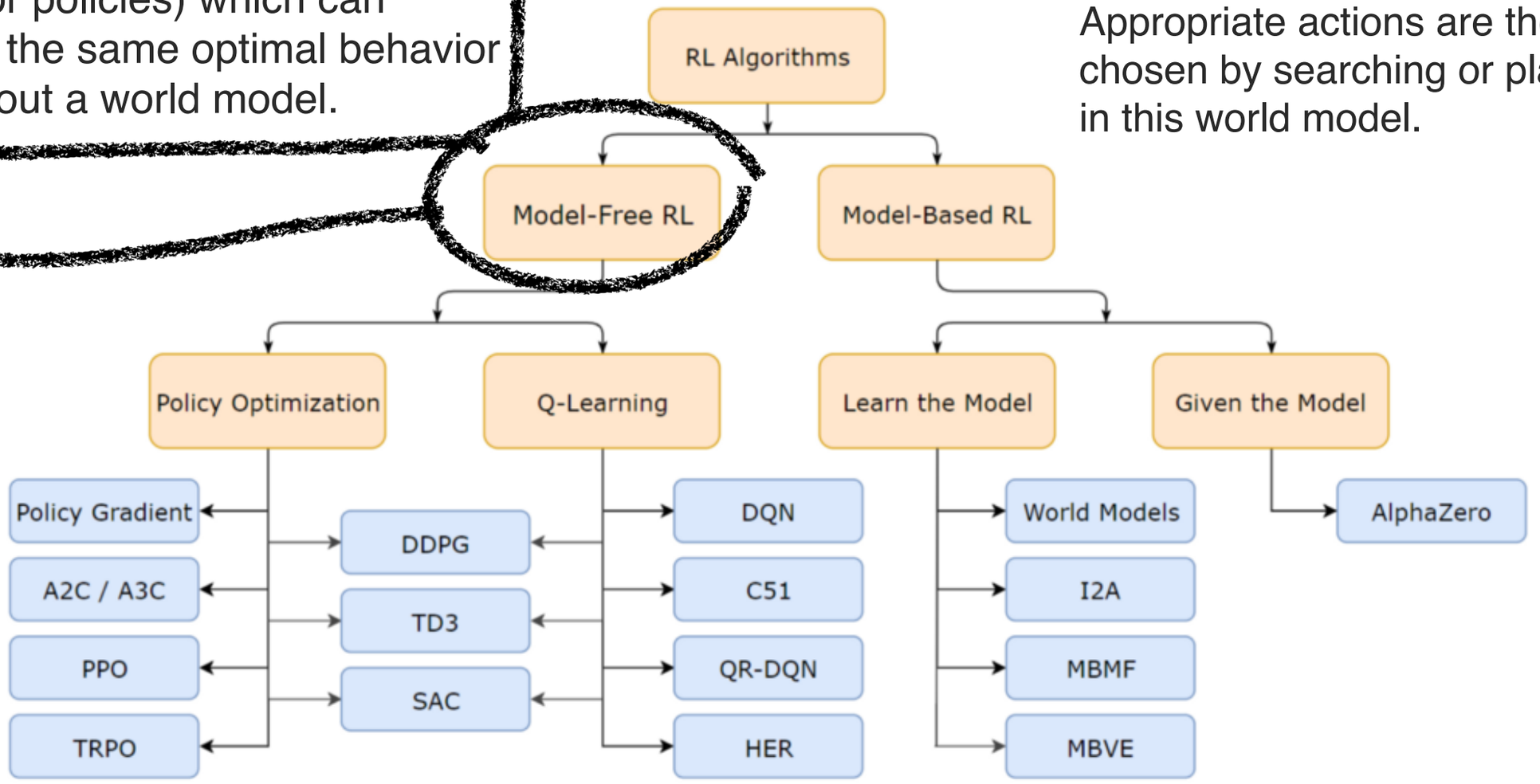
State-action value function or Q-Function

$$\boxed{Q_\pi(s,a) = \mathbb{E}_\pi\left[G_t \,|\, S_t = s, A_t = a\right]}$$

Reinforcement Learning algorithms

Model-free RL, uses experience to learn directly one or both of two simpler quantities (state/ action values or policies) which can achieve the same optimal behavior but without a world model.

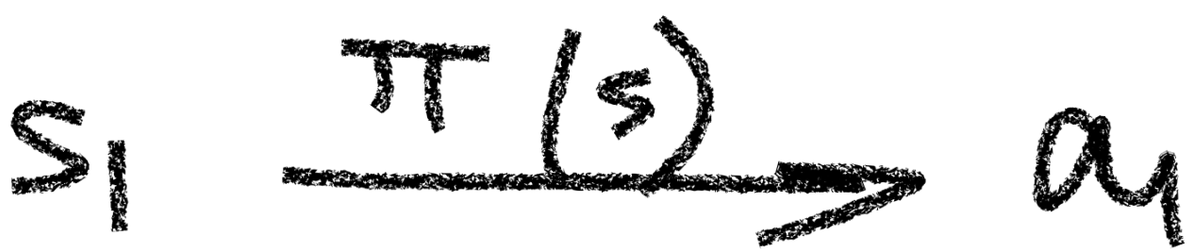
Model-based RL uses experience to construct an internal model of the transitions and immediate outcomes in the environment. Appropriate actions are then chosen by searching or planning in this world model.



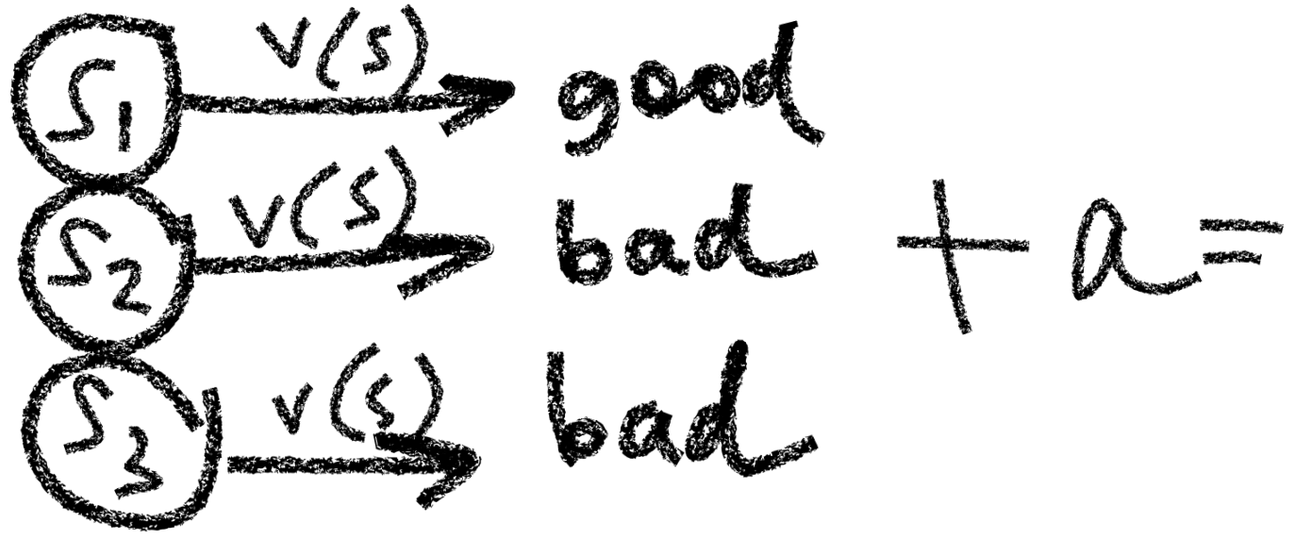
Source: https://medium.com/@SmartLabAI/reinforcement-learning-algorithms-an-intuitive-overview-904e2dff5bbc

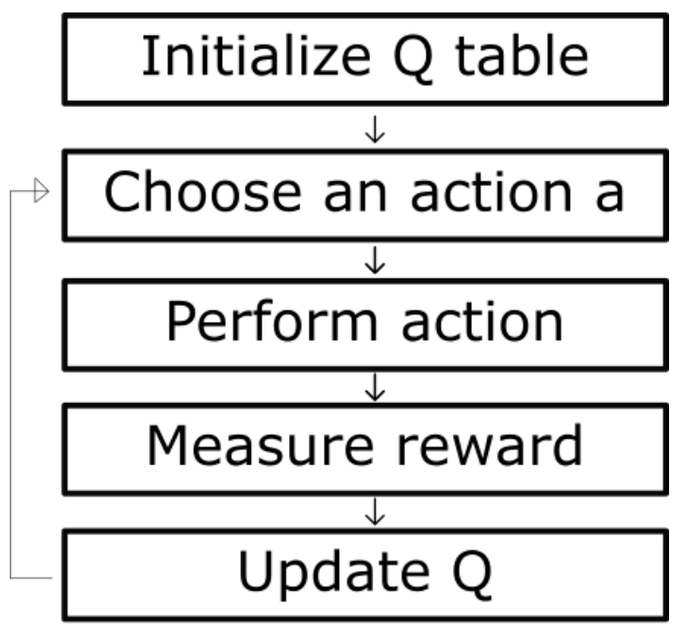
Model-free RL

Policy optimization or policy-iteration methods



Q-learning or value-iteration methods





Deep Q Neural Network (DQN)

approx $Q(s,a)$

Model-free RL

Policy Gradient (PG)

$$\pi_\theta = \mathbb{P}[a \mid s] \qquad J(\theta) = \mathbb{E}_{\pi\theta}\left[\sum \gamma r\right]$$ — policy score function

Main steps:

- Measure the quality of a policy with the policy score function.
- Use policy gradient ascent to find the best parameter that improves the policy.

Asynchronous Advantage Actor-Critic (A3C)

Several agents are trained in it's own copy of the environment and the model form these agent's are gathered in a master agent.

Similarly to PG — the update rule use the dicounted returns from a set of experiences.

The network will estimate both a value function ***V(s)*** (how good a certain state is to be in) and a policy ***π(s).***

Model-based RL

Model-based RL has a strong influence from control theory, and the goal is to plan through an *f(s,a)* control function to choose the optimal actions.

Learn the Model

- World models: the agent can learn from it's own "dreams" due to the Variable Auto-encoders

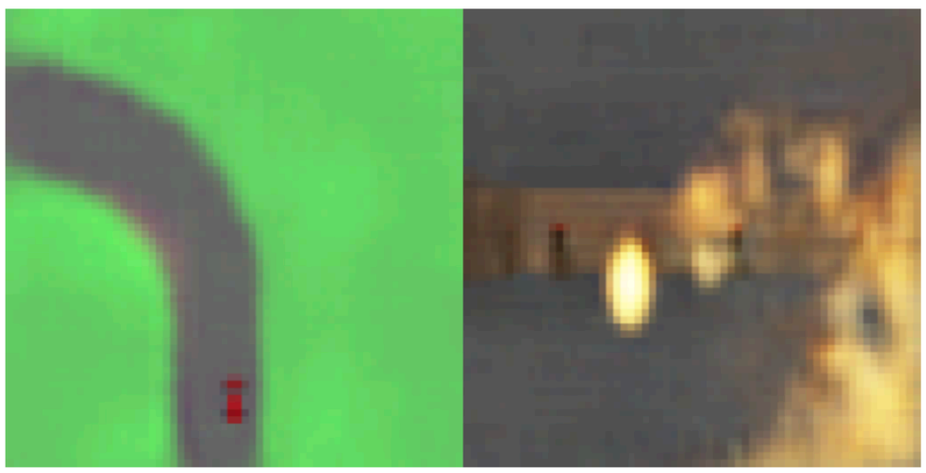
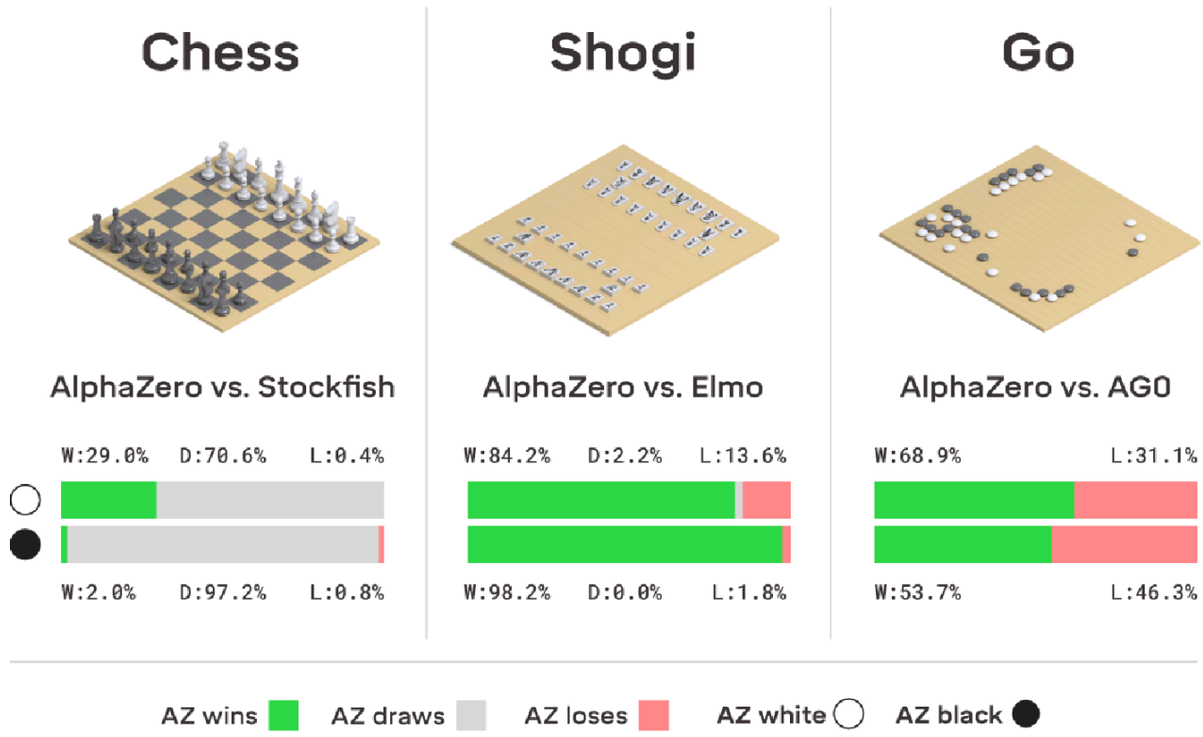


Figure 3. In this work, we build probabilistic generative models of OpenAI Gym environments. The RNN-based world models are trained using collected observations recorded from the actual game environment. After training the world models, we can use them mimic the complete environment and train agents using them.

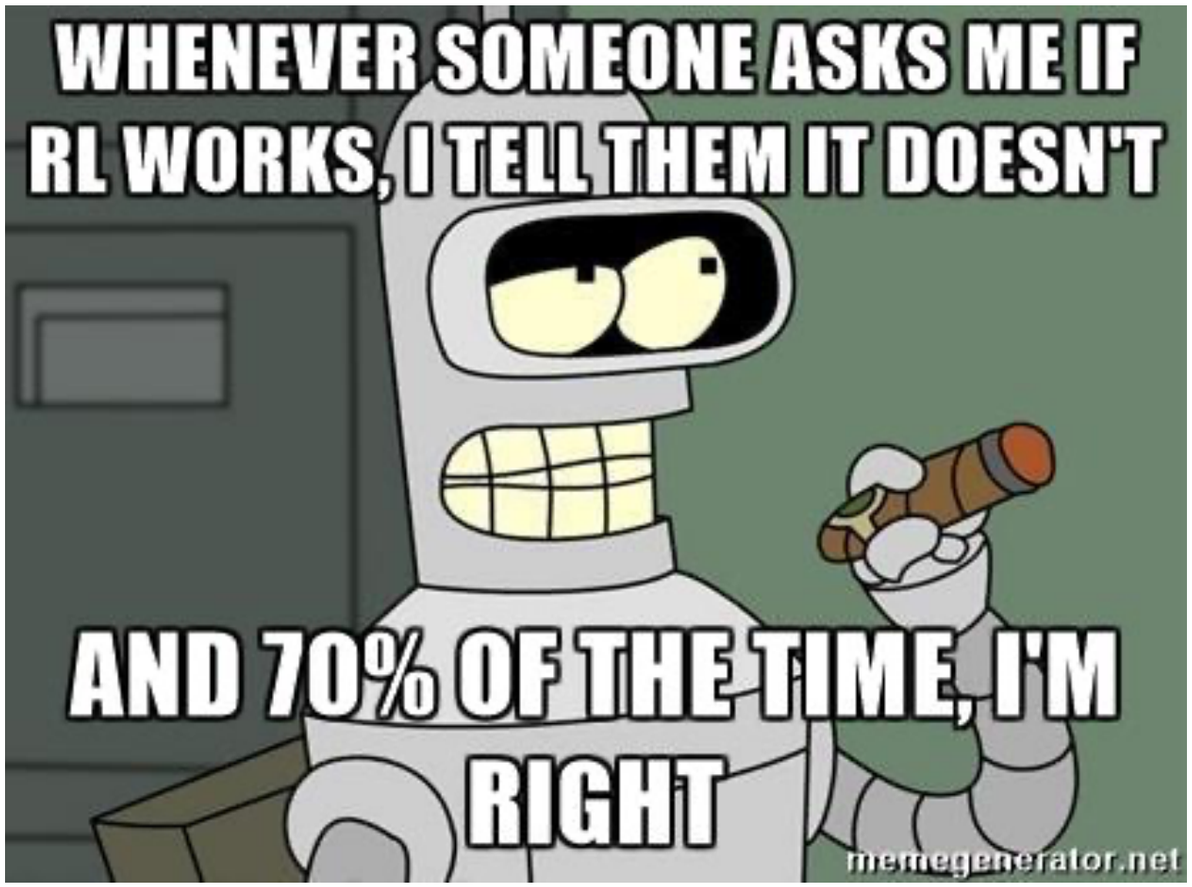
Given the Model



Just a few personal faves

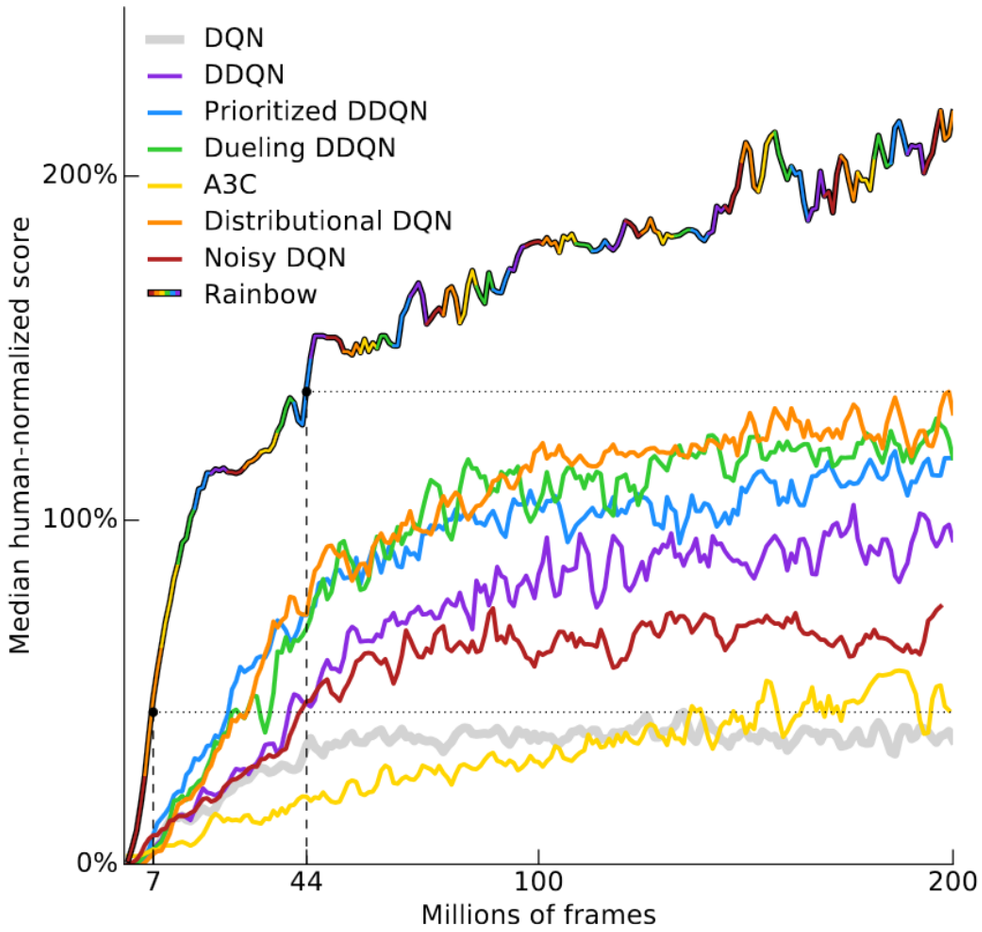
DeepMind's work on Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Policy updates

Why people still think it



does not work

Horribly Sample Inefficient



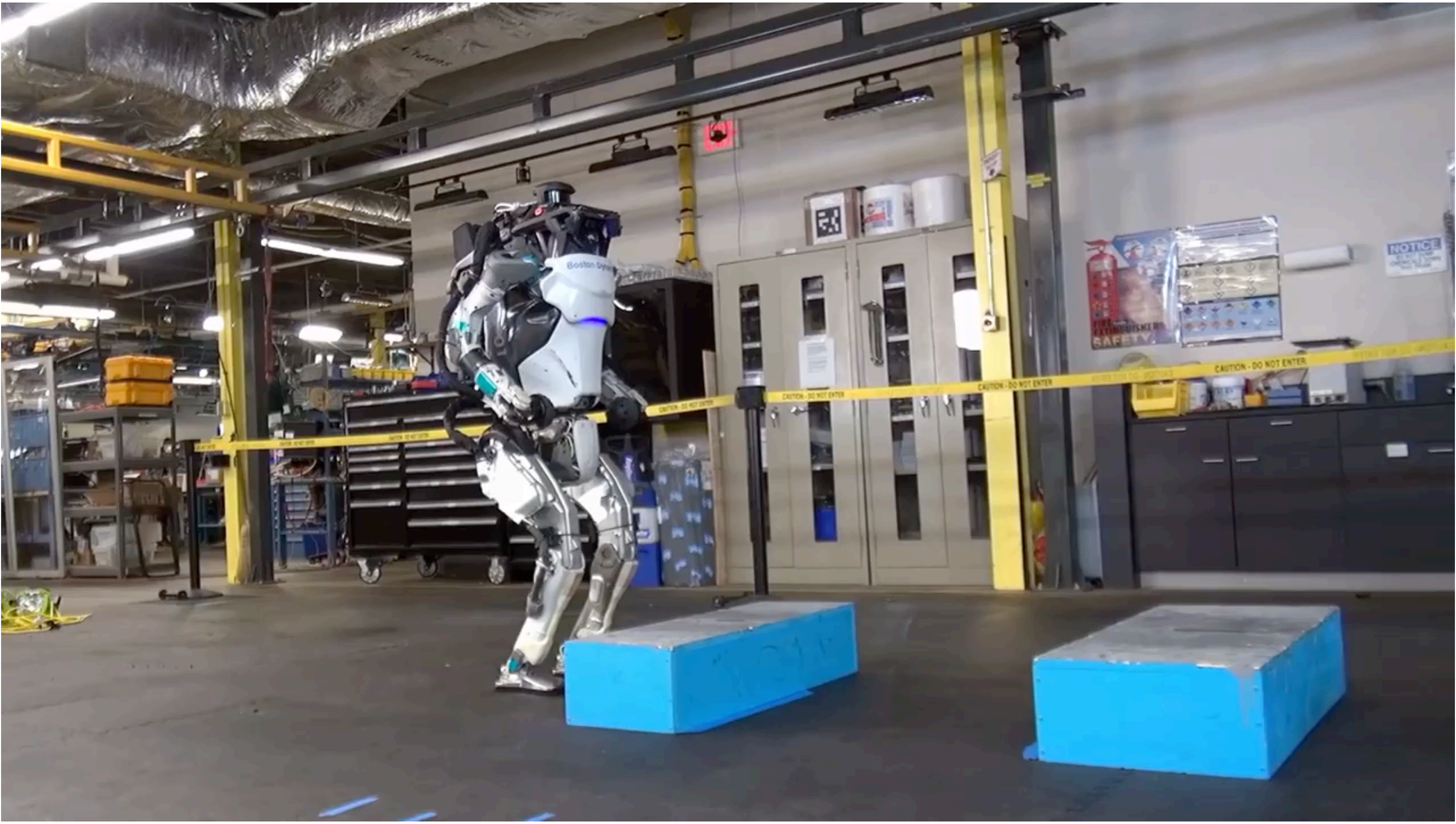
200 Million Frames

6400 CPU hours

Still not good enough

Synthesis of Complex Behaviors
with
Online Trajectory Optimization

(preliminary results)


CAUTION - DO NOT ENTER
NOTICE
SAFETY


START
START
SCORE
0
LAPS
—/3
TIME
0:01
TURBO
MORE GAMES

Natalia Soboleva

natalia.soboleva@skoltech.ru

@tasssshhhha

n.sob55@gmail.com