Reinforcement learning

Episode 0

Introduction to reinforcement learning

Yandex
Data Factory

LAMBDA

British Hedgehog
Preservation Society

Supervised learning

Given:

- objects and answers $\left(x, y\right)$
- algorithm family $a_\theta\left(x\right) \rightarrow y$
- loss function $L\left(y, a_\theta\left(x\right)\right)$

Find:

$$\theta' \leftarrow argmin_\theta L\left(y, a_\theta\left(x\right)\right)$$

Supervised learning

Given:

- objects and answers $\left(x, y\right)$

  [banner,page], ctr

- algorithm family $a_{\theta}\left(x\right) \to y$

  linear / tree / NN

- loss function $L\left(y, a_{\theta}\left(x\right)\right)$

  MSE, crossentropy

Find:

$$\theta' \leftarrow argmin_{\theta} L\left(y, a_{\theta}\left(x\right)\right)$$

Supervised learning

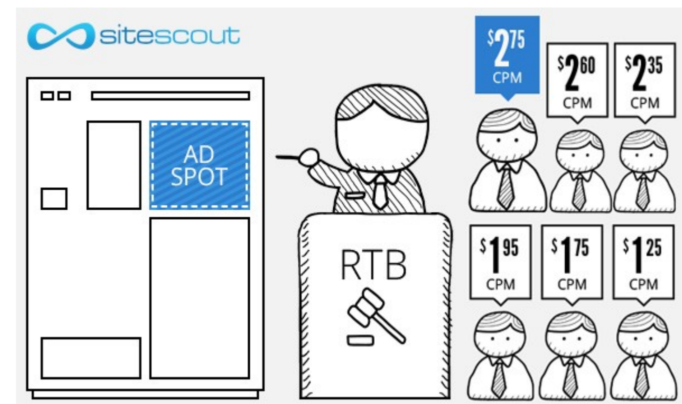
Great... except if we have no reference answers

Online Ads

Great... except if we have no reference answers

We have:

- YouTube at your disposal
- Live data stream
  (banner & video features, #clicked)
- (insert your favorite ML toolkit)



We want:

- Learn to pick relevant ads

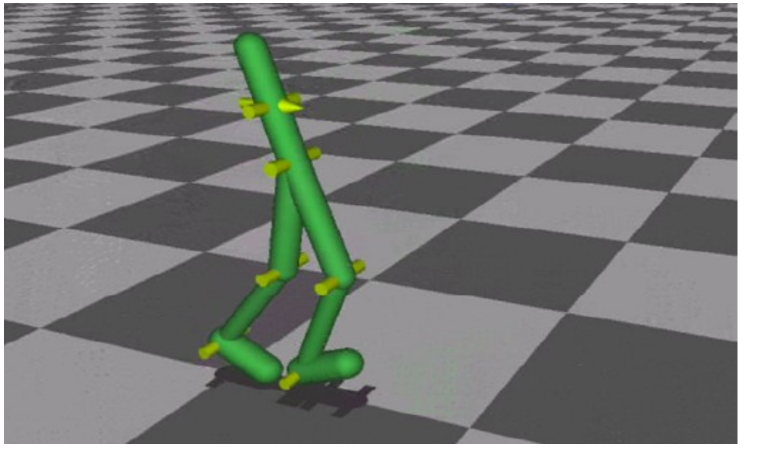
Ideas?

Giant Death Robot (GDR)

Great... except if we have no reference answers

We have:

- Evil humanoid robot
- A lot of spare parts
  to repair it :)

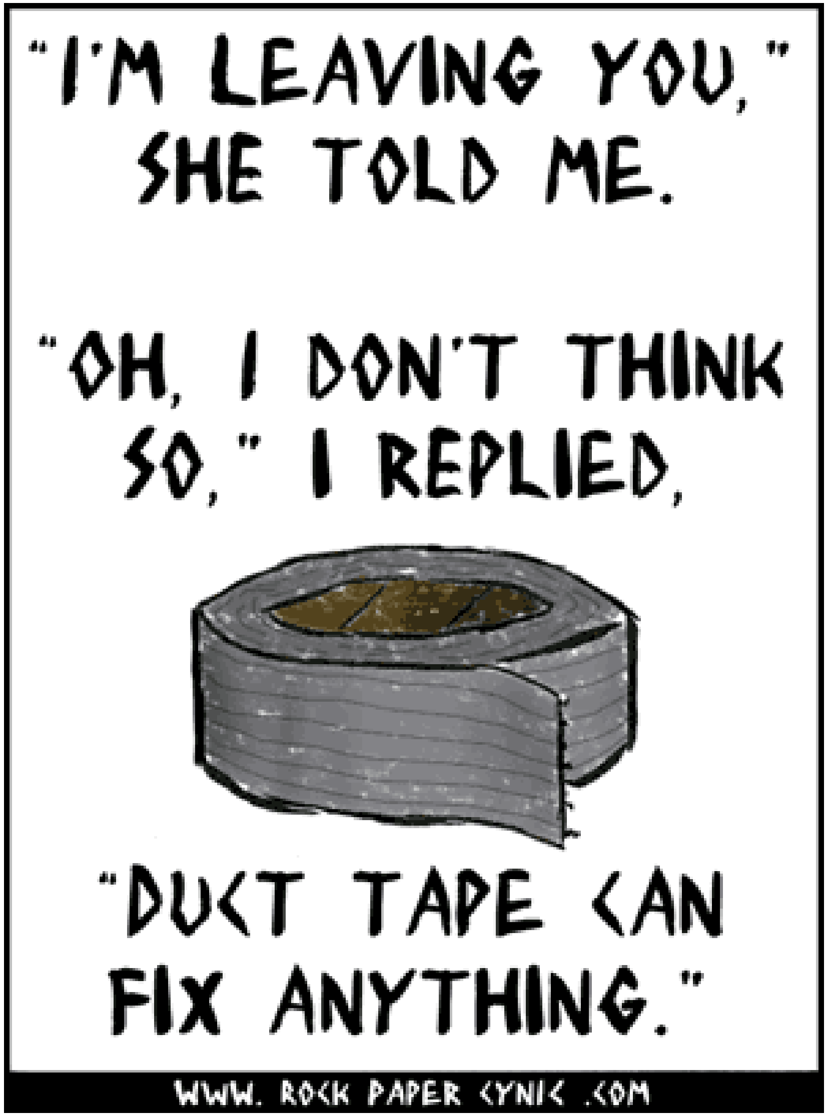


We want:

- ~~Enslave humanity~~
- Learn to walk forward

Ideas?

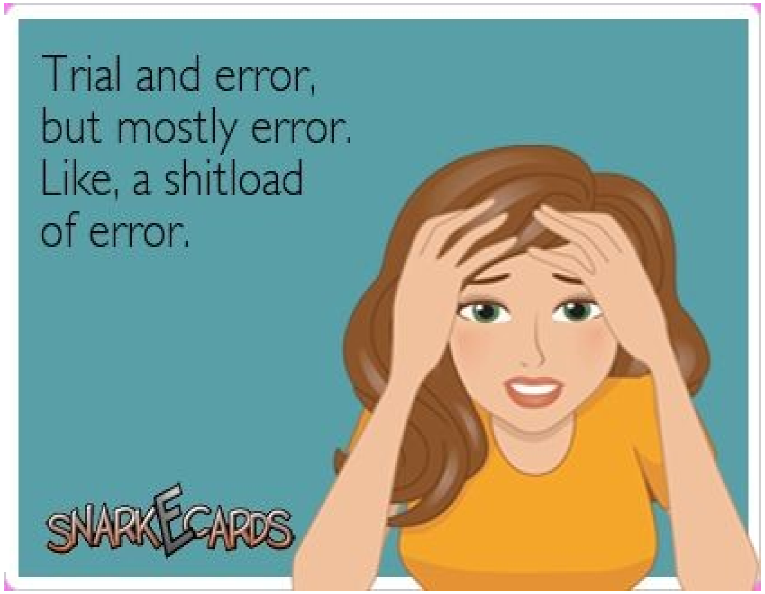
Duct tape approach



Duct tape approach

Common idea:

- Initialize with naïve solution
- Get data by trial and error and error and error and error
- Learn (situation) → (optimal action)
- Repeat



Duct tape approach

Problem 1:

- What exactly does the “optimal action” mean in the Giant Death Robot setting?

Push yourself forward as far as you can at each tick

VS

Do what allows you to walk farther over next N seconds

Duct tape approach

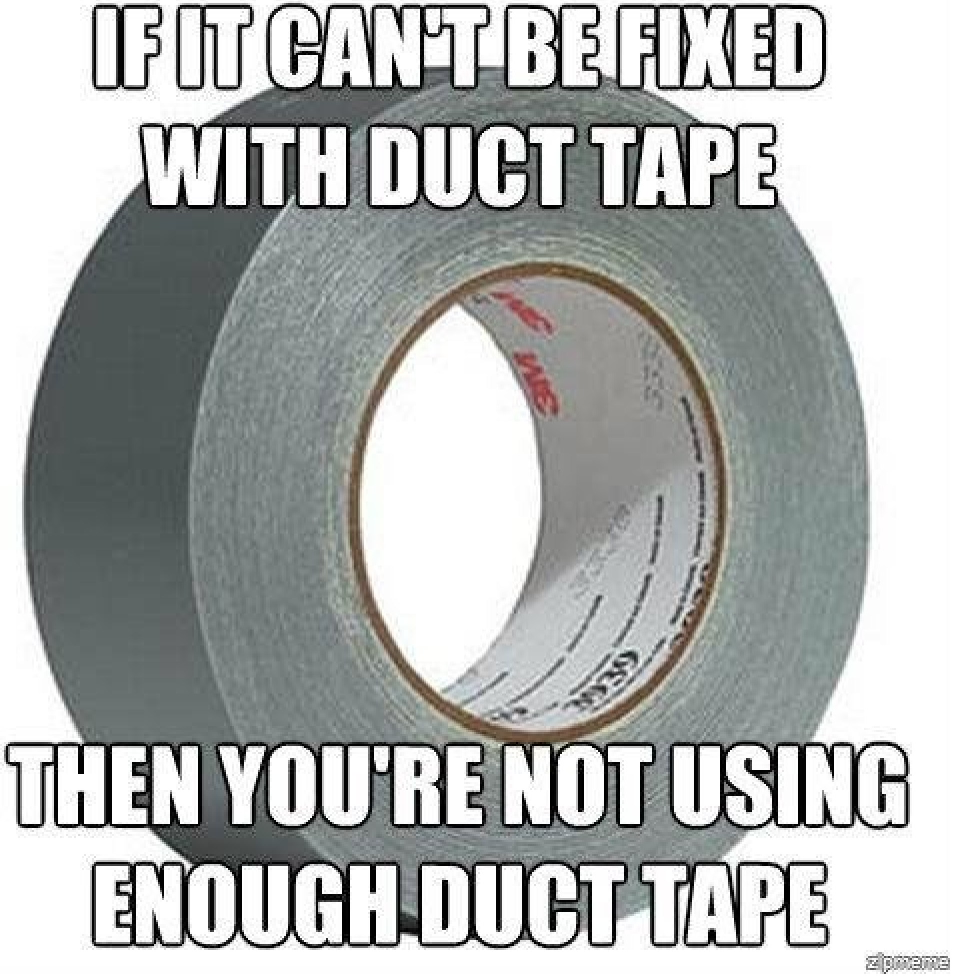
Problem 2:

- If you only act by the “current optimal” policy, you may never hit the global optimum.

- If your learned to fall down and crawl forward, that it will never get examples of how to walk because it always crawls.

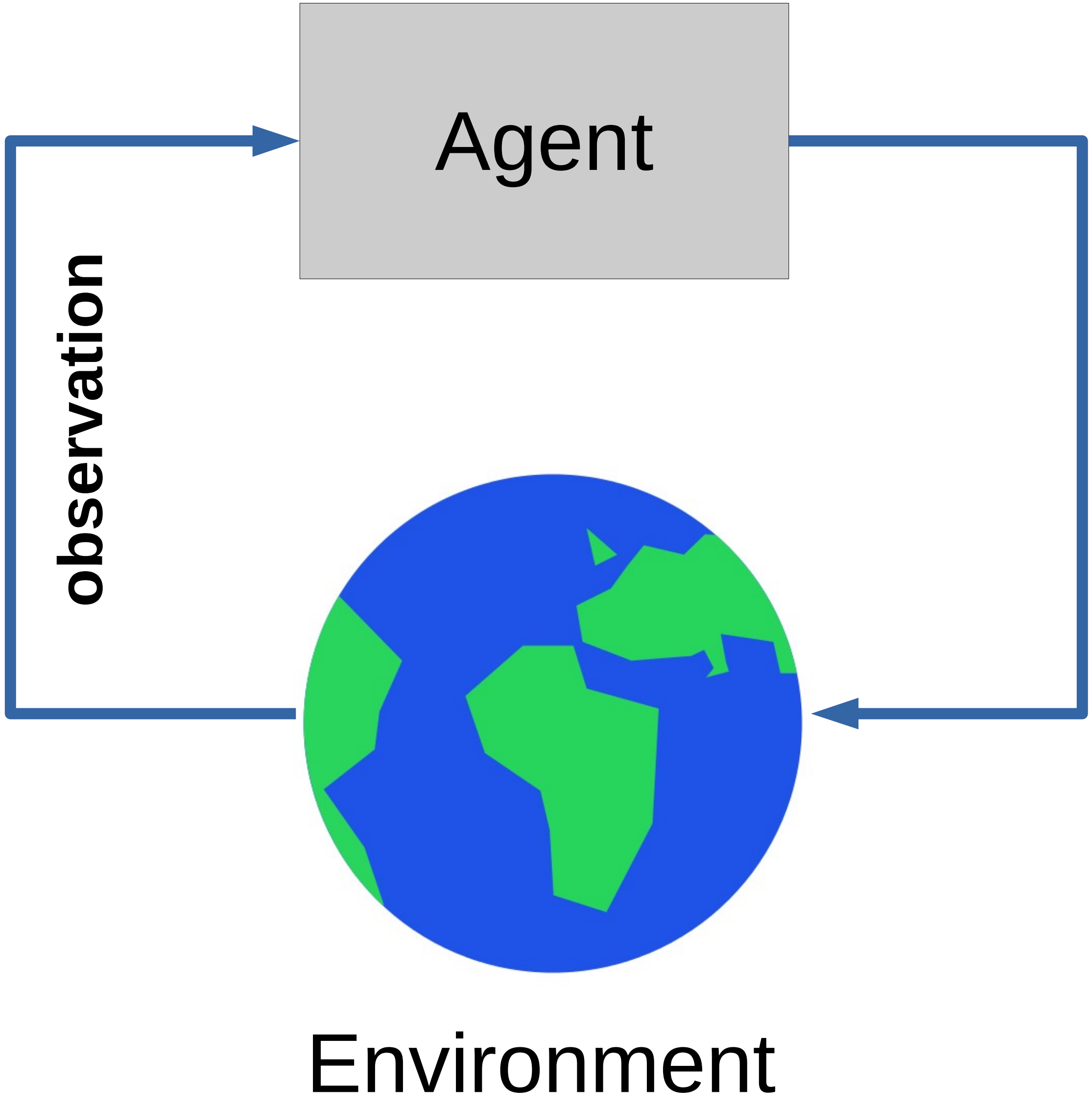
- **Ideas?**

Duct tape approach

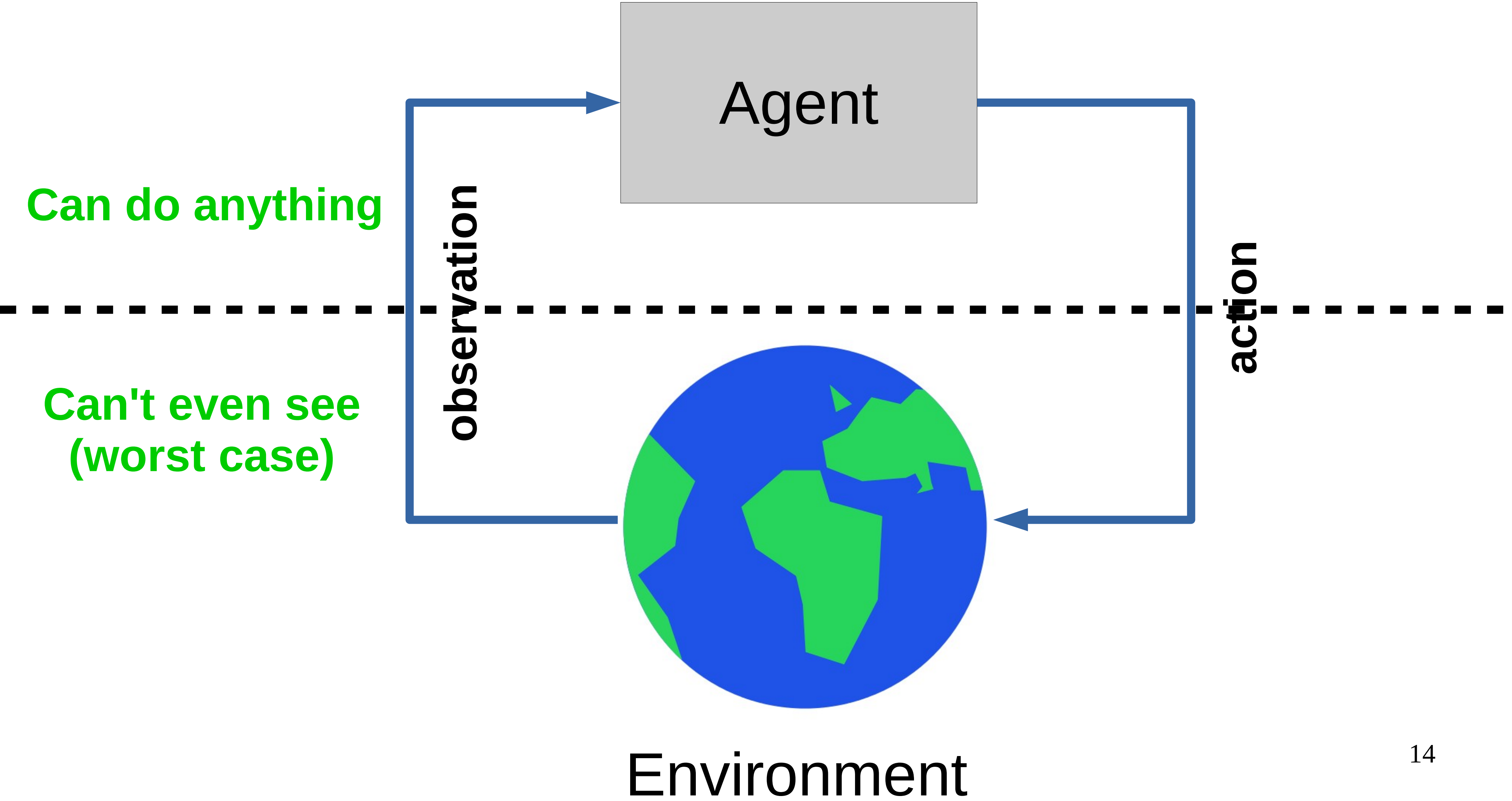


What is: reinforcement learning

What is: reinforcement learning



What is: reinforcement learning



What is: reinforcement learning

- Agent interacts with environment
  - site interacts with user
  - robot interacts with the physical world

- Feedback on agent performance
  - Agent receives immediate reward for his action
  - Usually a real number (more=better)

What is: reinforcement learning

- Agent interacts with environment
  - site interacts with user
  - robot interacts with the physical world

You get to pick actions, not just observe data

- Feedback on agent performance
  - Agent receives immediate reward for his action
  - Usually a real number (more=better)

Not given optimal actions as feedback

Reinforcement learning Vs regular ML

- Algorithm can influence what samples it gets
- Data is not i.i.d.
- Goal is to learn optimal policy
  - (observation→ what to do)

Reinforcement learning Vs regular ML

- Algorithm can influence what samples it gets

  Similar to "active learning"

- Data is not i.i.d.

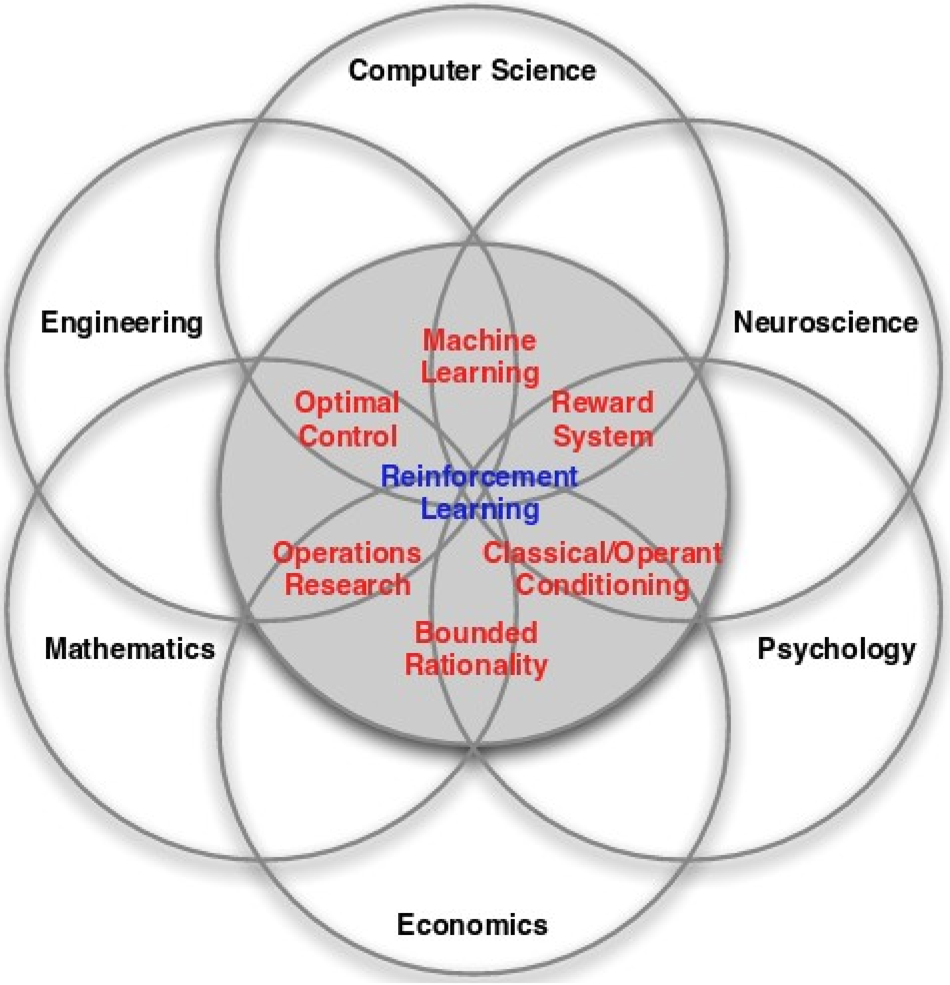
  Many optimization/inference require i.i.d.

- Goal is to learn optimal policy
  - (observation → what to do)

RL can be viewed as supervised learning*

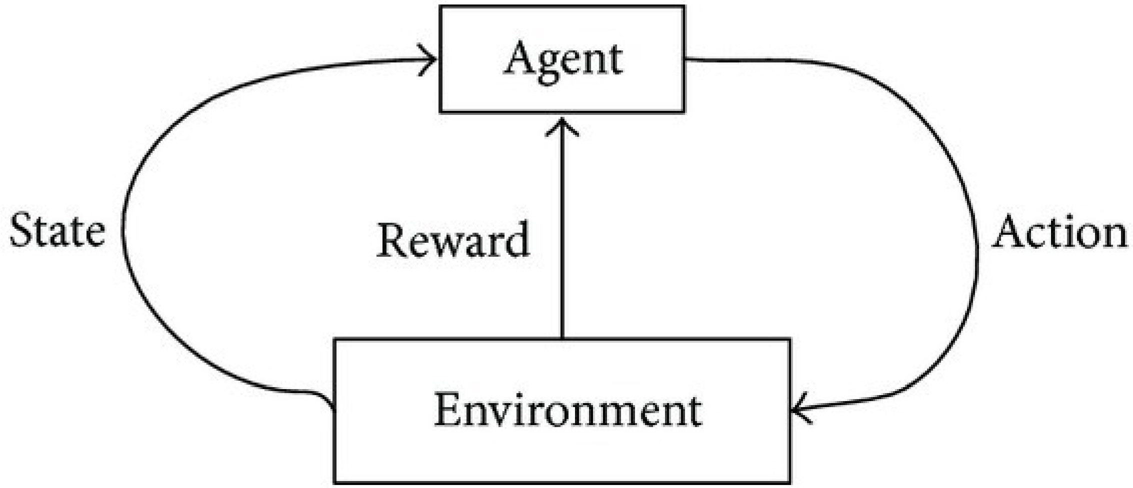
* with some duct tape

Reinforcement learning Vs world



- pic by David Silver

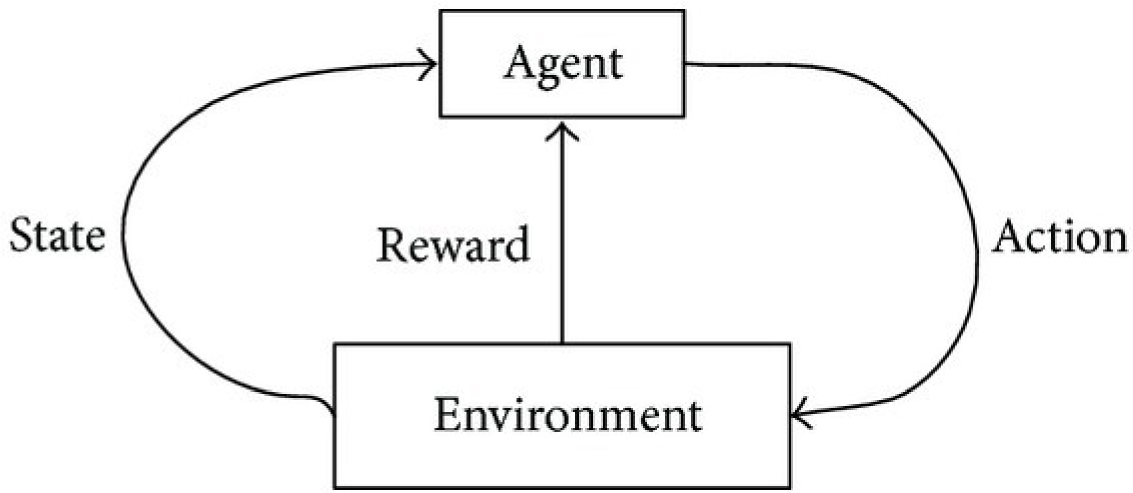
The MDP formalism



Classic MDP(Markov Decision Process)
Agent interacts with environment

- Environment states: $s \in S$
- Agent actions: $a \in A$
- State transition: $P(s_{t+1} | s_t, a_t)$
- Reward: $r_t = r(s_t, a_t)$

The MDP formalism



Classic MDP(Markov Decision Process)
Agent interacts with environment

- Environment states: $s \in S$
- Agent actions: $a \in A$
- State transition: $P(s_{t+1} | s_t, a_t)$
- Reward: $r_t = r(s_t, a_t)$

Markov assumption

Optimal policy

Naive objective:
Total reward

$$R = \sum_t r(s_t, a_t)$$

Deterministic policy:

- Find policy with highest expected reward

$$\pi(s) \to a : E[R] \to max$$

Context: FrozenLake

- A grid world with a goal tile and ice holes

SFFF (S: starting point, safe)
FHFH (F: frozen surface, safe)
FFFH (H: hole, fall to your doom)
HFFG (G: goal, where the frisbee is located)

Quiz: what states, actions and rewards are used?

Model-based RL

- Imagine you have an accurate model of the world
  - e.g. physics model for robot
- You know exactly:
  - $P(s_{t+1}|s_t, a_t)$
  - $r_t = r(s_t, a_t)$
  - Forall $s \in S$ $a \in A$

- How can you get optimal action?

Combinatorial optimization

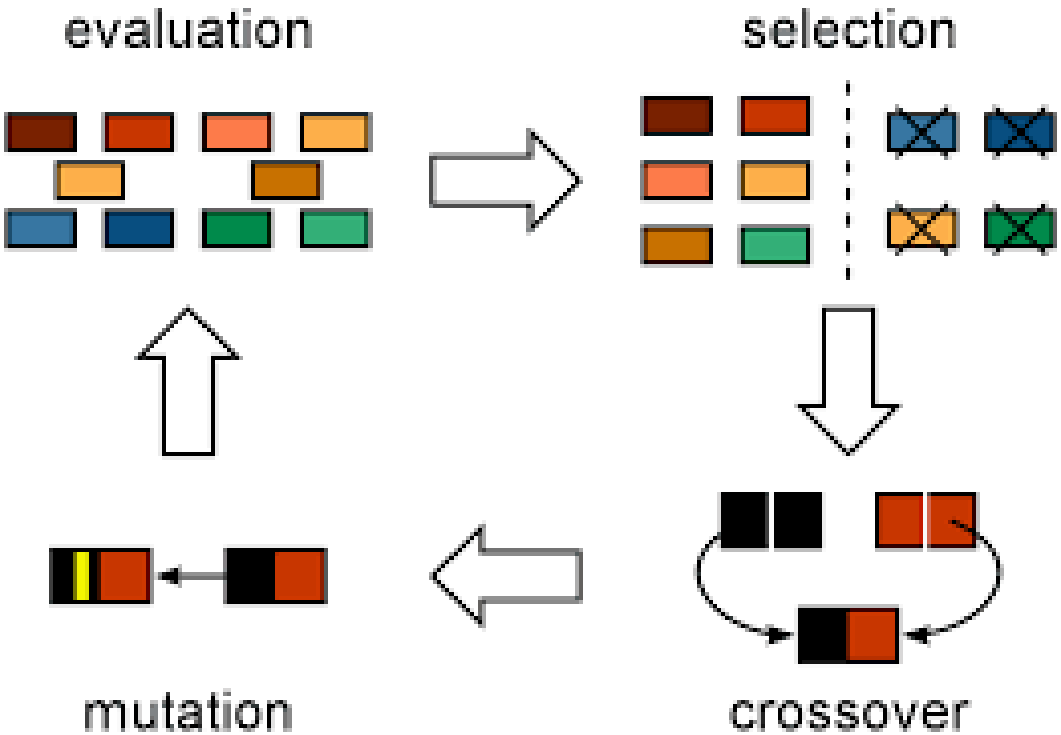
- Maximize score over policy
- No gradient
- Naive solution: iterate over all policies
  - Any problems with that?

Combinatorial optimization

- Maximize score over policy
- No gradient
- Naive solution: iterate over all policies
  - Bizillion candidates
- Efficient algorithms for particular problems

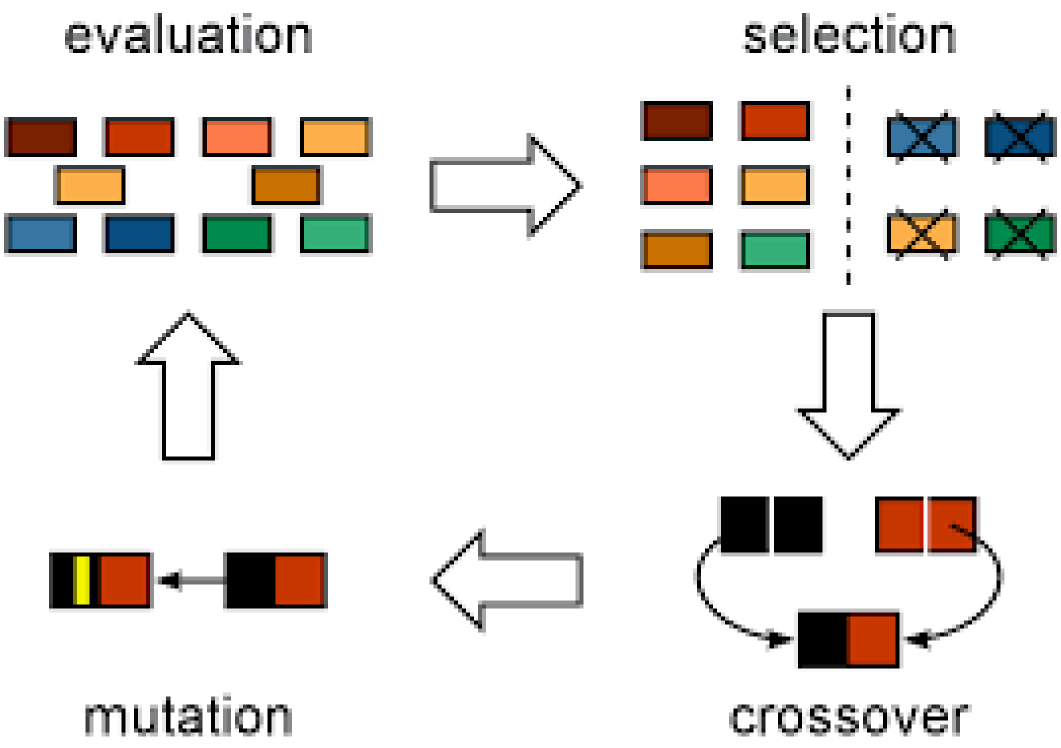
Genetic algorithms

- biologically inspired heuristic
- maintain a population of policies
- reproduce (crossover) → mutate → prune



Genetic algorithms

- biologically inspired heuristic   no guarantees
- maintain a population of policies
- reproduce (crossover) → mutate → prune

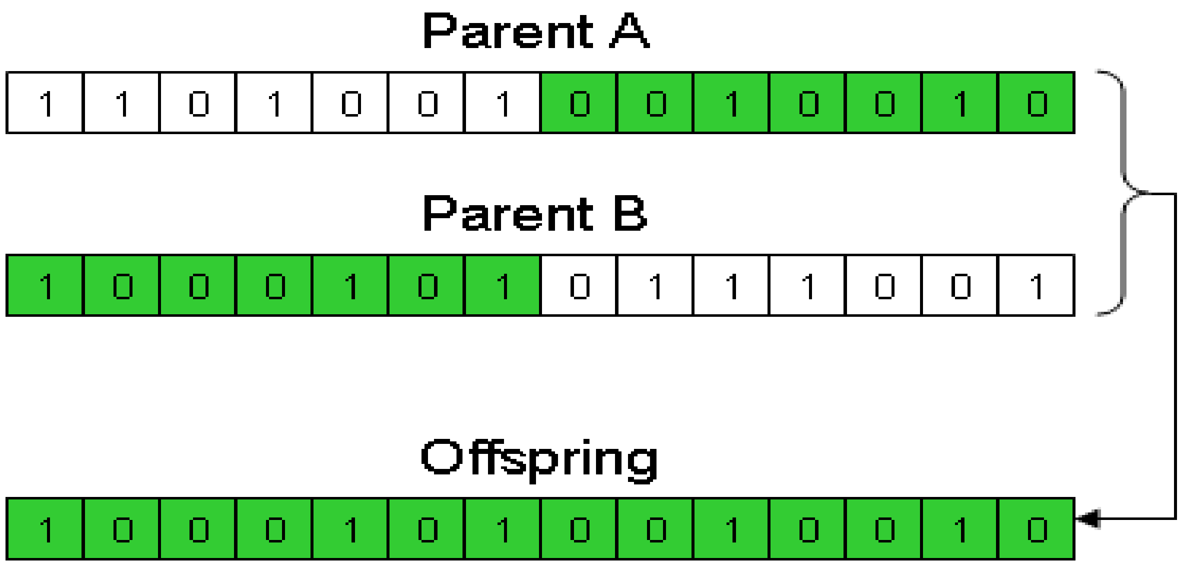


Genetic algorithms

- Keep a pool of N policies
- On each step,
  - take M random pairs and mix them
  - take K random policies and mutate them
  - compute fitness ~ how good each policy is
  - leave only top-N policies with highest fitness

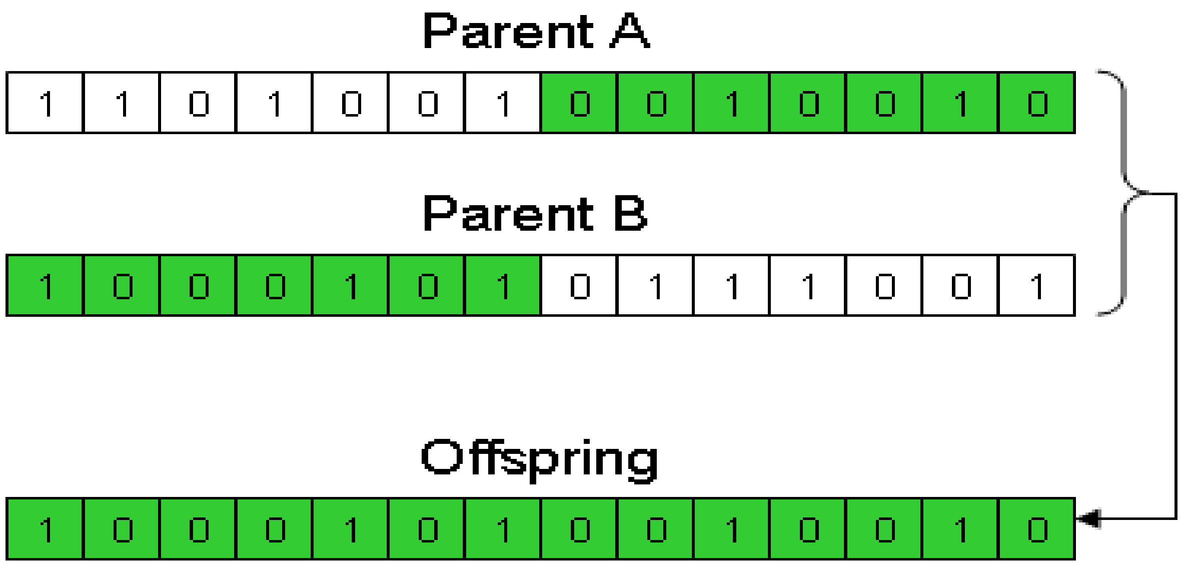
Crossover

- Take 2 random policies (“**parents**”)
- For each state, flip a coin
  - If **heads**, take action from the first parent
  - If **tails**, take action from the **second parent**



Crossover

- Take 2 random policies ("**parents**")
- For each state, flip a coin
  - If **heads**, take action from the first parent
  - If **tails**, take action from the **second parent**



Any other ways to make it?

Problem with GA & similar

- need a full session to start learning
- requires a lot of interaction
  - A lot of crashed robots / simulations

RL can do better*

*see you next lecture