

# RL@Picsart

Episode 0

## Introduction to reinforcement learning



# Terms

- **Ask!**
  - Even if the question feels stupid.
  - Chances are, half of the group is just like you.
  - If it's necessary, interrupt the speaker.
- **Contribute!**
  - Found an error? Got useful link? Ported the seminar to py3 from py2? Answered peer's question in the chat?
  - You're awesome!

<a convenient slide for public survey>

# Supervised learning

Given:

- objects and answers

$$(x, y)$$

- algorithm family

$$a_{\theta}(x) \rightarrow y$$

- loss function

$$L(y, a_{\theta}(x))$$

Find:

$$\theta' \leftarrow \operatorname{argmin}_{\theta} L(y, a_{\theta}(x))$$

# Supervised learning

Given:

- objects and answers
- algorithm family
- loss function

$(x, y)$   
[banner,page], ctr

$a_{\theta}(x) \rightarrow y$   
linear / tree / NN

$L(y, a_{\theta}(x))$   
MSE, crossentropy

Find:

$$\theta' \leftarrow \operatorname{argmin}_{\theta} L(y, a_{\theta}(x))$$

# Supervised learning

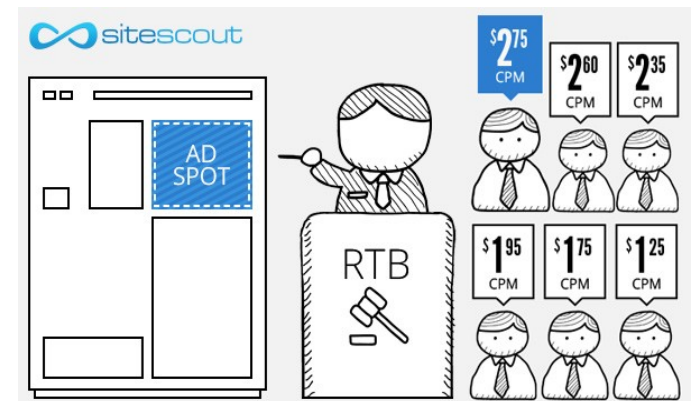
Great... except if we have no reference answers

# 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)

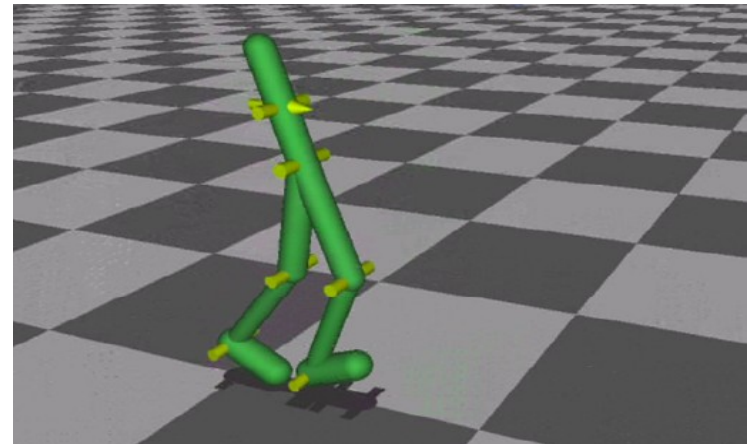
Great... except if we have no reference answers

## We have:

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

## We want:

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



Ideas?



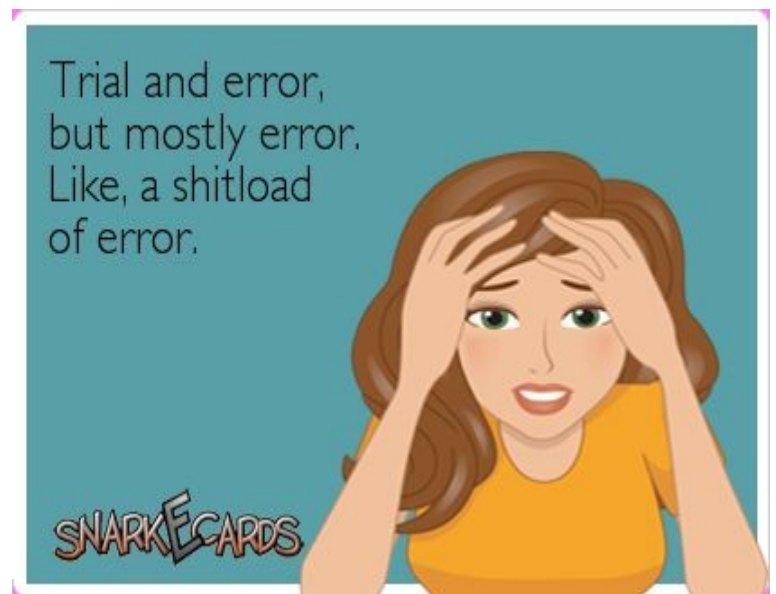
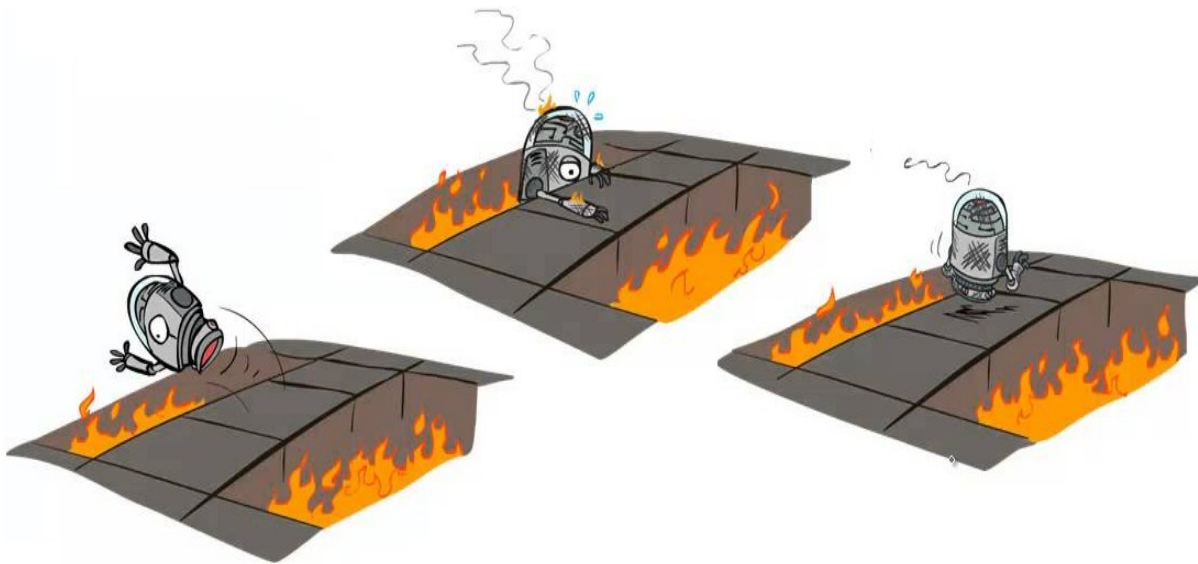
# Duct tape approach



# 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

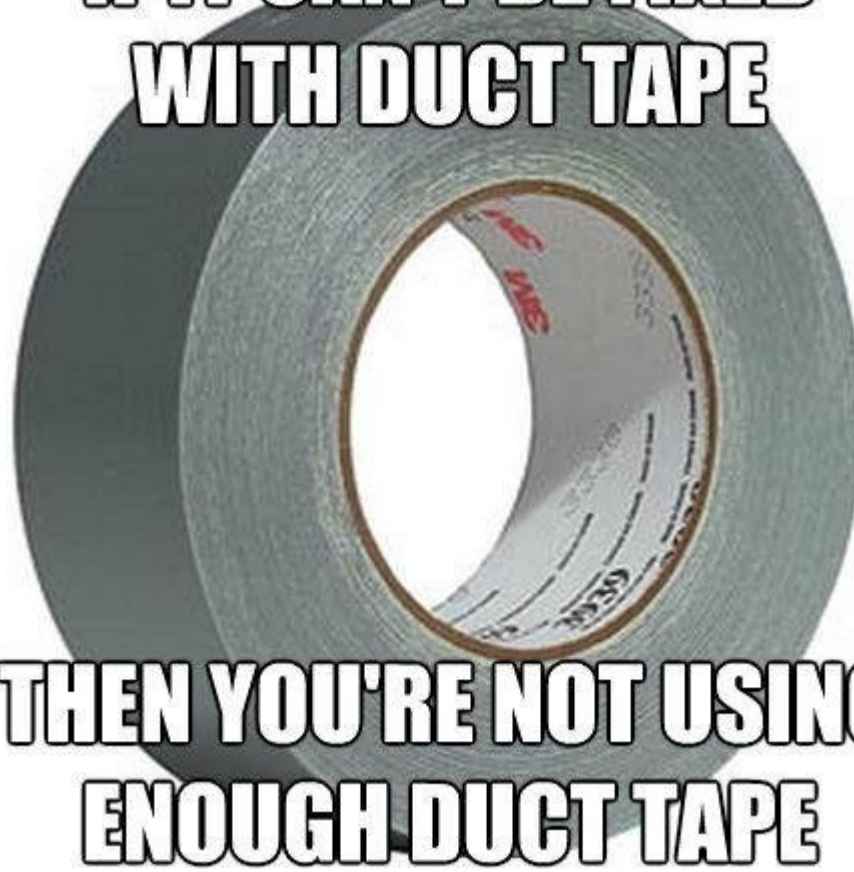
# Duct tape approach

## **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

**IF IT CAN'T BE FIXED  
WITH DUCT TAPE**



**THEN YOU'RE NOT USING  
ENOUGH DUCT TAPE**

zipmeme

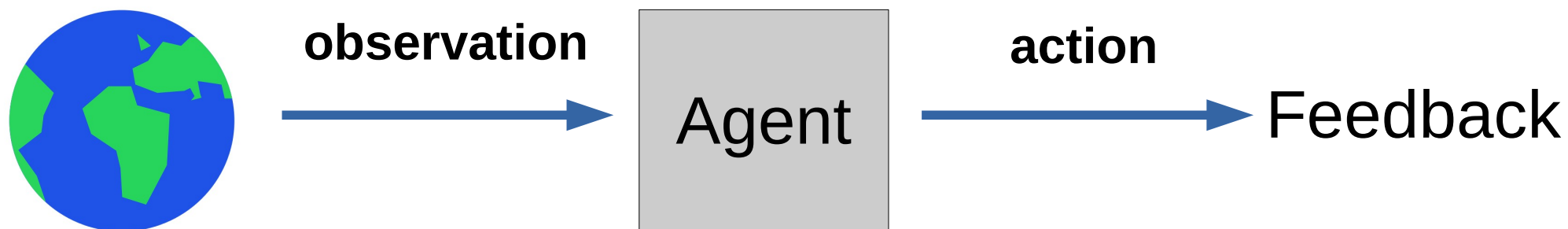
# What is: reinforcement learning

**STAND BACK**



**I'M GOING TO TRY  
SCIENCE**

# What is: bandit

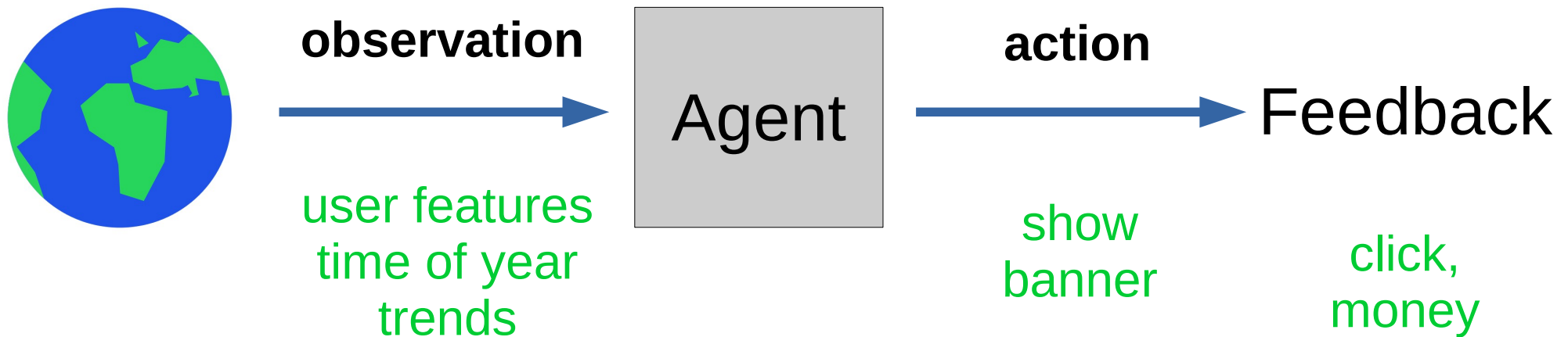


## Examples:

- banner ads (RTB)
- recommendations
- medical treatment

**Trivia:** what's observation, action and feedback in the banner ads problem?

# What is: bandit

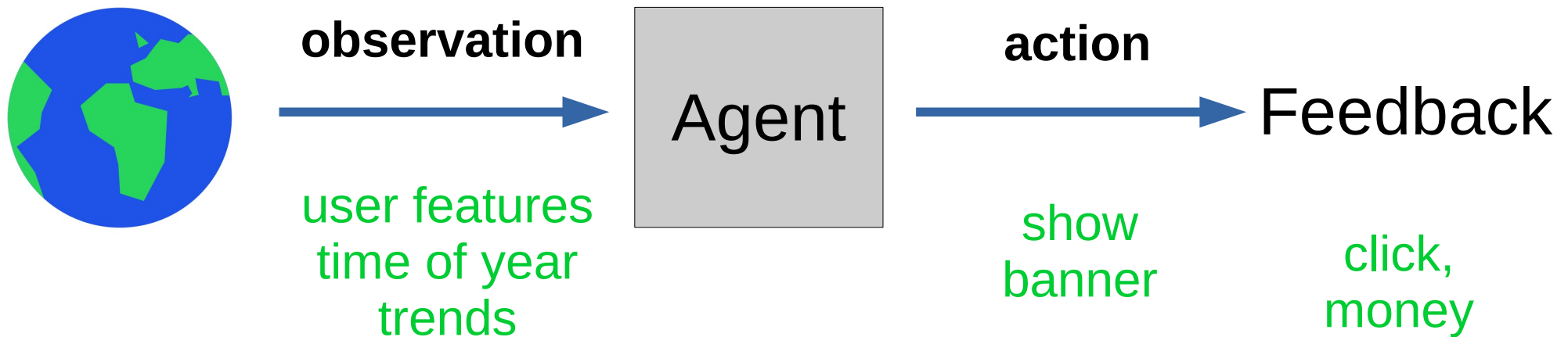


## Examples:

- banner ads (RTB)
- recommendations
- medical treatment

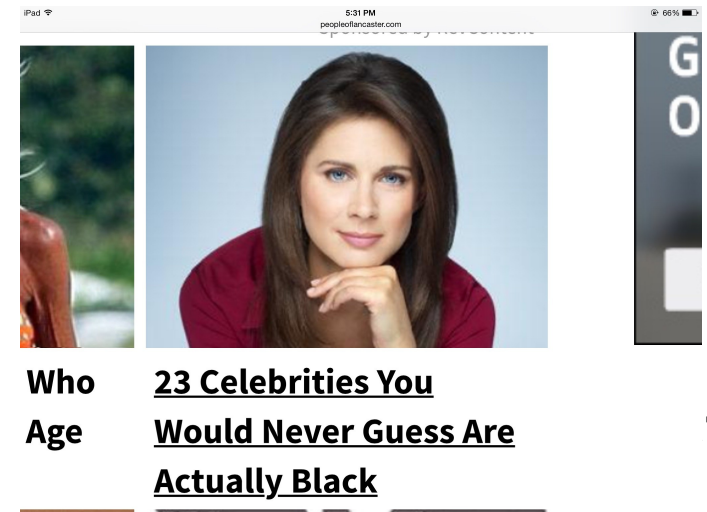


# What is: bandit

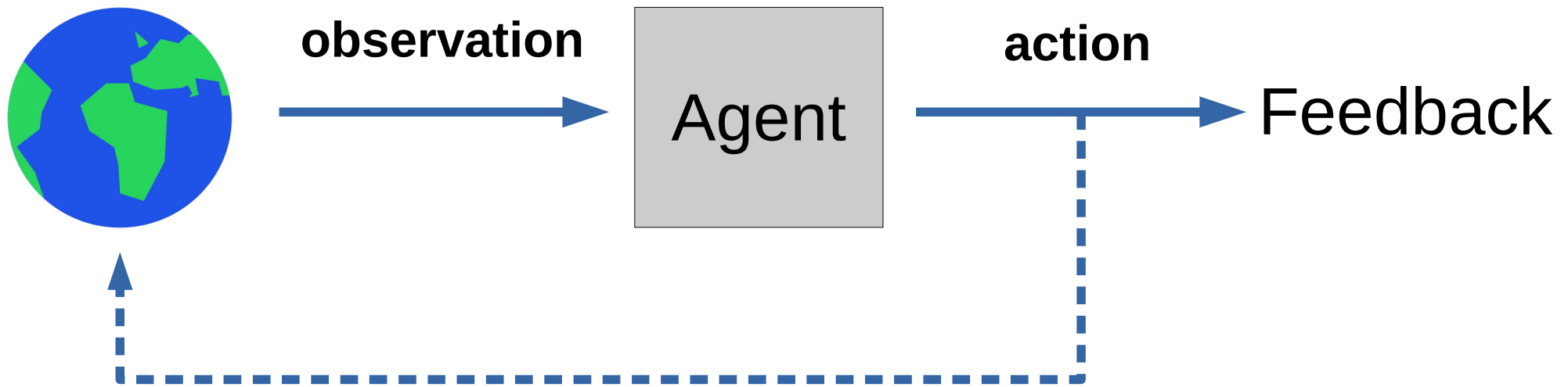


**Trivia:** You're Yandex/Google/Youtube. There's a kind of banners that would have great click rates: the “clickbait”.

Is it a good idea to show clickbait?

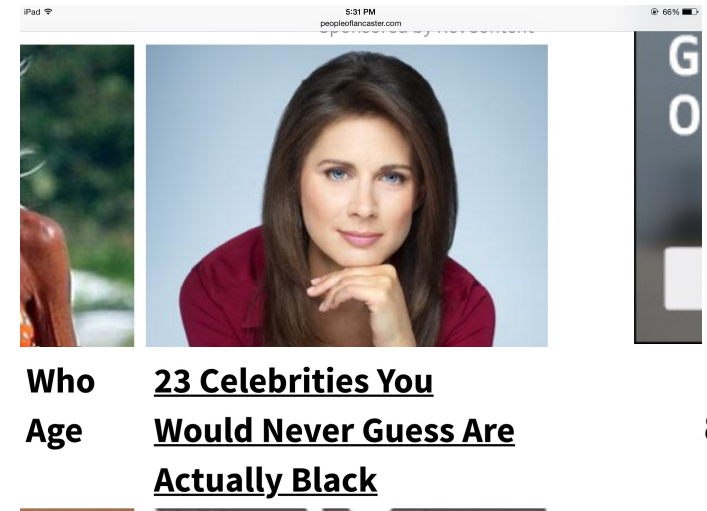


# What is: bandit

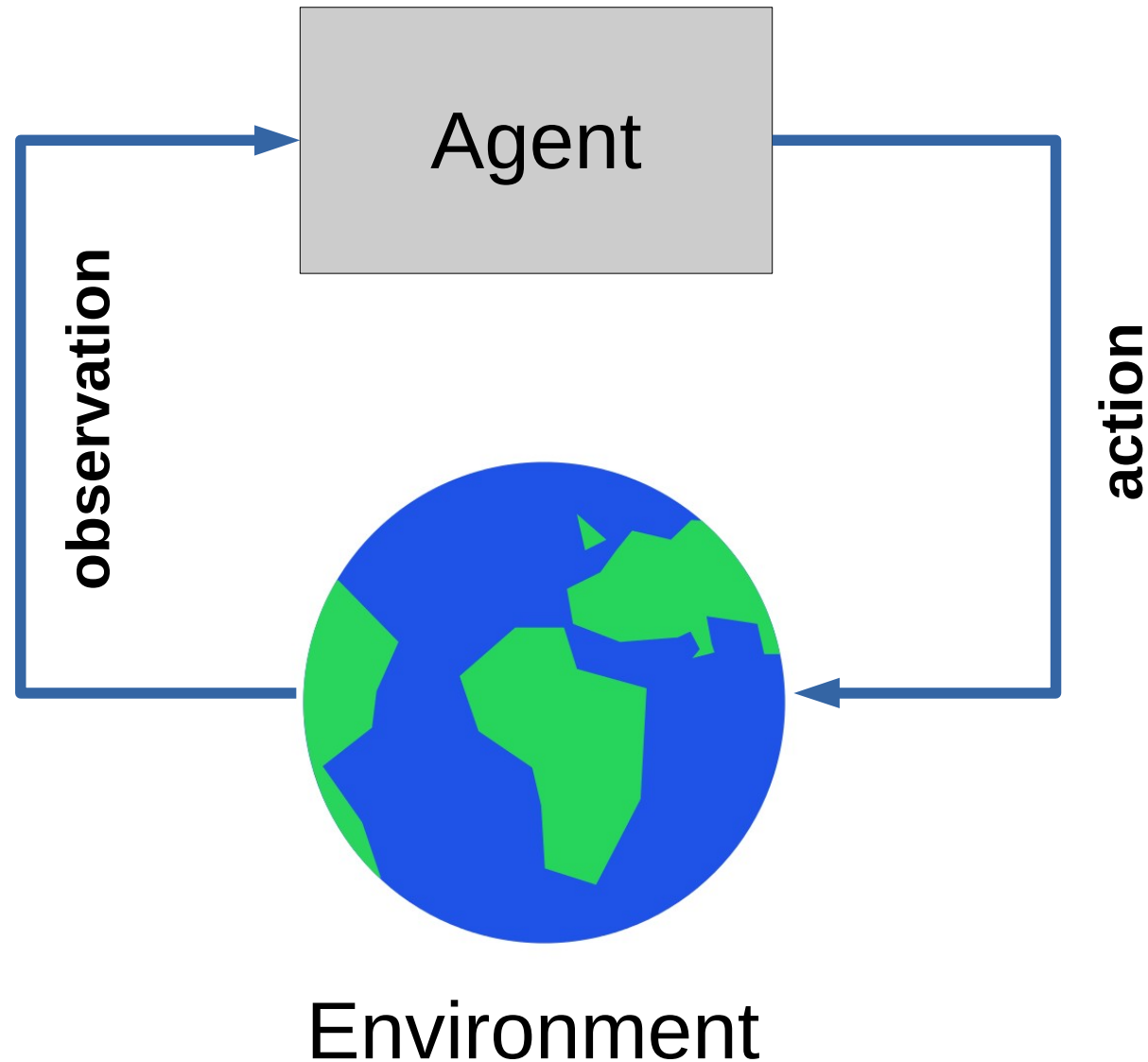


**Trivia:** You're Yandex/Google/Youtube. There's a kind of banners that would have great click rates: the “clickbait”.

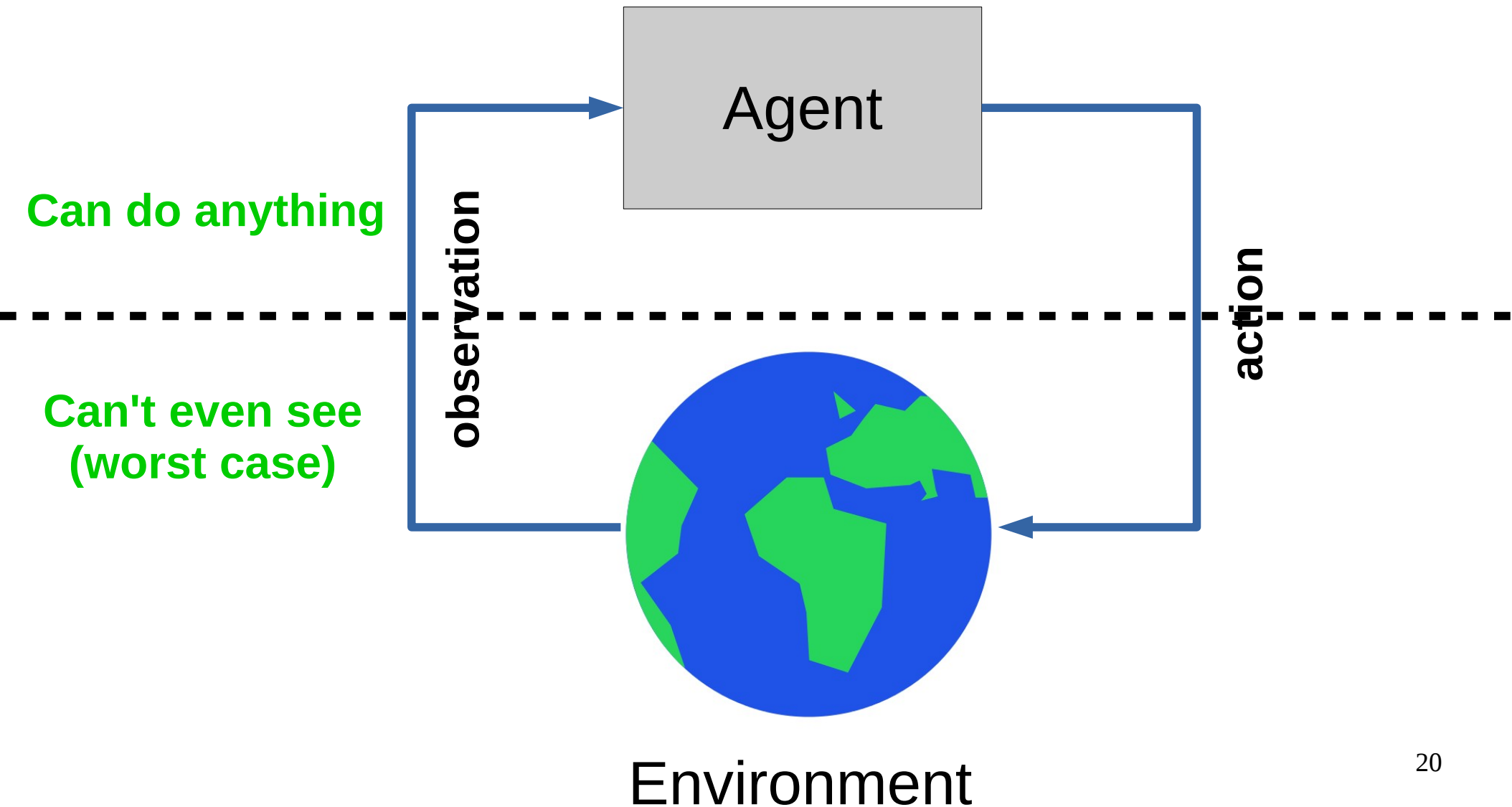
Is it a good idea to show clickbait?  
**No**, noone will trust you after that!



# What is: decision process



# What is: decision process



# What is: reinforcement learning

- Agent interacts with environment
  - site interacts with user
  - robot interacts with the physical world
- Feedback on agent performance
  - Agent receives feedback on **his** performance
  - 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 feedback on **his** performance
  - 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  $\rightarrow$  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)

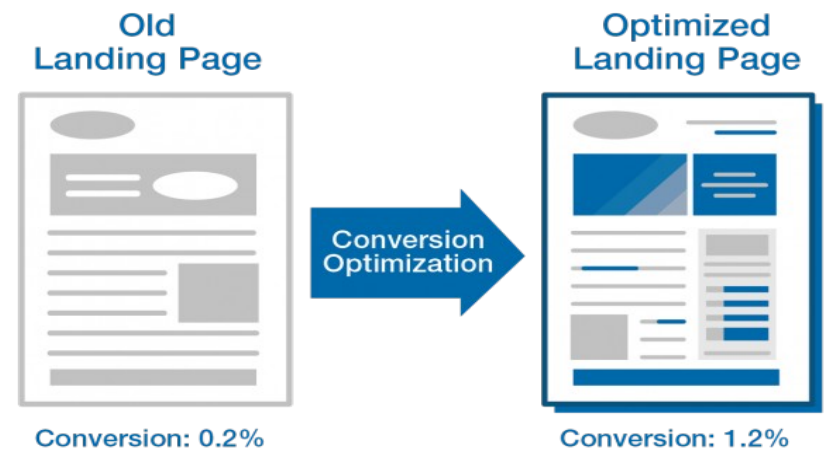
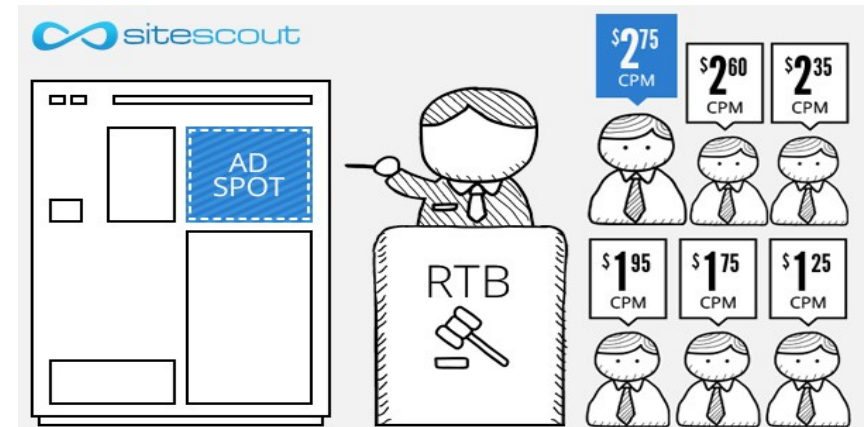
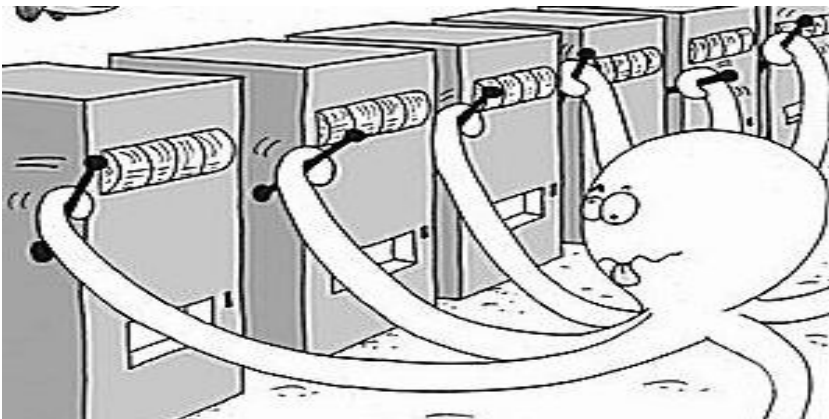
RL can be viewed as supervised learning\*

\* with some duct tape



# Reality check: web

- **Cases:**
  - Pick ads to maximize profit
  - Design landing page to maximize user retention
  - Recommend items to users
- **Example**
  - Observation – user features
  - Action – show banner #i
  - Feedback – did user click?



# Reality check: dynamic systems



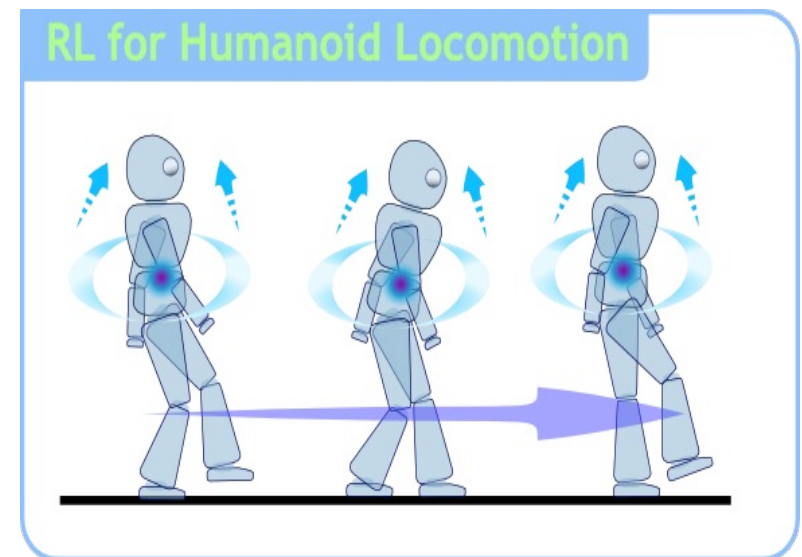
# Reality check: MOAR

- **Cases:**

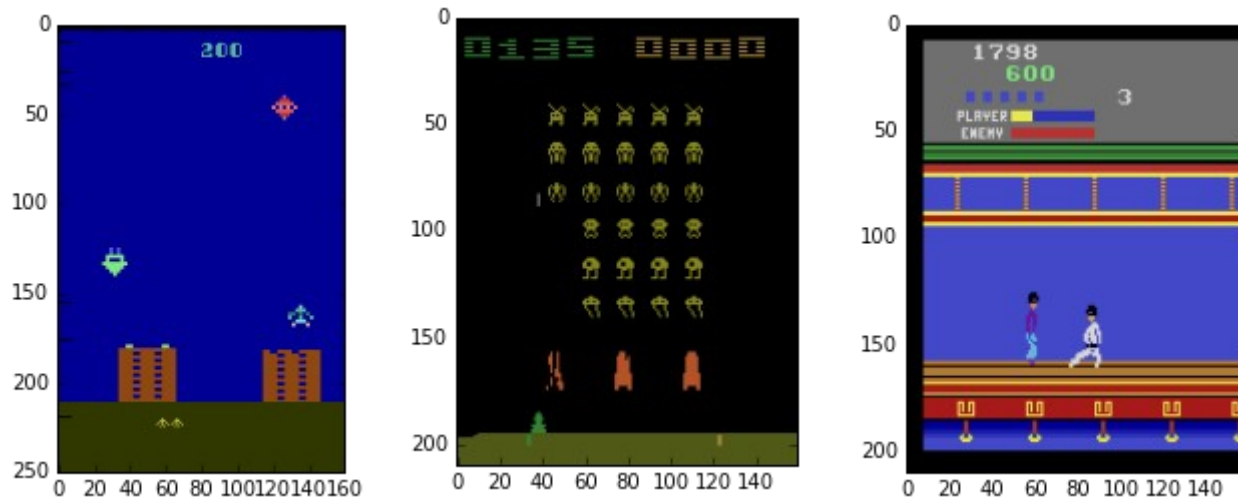
- Robots
- Self-driving vehicles
- Pilot assistant
- More robots!

- **Example**

- Observation: sensor feed
- Action: signals to motors
- Feedback: how far did it move forward before falling



# Reality check: videogames



- **Trivia:** What are observations, actions and feedback?

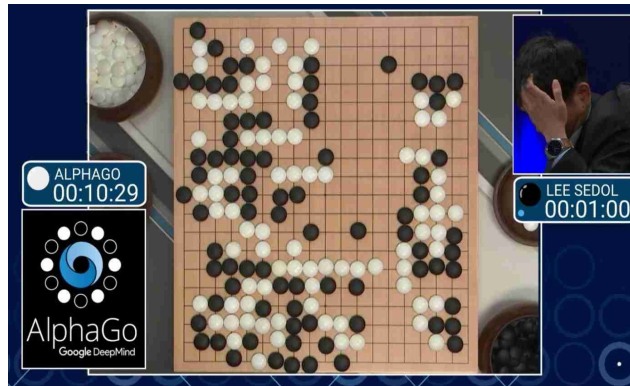


# Other use cases

- Personalized medical treatment



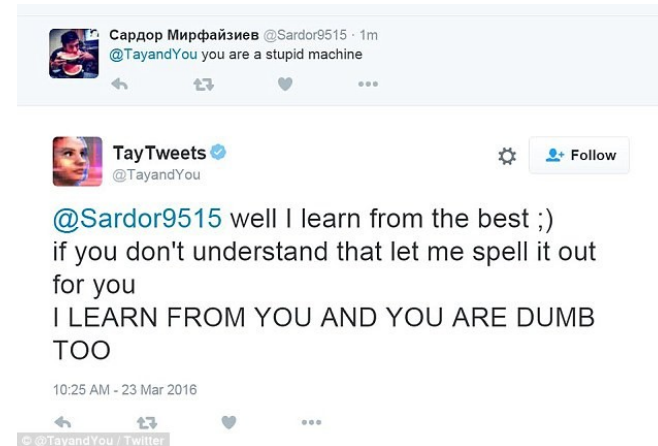
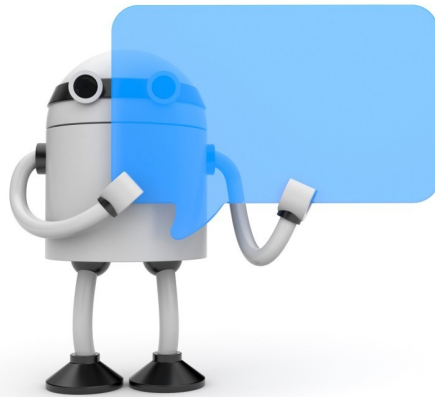
- Even more games (Go, chess, etc)



- **Trivia:** What are observations, actions and feedback?

# Other use cases

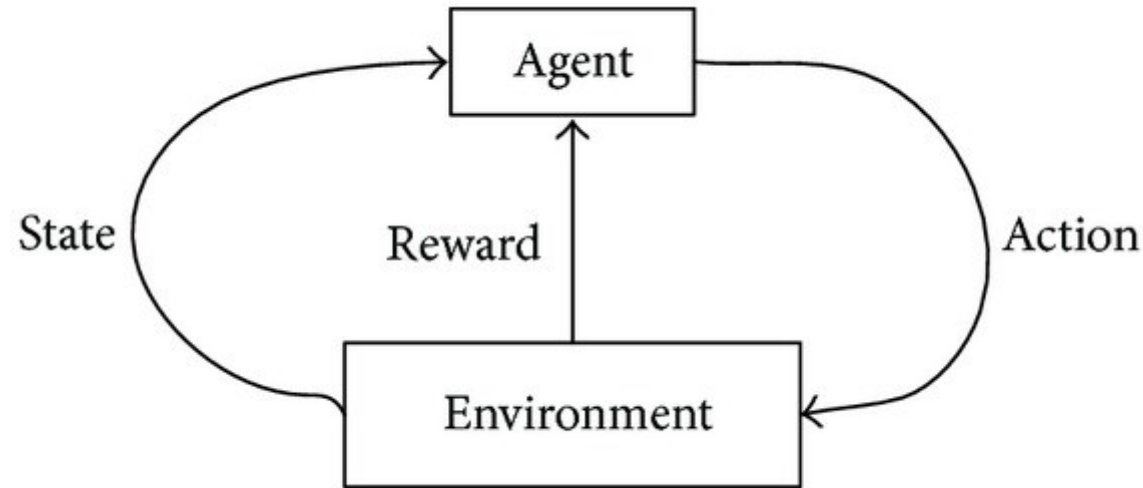
- Conversation systems (additional goals)



- Portfolio management (aka asset allocation)



# The MDP formalism

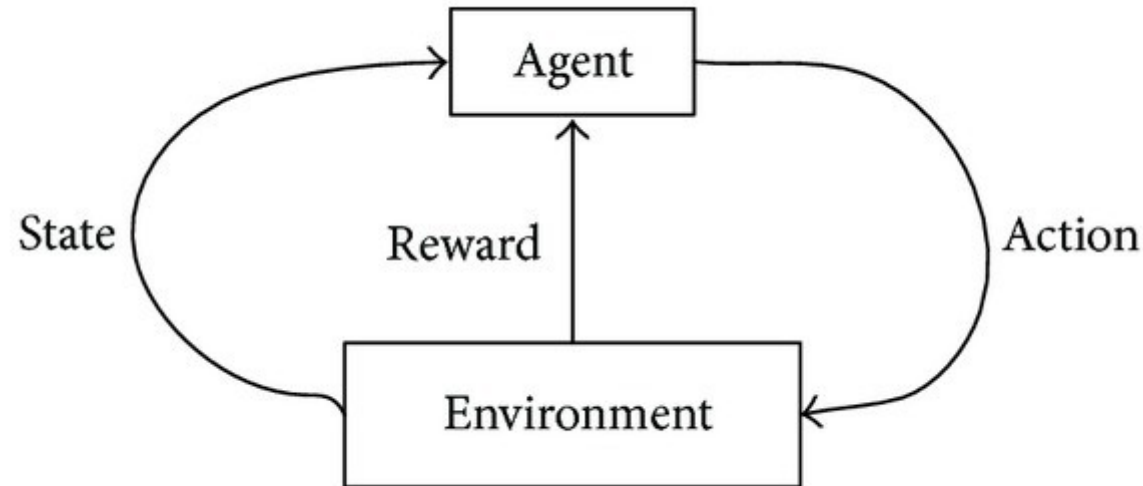


Classic MDP(Markov Decision Process)

Agent interacts with environment

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

# The MDP formalism



Classic MDP(Markov Decision Process)

Agent interacts with environment

- Environment states:  $s \in S$

- Agent actions:  $a \in A$

- State transition:  $P(s_{t+1}|s_t, a_t)$

Markov  
assumption



# Optimal policy (Monte-carlo)



- Naive objective:  $R(z)$

$$z = [s_0, a_0, s_1, a_1, s_2, a_2, \dots, s_n, a_n]$$

Deterministic policy:

- Find policy with highest expected reward

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

# Context: FrozenLake

- A grid world with a goal tile and ice holes

<b>SFFF</b>	(S: starting point, safe)
<b>FHFH</b>	(F: frozen surface, safe)
<b>FFFH</b>	(H: hole, fall to your doom)
<b>HFFG</b>	(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:
  - $P(s_{t+1}|s_t, a_t)$
  - $R(z)$
  - For all  $s \in S$   $a \in A$
- How can you get optimal action?

# Combinatorial optimization

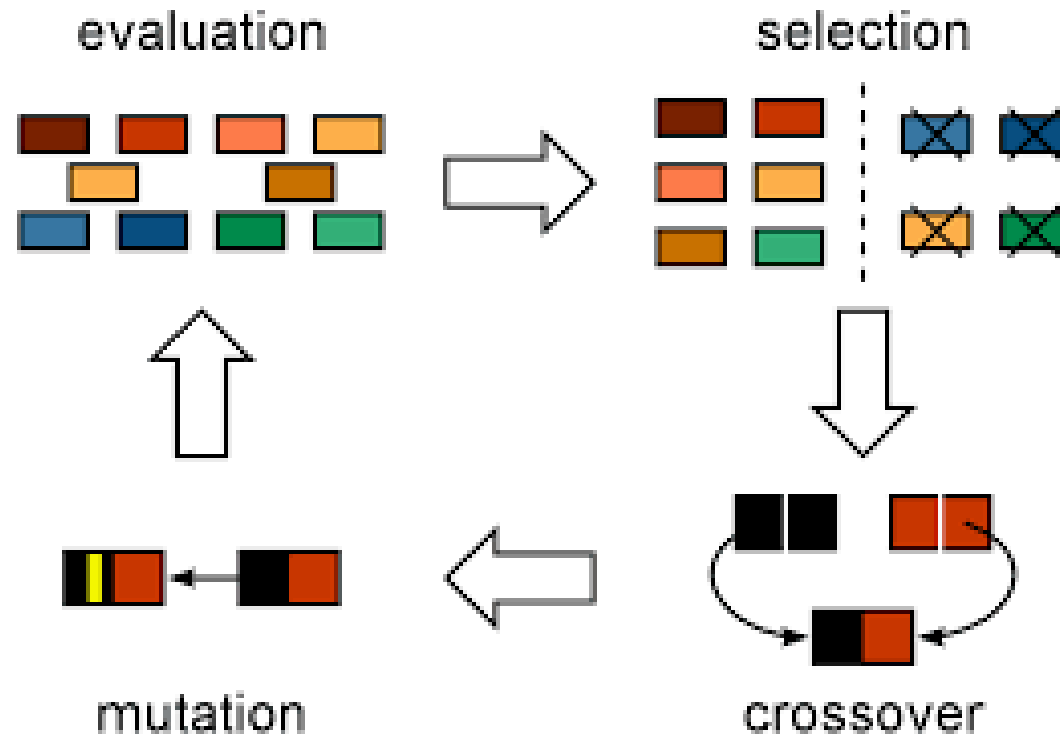
- Maximize score over policy
- No gradient
- Naive solution: iterate over all policies
  - Any problems with that?

# Combinatorial optimization

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

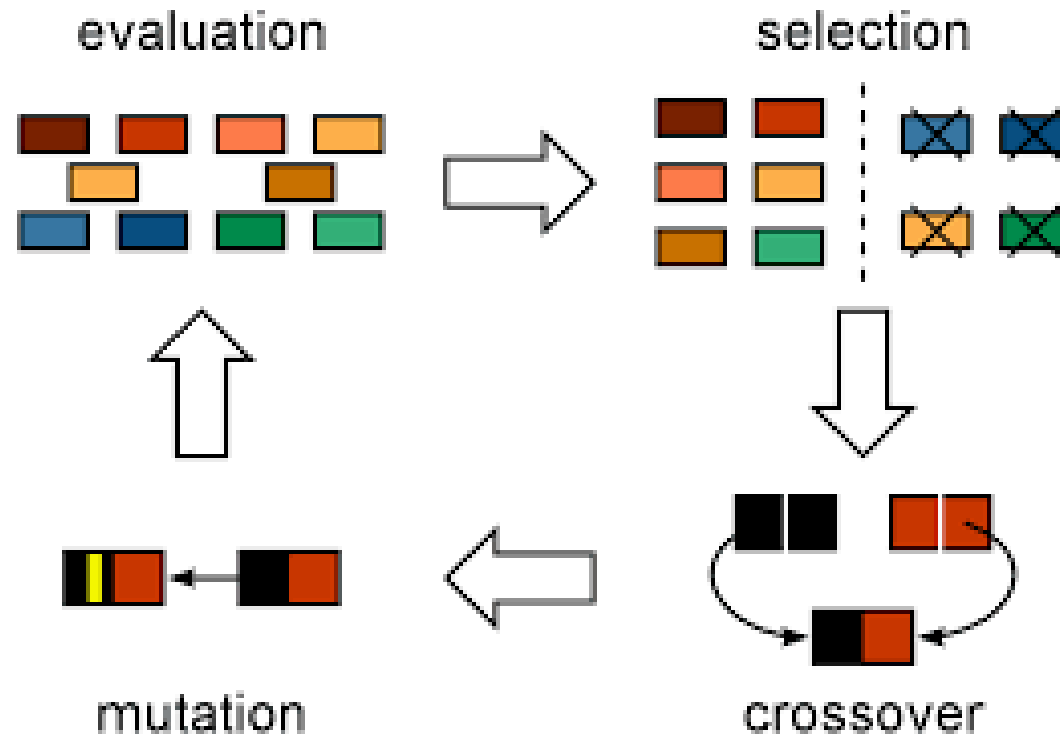
# Genetic algorithms

- 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



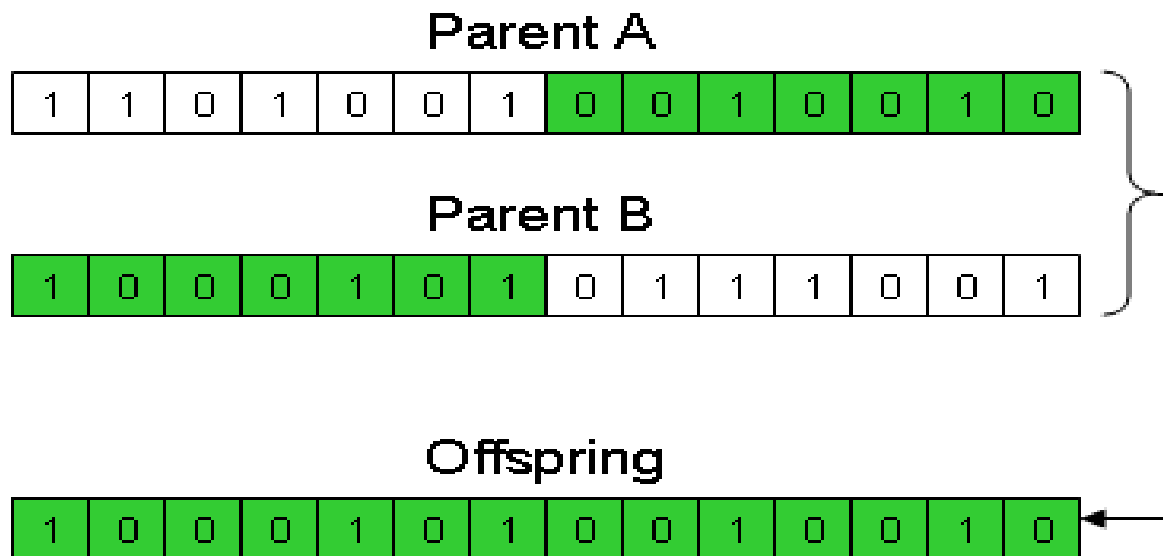
# Genetic algorithms

- 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  $\sim$  how good each policy is
  - leave only top- $N$  policies with highest fitness



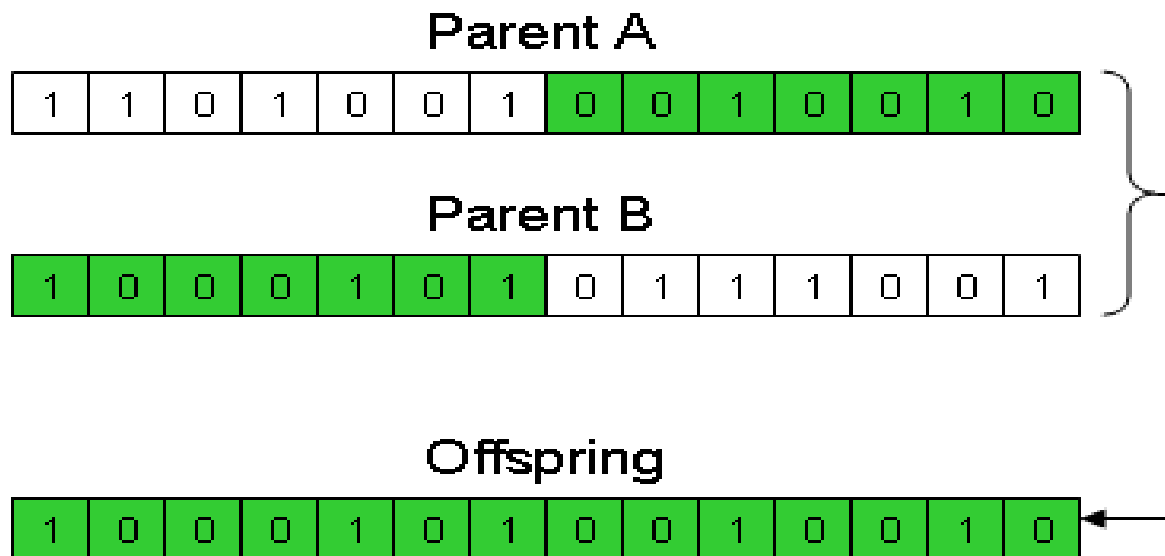
# Crossover

- 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 crossover?**

# Genetic Algorithms

- Pros
  - It sorta sometimes works
  - You get to feel like a god!
- Cons
  - Convergence not guaranteed
  - Requires a lot of samples
  - A lot of parameter tuning
  - Hard to scale on large state spaces
- Moar
  - Differential evolution, Ant colony algorithm, ...

# To be continued...

- Next time

Stochastic optimization

Crossentropy Method in detail

Approximate policy with neural nets...

or RandomForestClassifier :)

# About the course

- Practical skills first
  - Methods that scale to real life problems
- A **lot** of coding labs (11+)
  - Simple “default” tasks with bonus opportunities.
  - You don't have to do all of them, see *grading*
- Recent discoveries
  - Articles from '16, maybe '17
- **Not** covering planning (true  $\exists$  model-based)
- **Brief** on monte-carlo RL and bandits