# 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 $a_\theta(x) \to y$
- loss function $L(y, a_\theta(x))$

Find:

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

# Supervised learning

Given:

- objects and answers $(x, y)$

  [banner,page], ctr

- algorithm family $a_\theta(x) \to y$

  linear / tree / NN

- loss function $L(y, a_\theta(x))$

  MSE, crossentropy

Find:

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

# Supervised learning

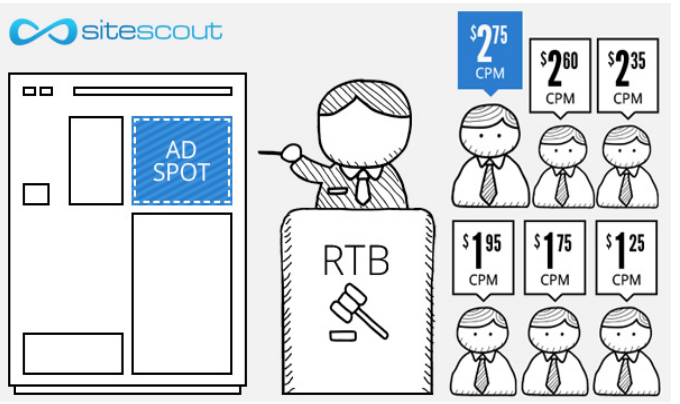
Great... except if we have no reference answers

# 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

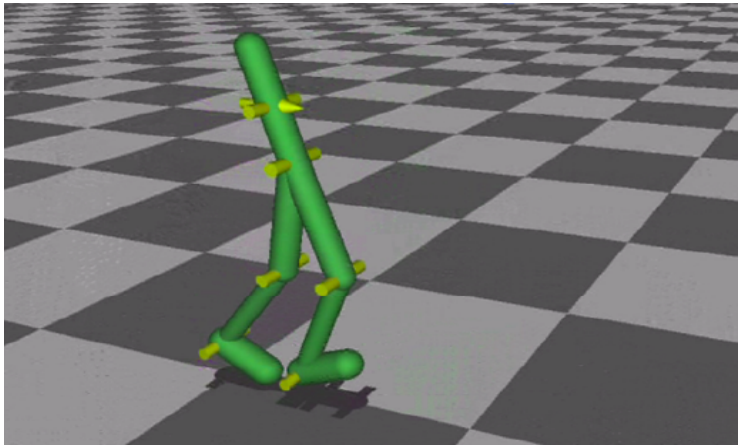
**Ideas?**

# 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 :)

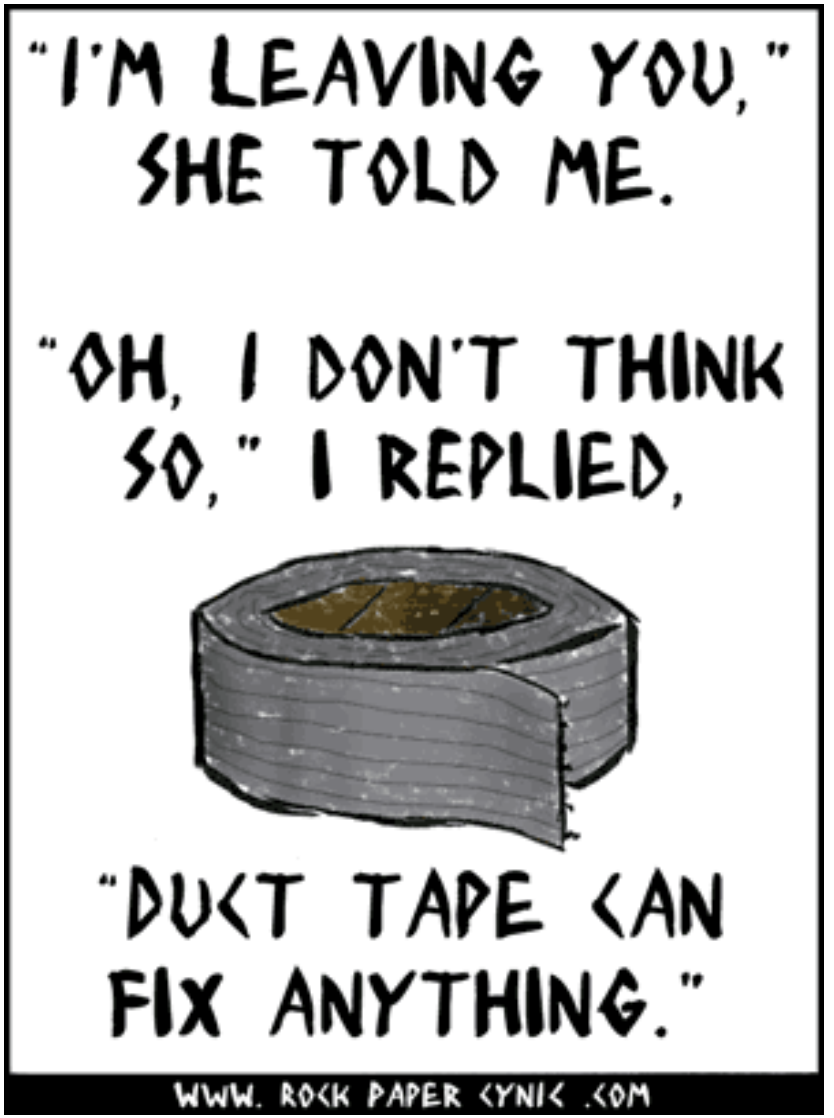


**We want:**

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

**Ideas?**

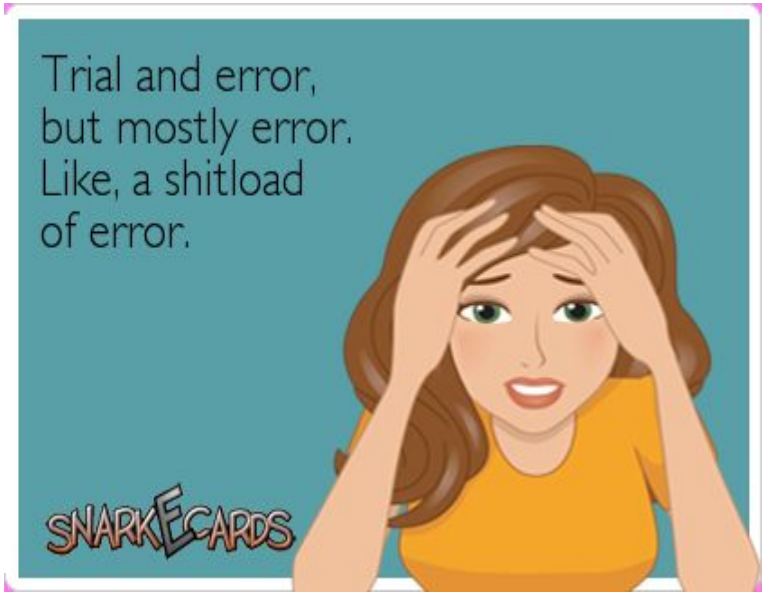
# Duct tape approach



# 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

# What is: reinforcement learning

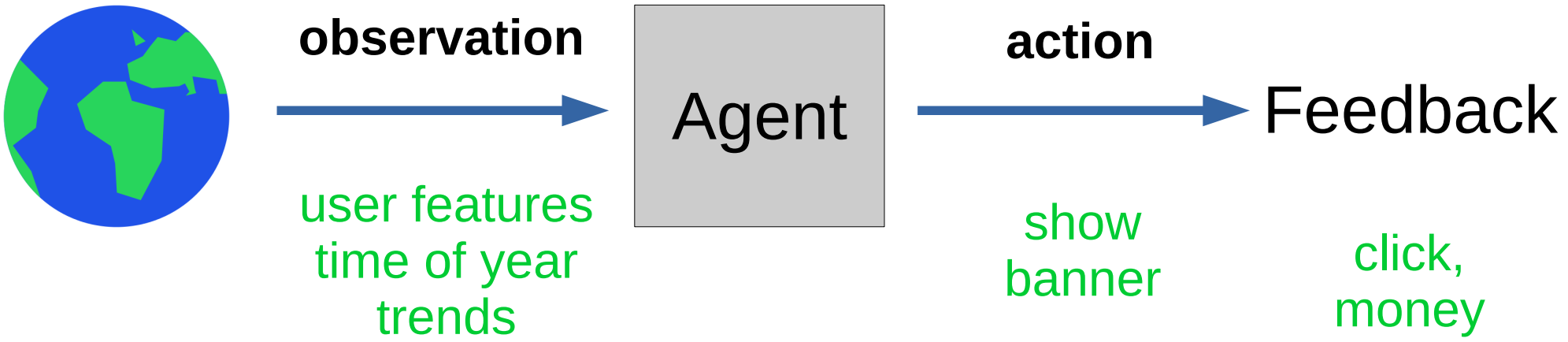
# What is: bandit


**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

**observation**

user features
time of year
trends

Agent

**action**

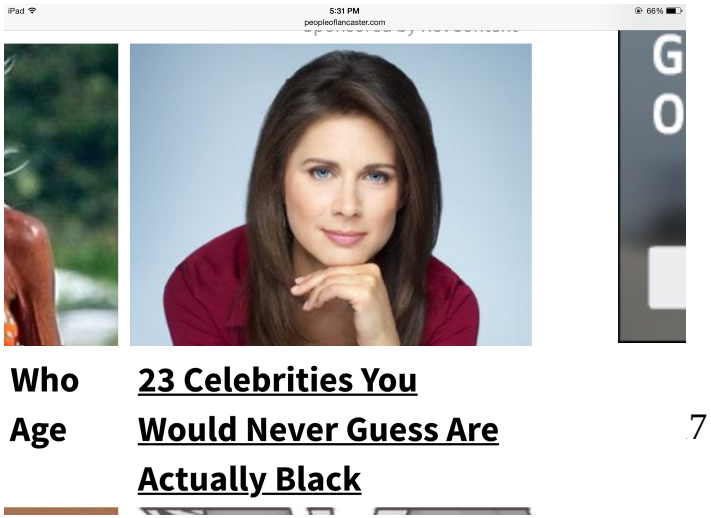
show
banner

Feedback

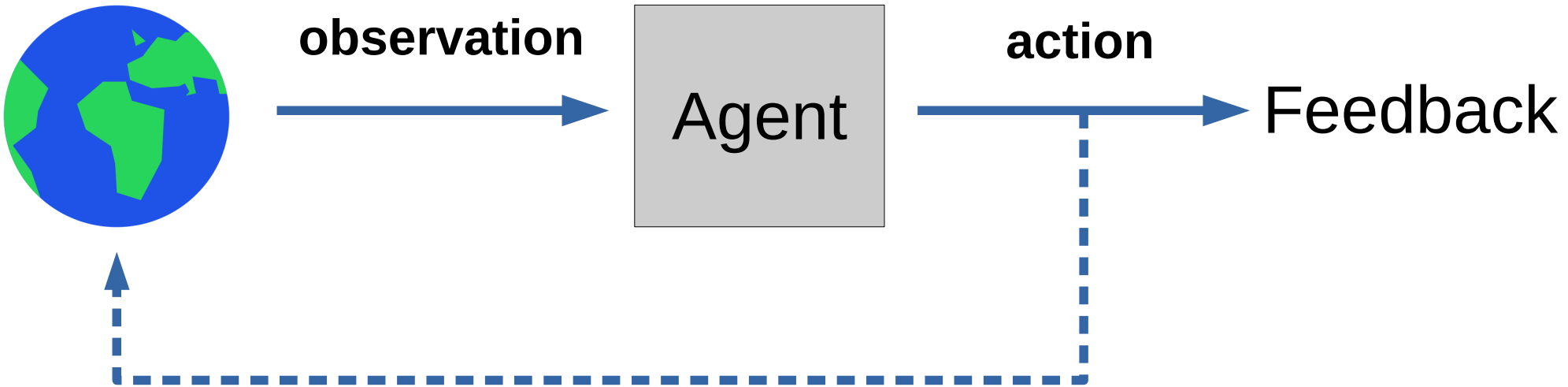
click,
money

**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?

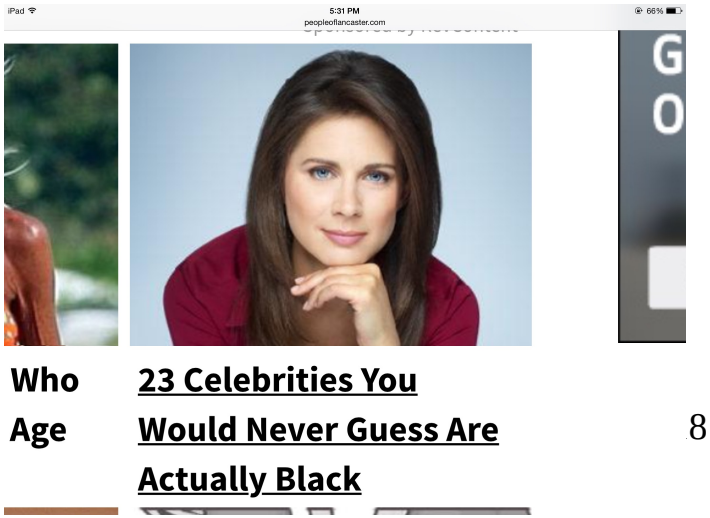


# What is: bandit

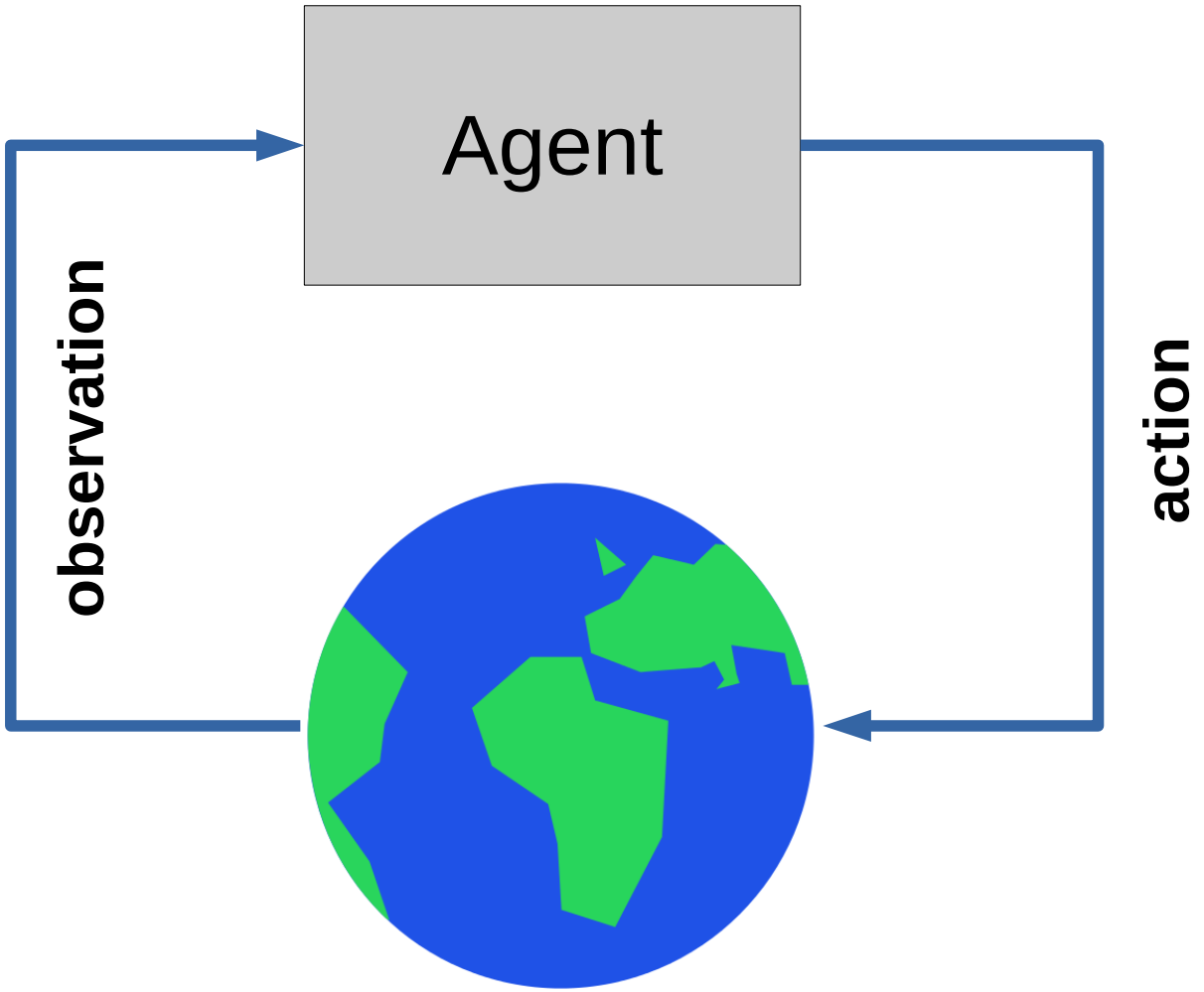



**Trivia:** You're Yandex/Google/Youtube. There's a kind of banners that would have great click rates: the “clickbait”.

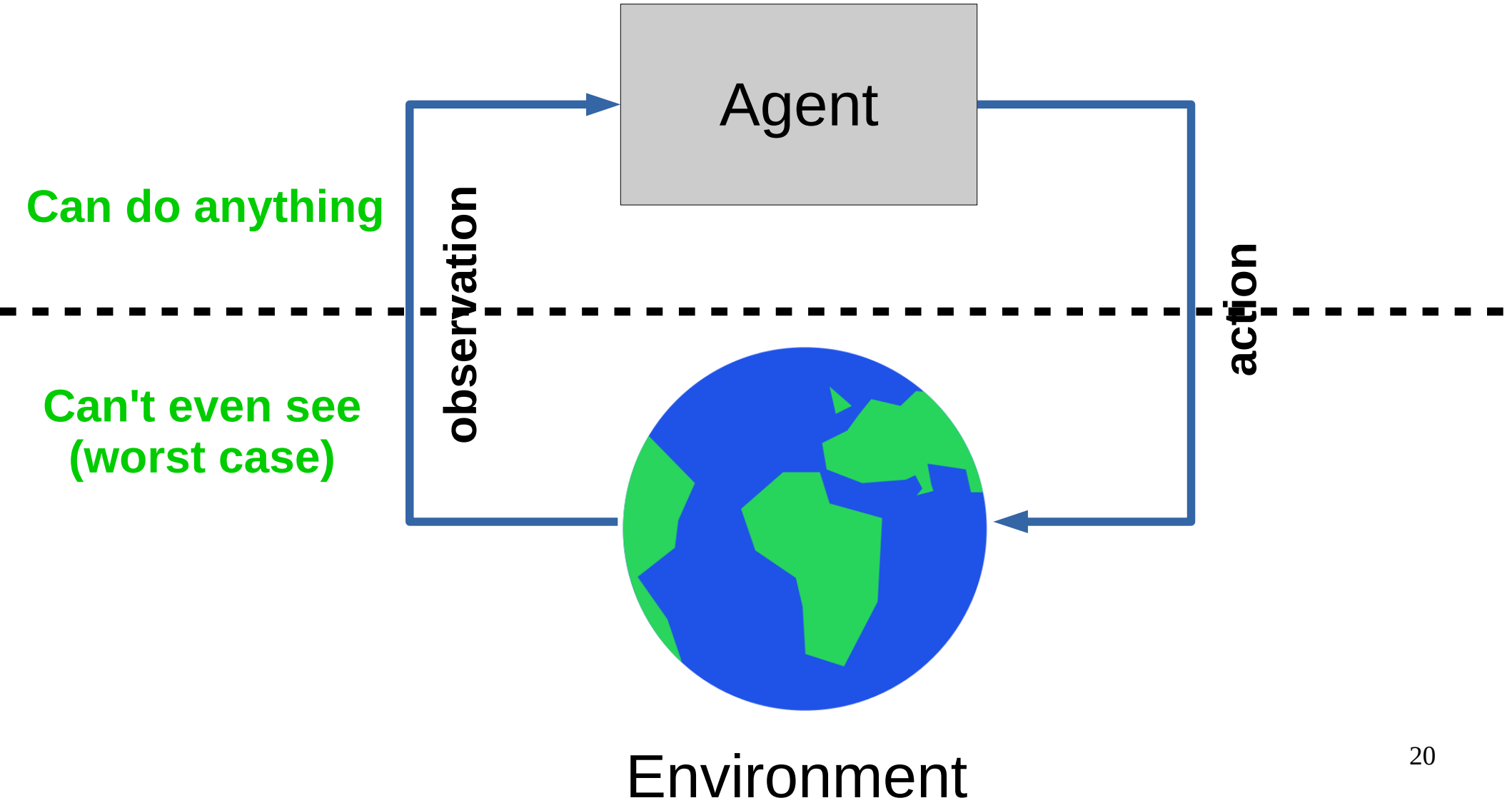
Is it a good idea to show clickbait?
**No**, no one will trust you after that!



# 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 **her** 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 → 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)

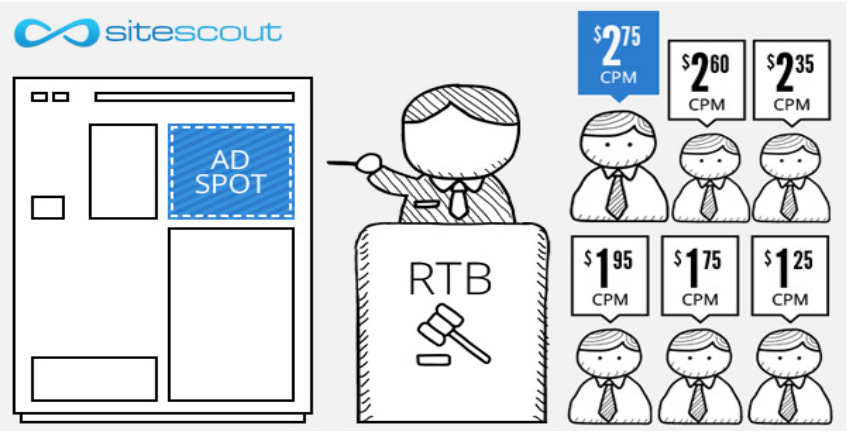
  RL can be viewed as supervised learning*

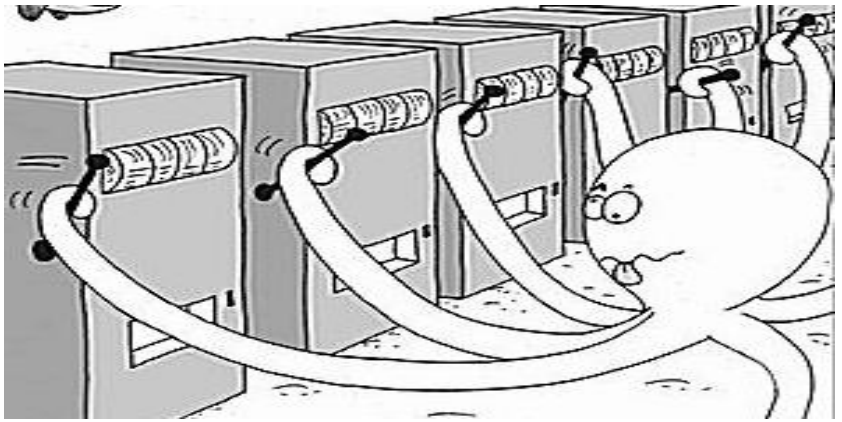
  * with some duct tape

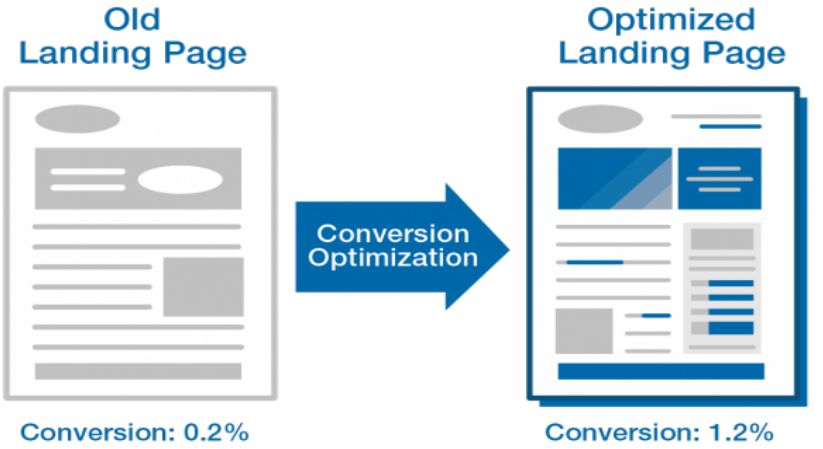
# Reality check: web

- **Cases:**
  - Pick ads to maximize profit
  - Design landing page to maximize user retention
  - Recommend items to users

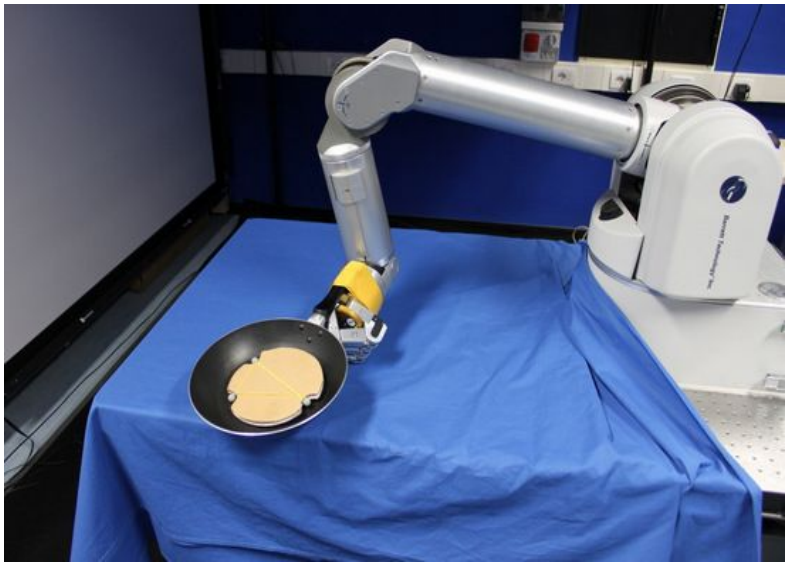
- **Example**
  - Observation – user features
  - Action – show banner #i
  - Feedback – did user click?





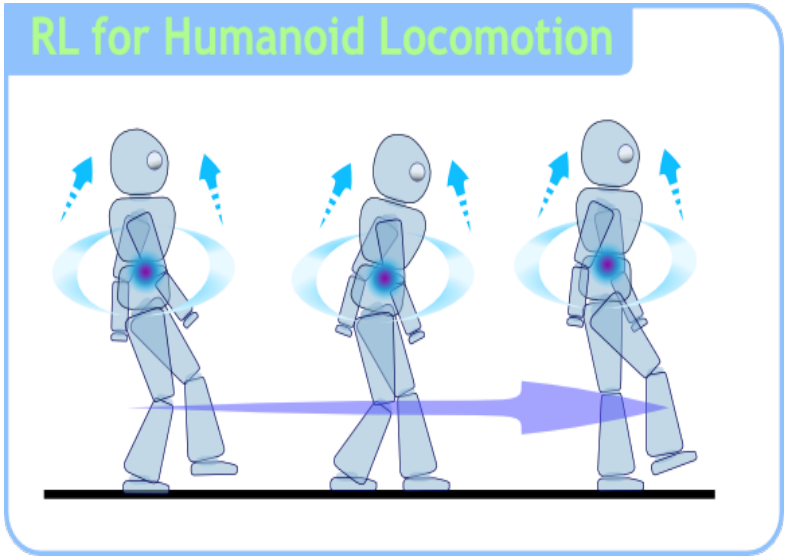
# Reality check: dynamic systems

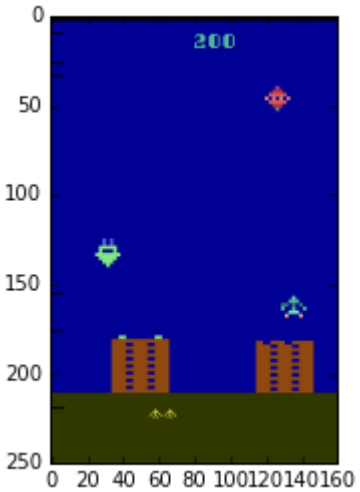
# Reality check: MOAR

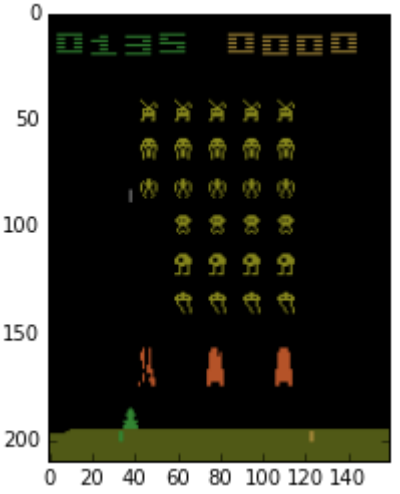
- **Cases:**
  - Robots
  - Self-driving vehicles
  - Pilot assistant
  - More robots!

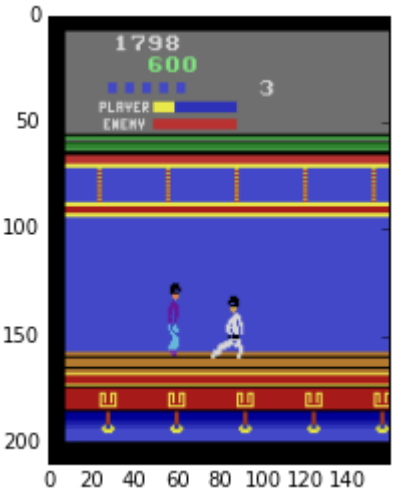
- **Example**
  - Observation: sensor feed
  - Action: signals to motors
  - Feedback: how far did it move forward before falling



# ~~Reality~~ check: videogames



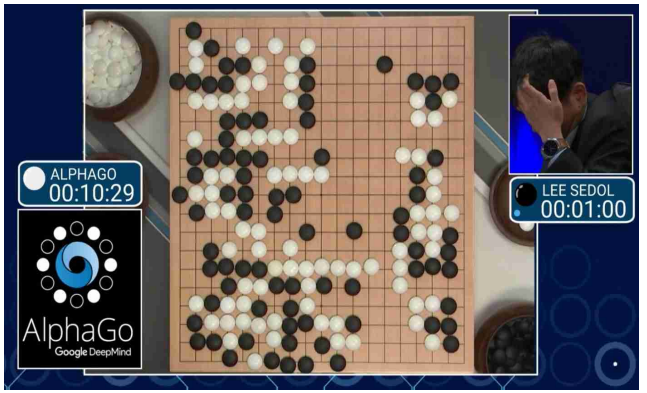


- **Trivia:** What are observations, actions and feedback?

# Other use cases

- Personalized medical treatment

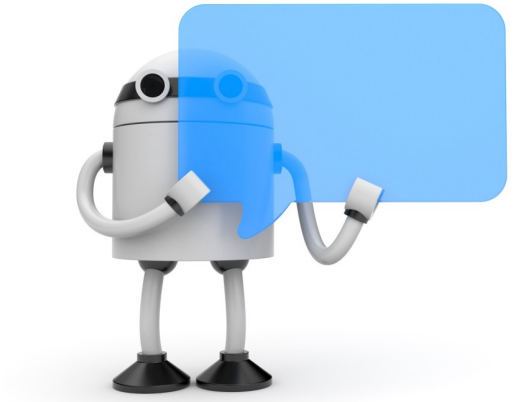
- Even more games (Go, chess, etc)

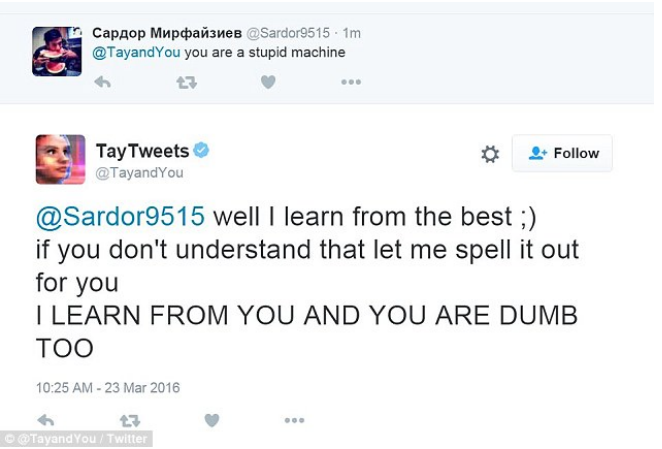



- **Trivia:** What are observations, actions and feedback?

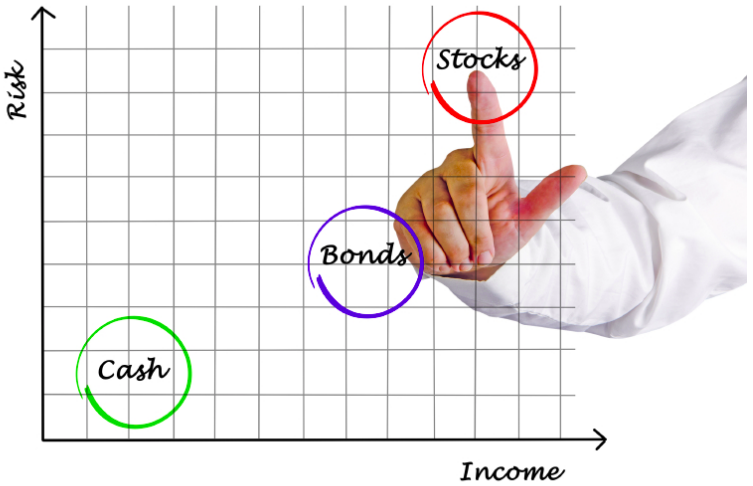
# Other use cases

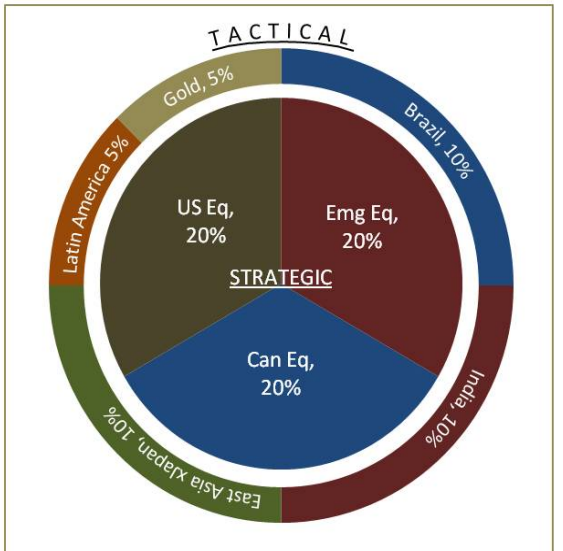
- Conversation systems (additional goals)



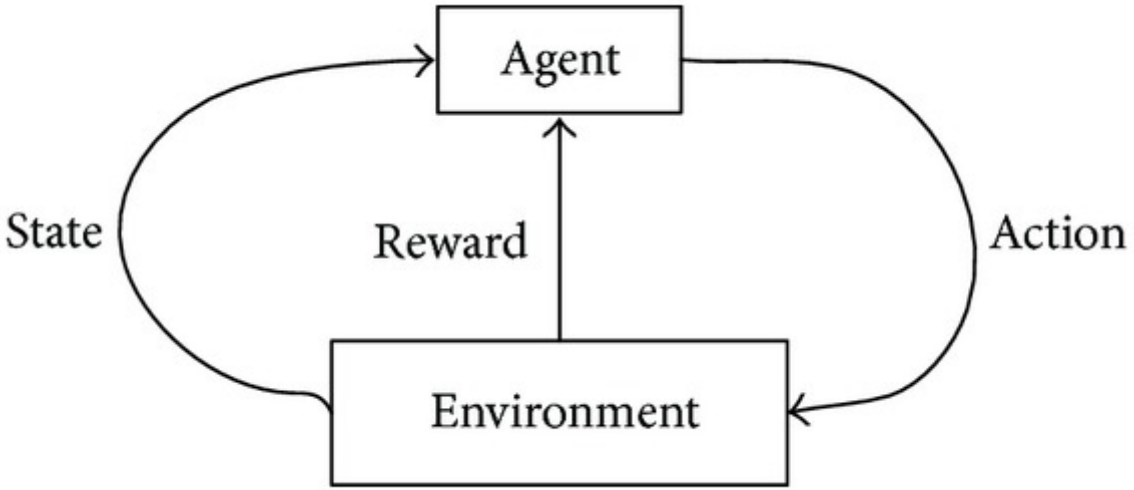



- Portfolio management (aka asset allocation)



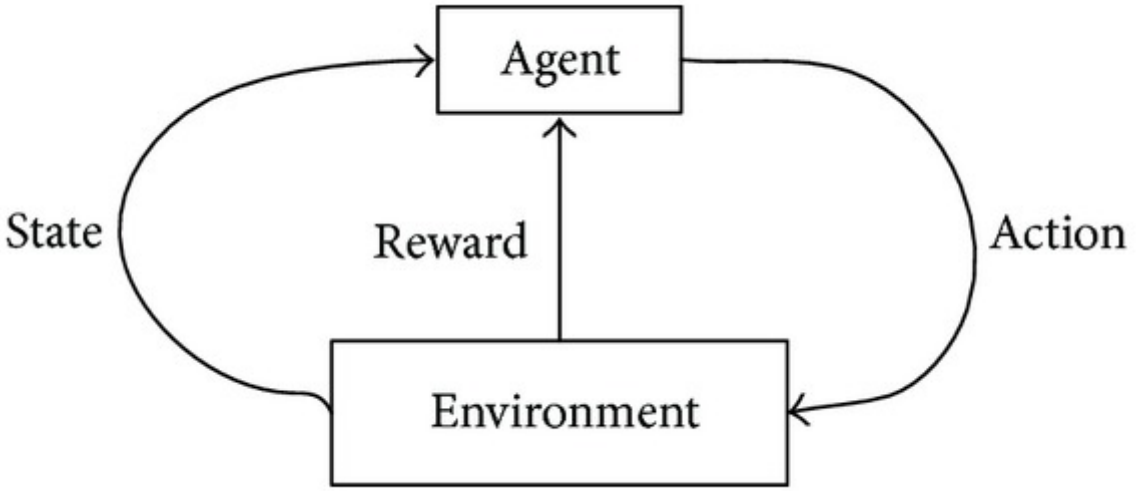
# The MDP formalism




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**

# Exploration vs Exploitation Dilemma

- Online decision making involves a fundamental choice:
  - Exploitation: make the best decision given current information
  - Exploration: gather more information
- The best long–term strategy may involve short–term sacrifices
- Gather enough information to make the best overall decisions

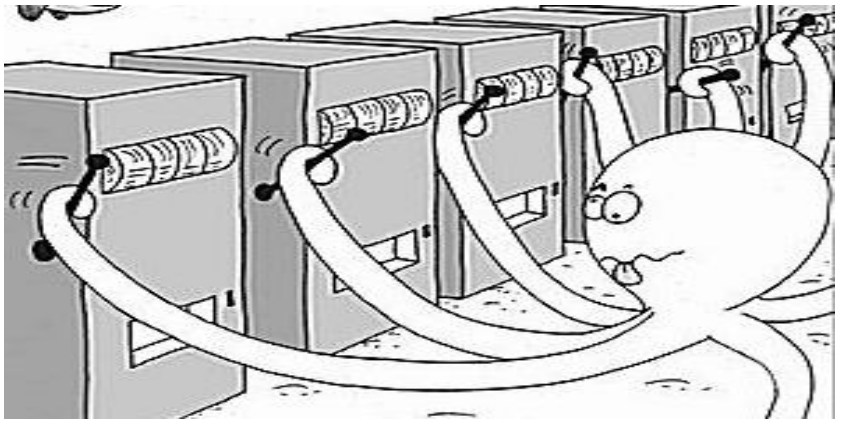
# Bandits

# Bandits?

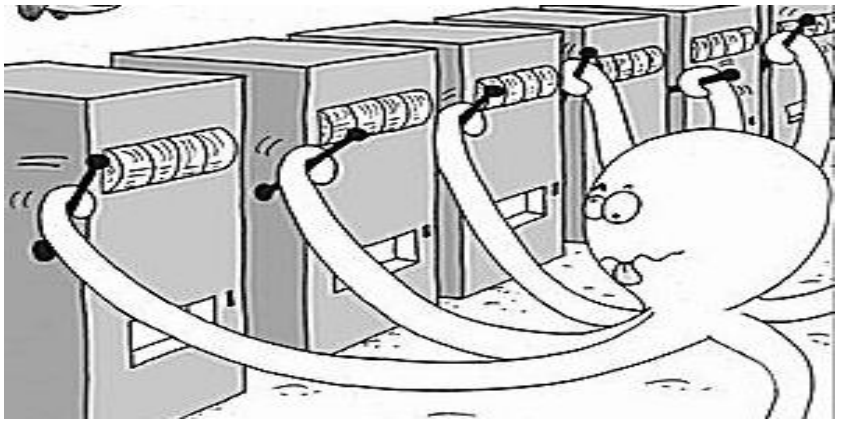
# Bandits?


ALL JACKPOTS ON CENTER LINE
MONEY HONEY Special
pays 200 coins
BAR BAR BAR
pays 150 coins
pays 100 coins
pays 50 coins
ALL JACKPOTS PAID AUTOMATICALLY IN COINS
3
IN ANY POSITION EXCEPTING CENTER LINE JACKPOT
PAYS 18 COINS
JACKPOTS
18
BAR 18
14
BAR 14
10
BAR 10
5
25¢
PRESS TO RETURN COIN
PUSH BUTTON
Bally

- Where might you find bandit problems?



- Clinical Trials
- Feynman: restaurants
- E-advertising (Yahoo, MSFT)
- Rewards to users (Diabetes study, DMN)



- Utility functions

# Action-Value Methods

$$Q_t(a) = \frac{r_1 + r_2 + \cdots + r_{k_a}}{k_a}.$$

- ε-greedy
- Vs. running update?

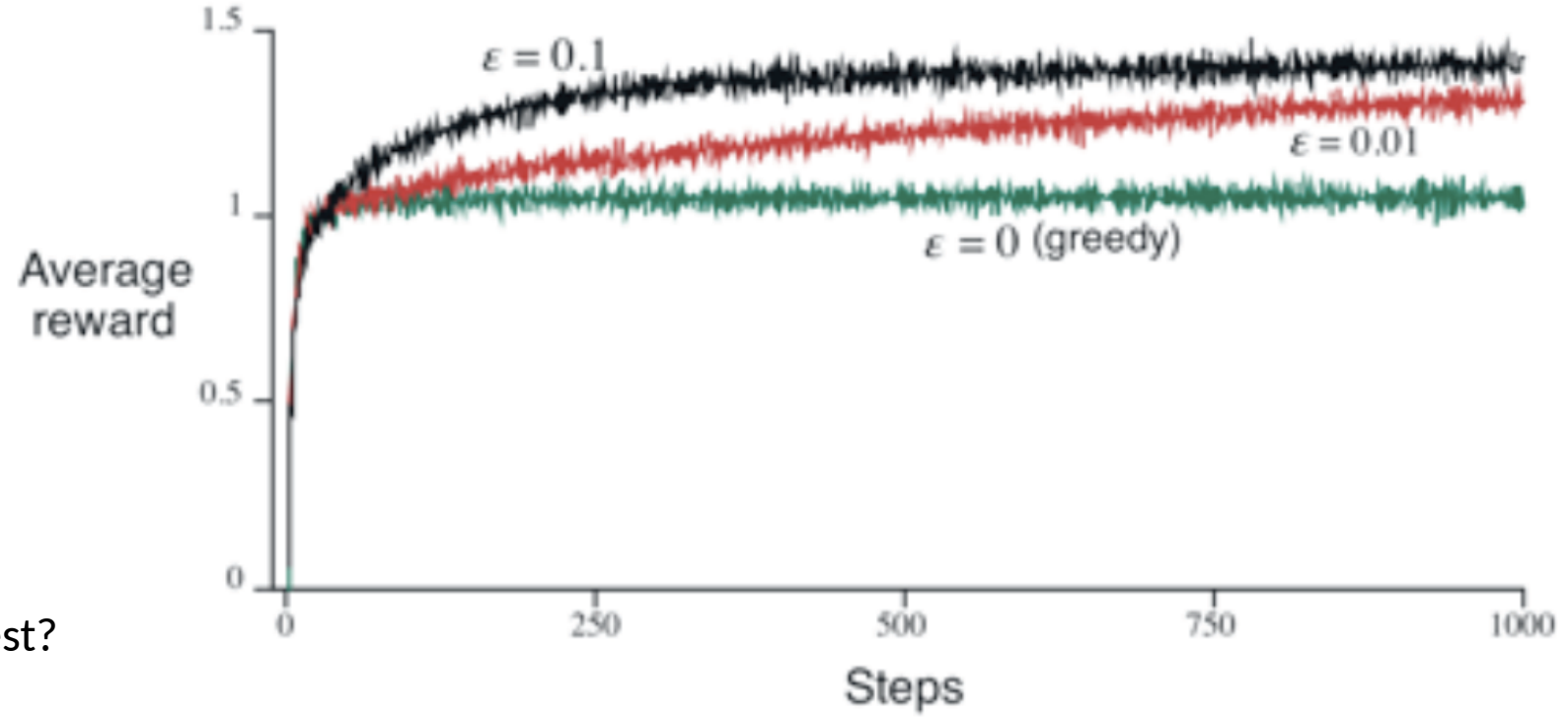
# Action-Value Methods

$$Q_t(a) = \frac{r_1 + r_2 + \cdots + r_{k_a}}{k_a}.$$

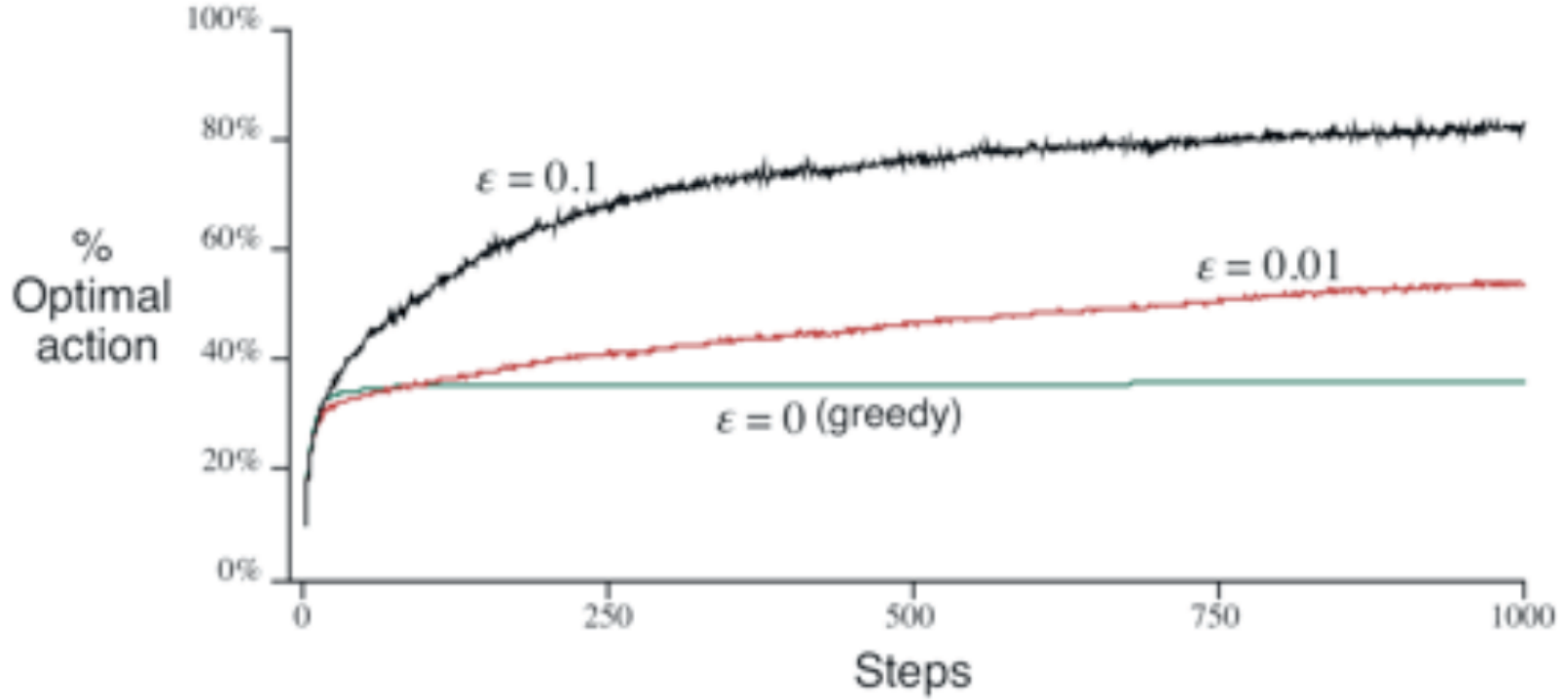
- ε-greedy
- Vs. running update?

**$\epsilon$–Greedy**

$$a_t = \begin{cases} a_t^* & \text{with probability } 1 - \epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases}$$




Which is best?



# Softmax

$$\frac{e^{Q_t(a)/\tau}}{\sum_{b=1}^{n} e^{Q_t(b)/\tau}},$$

- Gibbs / Boltzmann distribution
- Action a on $t^{th}$ play
- Temperature goes to zero
  - (may be harder to set)

# Nonstationary

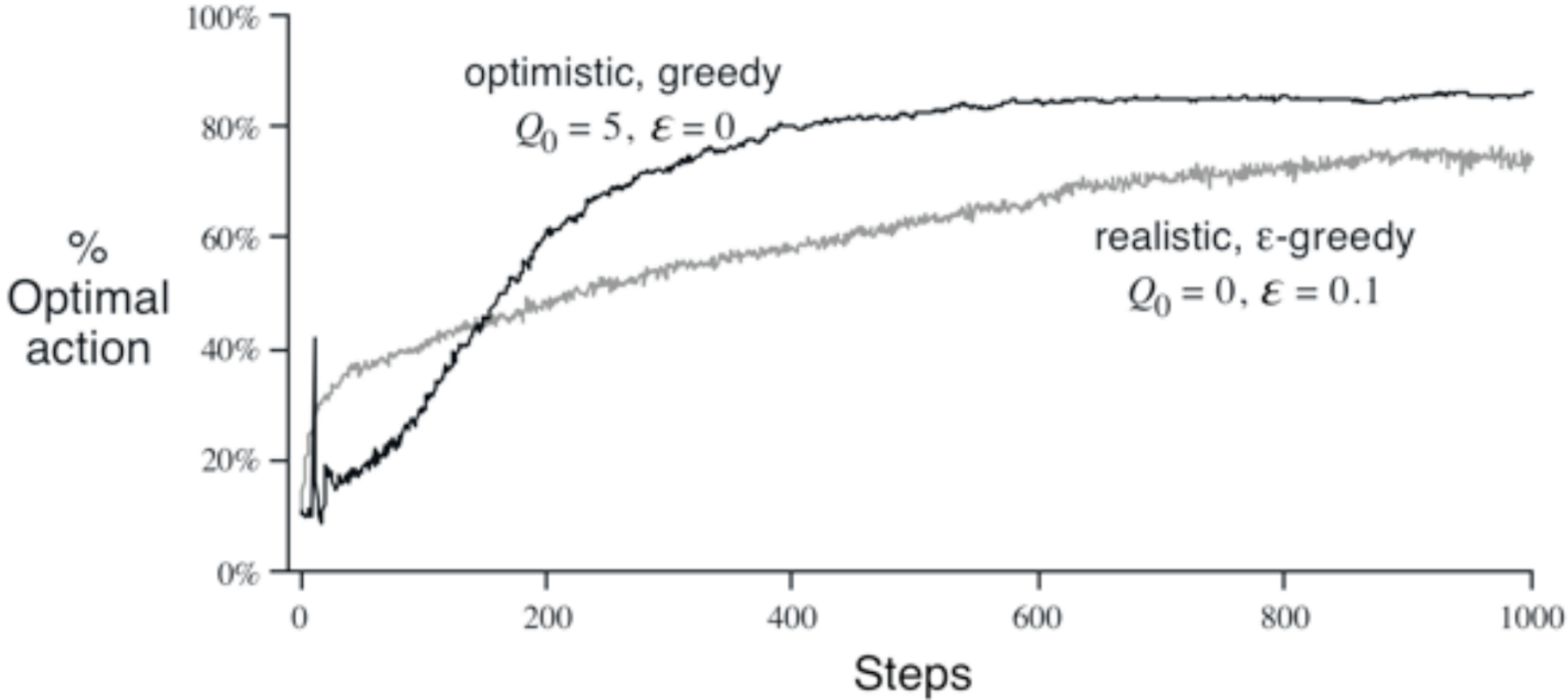
- Exponential, recency-weighted average

$$Q_{k+1} = Q_k + \alpha \left[ r_{k+1} - Q_k \right],$$

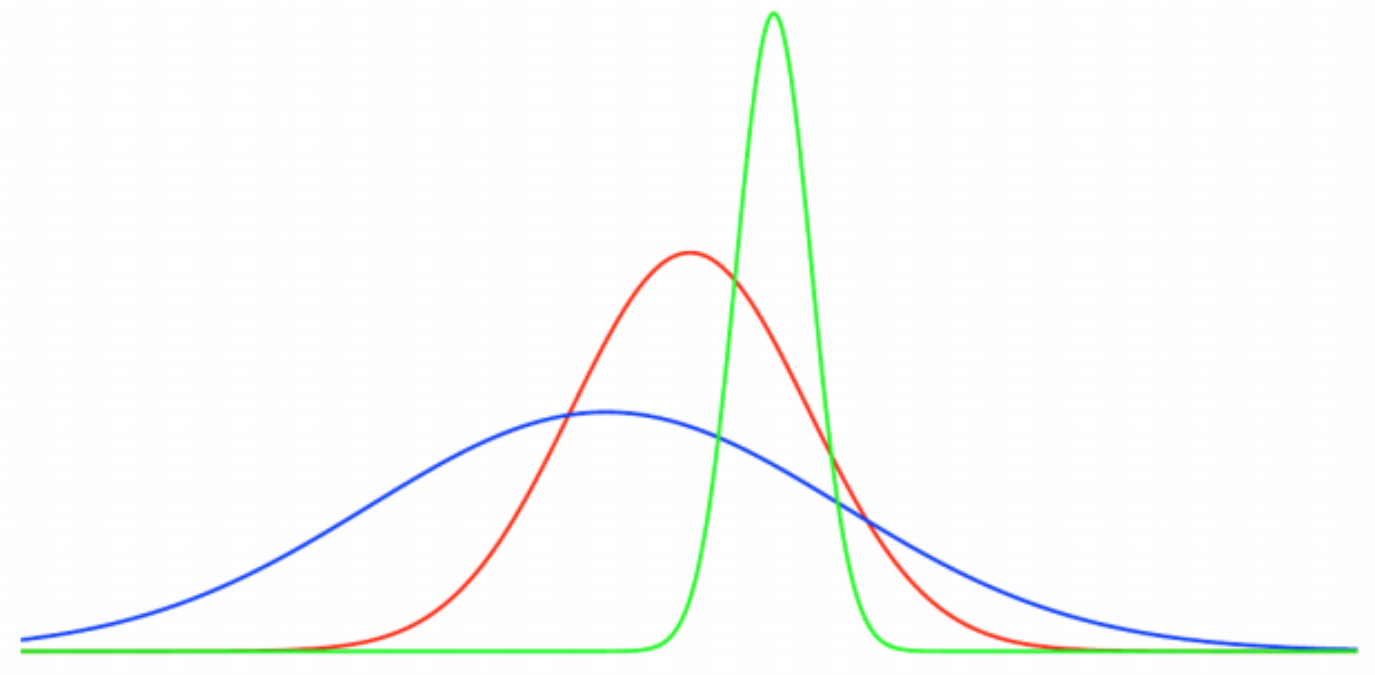
- Learning rate can vary per step
  - Why?

# Initialization

- Optimistic
- Pessimistic
- Something else?



# Optimism in Face of Uncertainty



The more **uncertain** we are about an action–value

The more important it is to **explore** that action

It could turn out to be the **best action**

# Teaching

- http://www.youtube.com/watch?v=VTbbYLvhDSM

- N-armed bandit
- Multiple n-armed bandits (contextual bandit)
  - Bei's research problem
- Reinforcement Learning

- Unfortunately, interval estimation methods are problematic in practice because of the complexity of the statistical methods used to estimate the confidence intervals.

- There is also a well-known algorithm for computing the Bayes optimal way to balance exploration and exploitation. This method is computationally intractable when done exactly, but there may be efficient ways to approximate it.

# Bandit Algorithms

- Goal: minimize regret
- Regret: defined in terms of average reward
- Average reward of best action is $\mu^*$ and any other action j as $\mu_j$. There are K total actions. $T_j(n)$ is # times tried action j during our n executed actions.

$$regret(n) = \mu^* n - \sum_{j=1}^{K} \mathbb{E}[T_j(n)]$$

# Upper Confidence Bounds

Estimate an **upper confidence** $\hat{U}_t(a)$ for each action value

Such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with **high probability**

This depends on the **number of items** $N(a)$ has been selected

- Small $N_t(a) \Rightarrow$ large $\hat{U}_t(a)$ (estimated value is **uncertain**)
- Large $N_t(a) \Rightarrow$ small $\hat{U}_t(a)$ (estimated value is **accurate**)

Select action maximizing **Upper Confidence Bound** (UCB)

$$a_t = arg \max_{a \in \mathcal{A}} \hat{Q}(a) + \hat{U}(a)$$

# Hoeffding's Inequality

**Theorem**

*Hoeffding's Inequality Let $X_1, \ldots, X_t$ be i.i.d. random variables in $[0, 1]$, and let $\overline{X}_t = \frac{1}{T} \sum_{t=1}^{T} X_t$ be the sample mean. Then*

$$\mathbb{P}[\mathbb{E}[X] > \overline{X}_t + u] \leq e^{-2tu^2}$$

We will apply Hoeffding's Inequality to rewards of the bandit conditioned on selecting action $a$

$$\mathbb{P}[Q(a) > \hat{Q}_t(a) + U_t(a)] \leq e^{-2N_t(a)U_t(a)^2}$$

# Calculating Upper Confidence Bound

Pick a probability $p$ that **true value** exceeds UCB

Now solve for $U_t(a)$

$$e^{-2N_t(a)U_t(a)^2} = p$$

$$U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$$

**Reduce** $p$ as we observe more rewards, e.g., $p = t^{-4}$

**Ensures** we select optimal actions as $t \to \infty$

$$U_t(a) = \sqrt{\frac{2 \log t}{N_t(a)}}$$

# UCB1

This leads to the UCB1 algorithm

$$a_t = arg\max_{a \in \mathcal{A}} Q(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$$

**Theorem**

*At time $T$, the expected total regret of the UCB policy is at most*

$$\mathbb{E}[L_T] \leq 8 \log T \sum_{a | \Delta_a < \Delta_{a^*}} \frac{1}{\Delta_a} + \left(1 + \frac{\pi^2}{3}\right) \sum_{a \in \mathcal{A}} \Delta_a$$

# UCB1 - Tuned

- Can compute sample variance for each action, $\sigma_j$

Then use the upper confidence bound for action $j$ of:

$$\sqrt{\frac{\ln n}{n_j} \min\left(\frac{1}{4}, \left(\sigma_j + \frac{2 \ln n}{n_j}\right)\right)}$$

- Easy hack for non-stationary environments?

# UCB vs e–Greedy on 10–armed Bandit

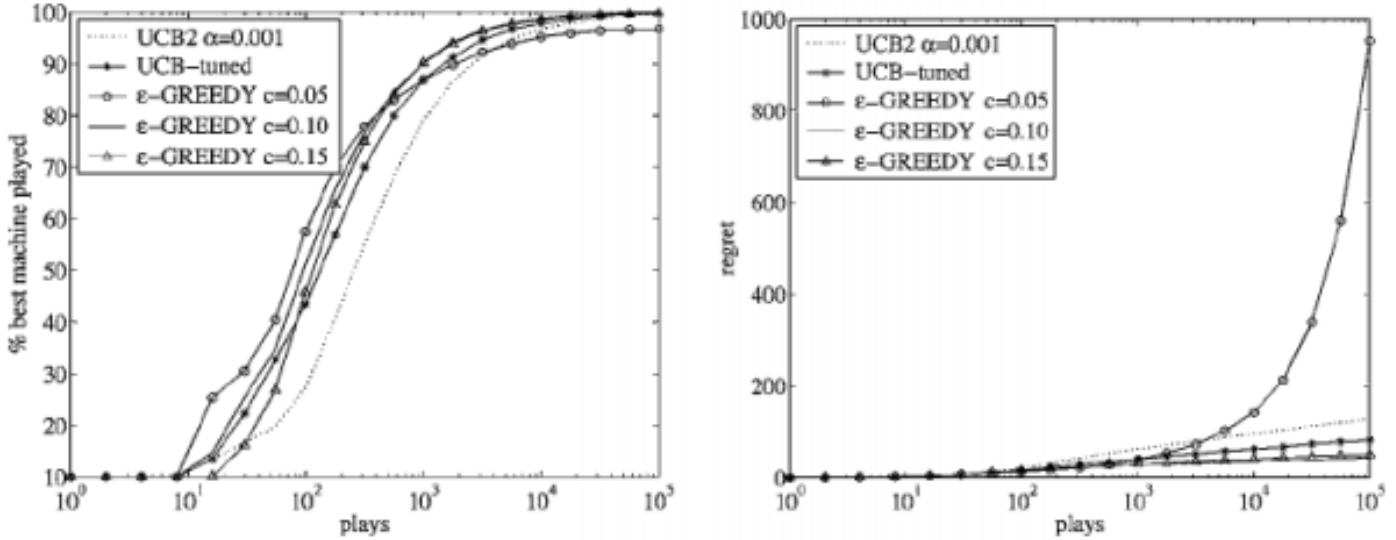
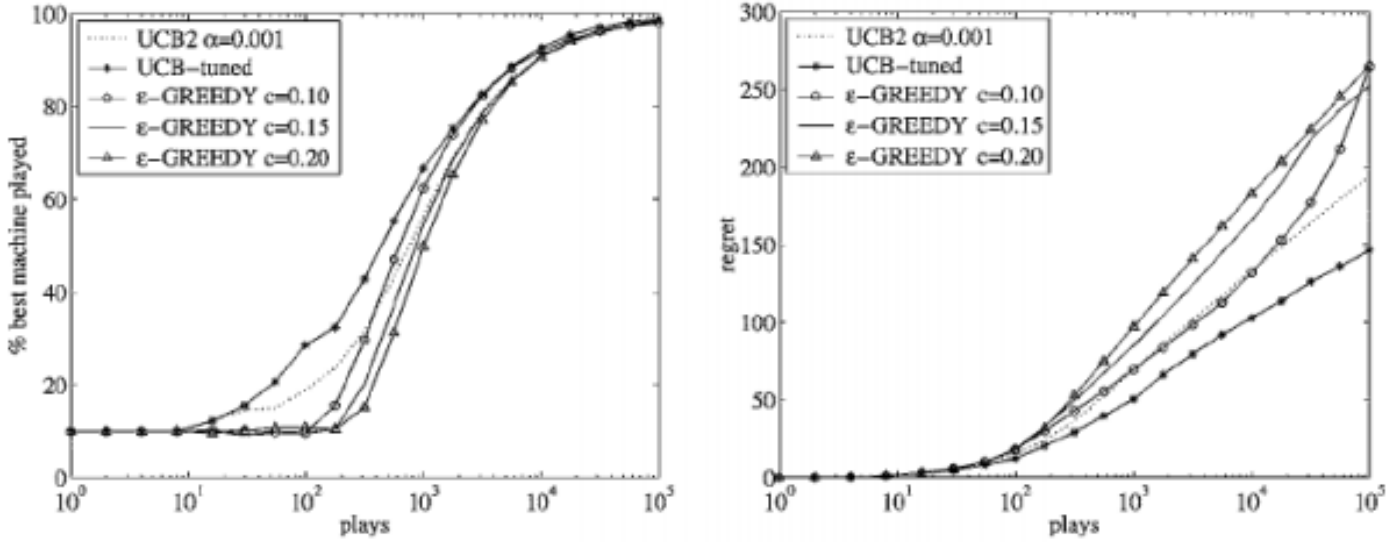



*Figure 9.* Comparison on distribution 11 (10 machines with parameters 0.9, 0.6, . . . , 0.6).



# Adversarial Bandits

- Optimism can be naïve

**Definition:** An *adversarial bandit problem* is a pair $(K, \mathbf{x})$, where $K$ represents the number of *actions* (henceforth indexed by $i$), and $\mathbf{x}$ is an infinite sequence of *payoff vectors* $\mathbf{x} = \mathbf{x}(1), \mathbf{x}(2), \ldots$, where $\mathbf{x}(t) = (x_1(t), \ldots, x_K(t))$ is a vector of length $K$ and $x_i(t) \in [0, 1]$ is the reward of action $i$ on step $t$.

- Reward vectors must be fixed in advance of the algorithm running.
- Payoffs can depend adversarially on the algorithm the player decides to use.
- Ex: if the player chooses the strategy of always picking the first action, then the adversary can just make that the worst possible action to choose.
- Rewards cannot depend on the random choices made by the player during the game.

- Why can’t the adversary just make all the payoffs zero? (or negative!)

- Why can’t the adversary just make all the payoffs zero? (or negative!)
- In this event the player won’t get any reward, but he can emotionally and psychologically accept this fate. If he never stood a chance to get any reward in the first place, why should he feel bad about the inevitable result?

- Why can’t the adversary just make all the payoffs zero? (or negative!)
- In this event the player won’t get any reward, but he can emotionally and psychologically accept this fate. If he never stood a chance to get any reward in the first place, why should he feel bad about the inevitable result?
- What a truly cruel adversary wants is, at the end of the game, to show the player what he could have won, and have it far exceed what he actually won. In this way the player feels regret for not using a more sensible strategy, and likely returns to the casino to lose more money.

- Why can’t the adversary just make all the payoffs zero? (or negative!)
- In this event the player won’t get any reward, but he can emotionally and psychologically accept this fate. If he never stood a chance to get any reward in the first place, why should he feel bad about the inevitable result?
- What a truly cruel adversary wants is, at the end of the game, to show the player what he could have won, and have it far exceed what he actually won. In this way the player feels regret for not using a more sensible strategy, and likely returns to the casino to lose more money.
- The trick that the player has up his sleeve is precisely the randomness in his choice of actions, and he can use its objectivity to partially overcome even the nastiest of adversaries.

- Exp3: Exponential-weight algorithm for Exploration and Exploitation

1. Given $\gamma \in [0, 1]$, initialize the weights $w_i(1) = 1$ for $i = 1, \ldots, K$.
2. In each round $t$:
   1. Set $p_i(t) = (1 - \gamma)\frac{w_i(t)}{\sum_{j=1}^{K} w_j(t)} + \frac{\gamma}{K}$ for each $i$.
   2. Draw the next action $i_t$ randomly according to the distribution of $p_i(t)$.
   3. Observe reward $x_{i_t}(t)$.
   4. Define the estimated reward $\hat{x}_{i_t}(t)$ to be $x_{i_t}(t)/p_{i_t}(t)$.
   5. Set $w_{i_t}(t+1) = w_{i_t}(t)e^{\gamma \hat{x}_{i_t}(t)/K}$
   6. Set all other $w_j(t+1) = w_j(t)$.

# k-Meterologists Problem

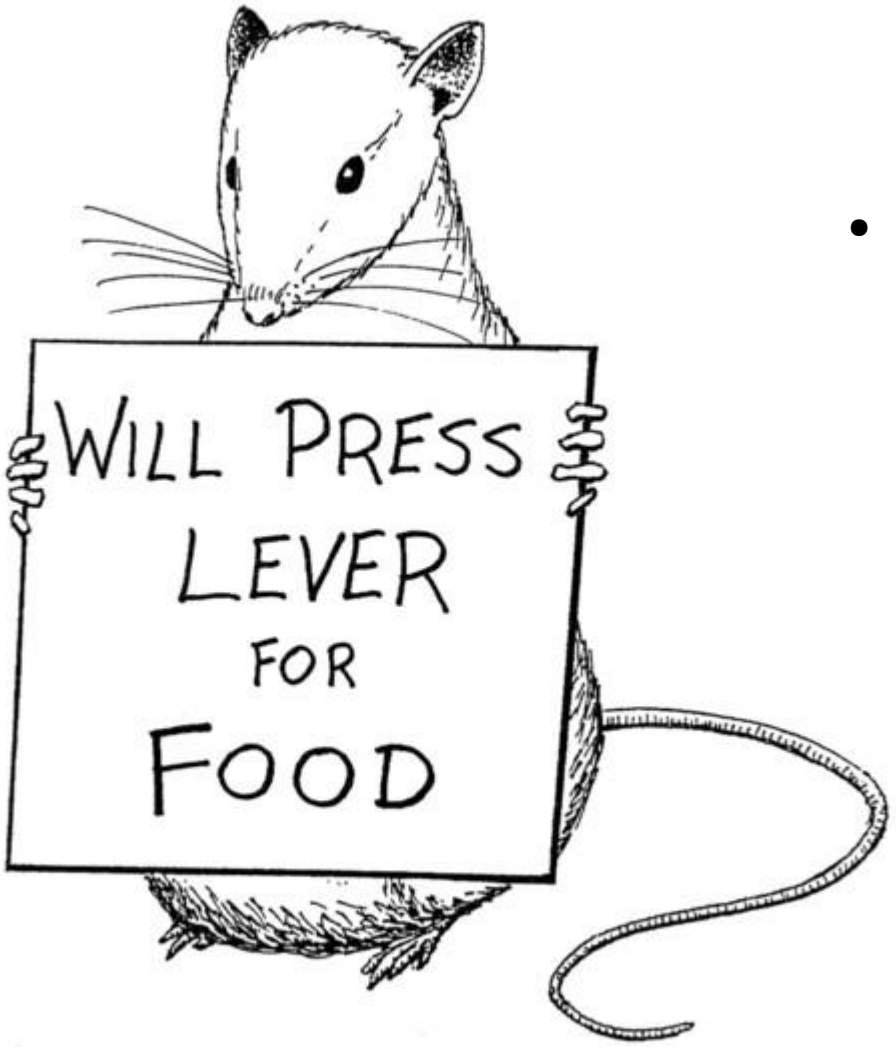
- ICML-09, Diuk, Li, and Leffler

- Imagine that you just moved to a new town that has multiple (**k**) radio and TV stations. Each morning, you tune in to one of the stations to find out what the weather will be like. Which of the **k** different meteorologists making predictions every morning is the most trustworthy? Let us imagine that, to decide on the best meteorologist, each morning for the first **M** days you tune in to all **k** stations and write down the probability that each meteorologist assigns to the chances of rain. Then, every evening you write down a 1 if it rained, and a 0 if it didn't. Can this data be used to determine who is the best meteorologist?

- Related to expert algorithm selection

Now back to… reinforcement learning!

# Optimal policy (Monte-carlo)




- Naive objective: $R(z)$

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

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

  **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:
  - $P(s_{t+1}|s_t, a_t)$
  - $R(z)$
  - Forall $s \in S \quad a \in A$
- How can you get optimal action?

# Combinatorial optimization

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

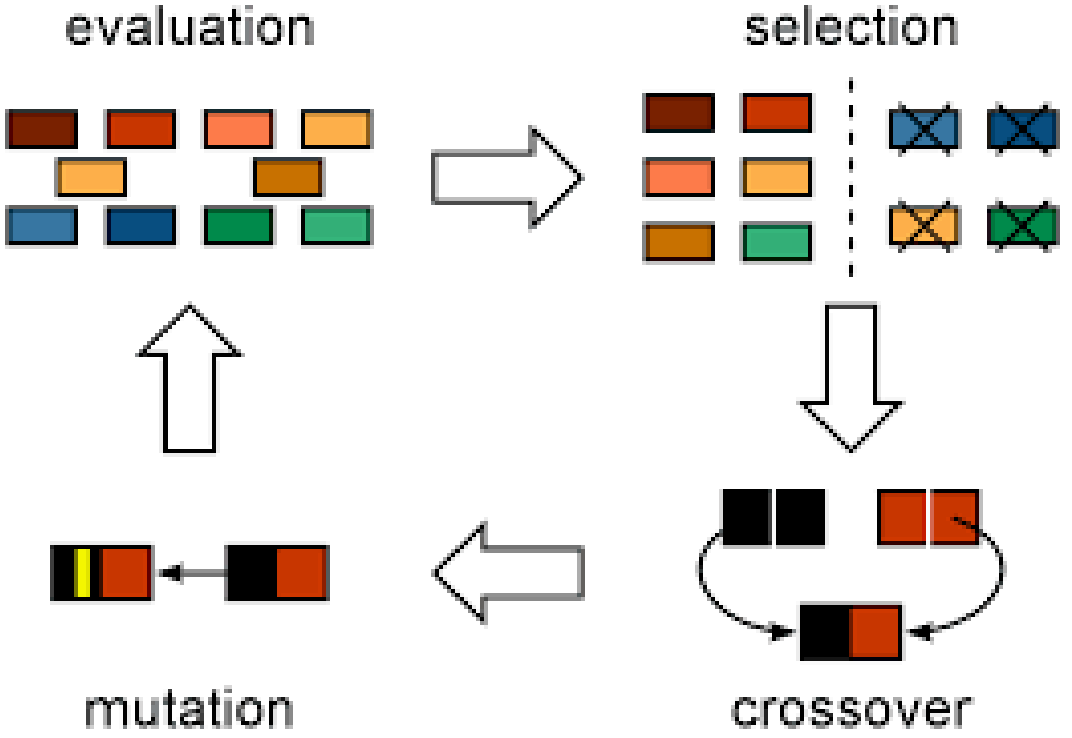
# Combinatorial optimization

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

# Genetic Algorithms

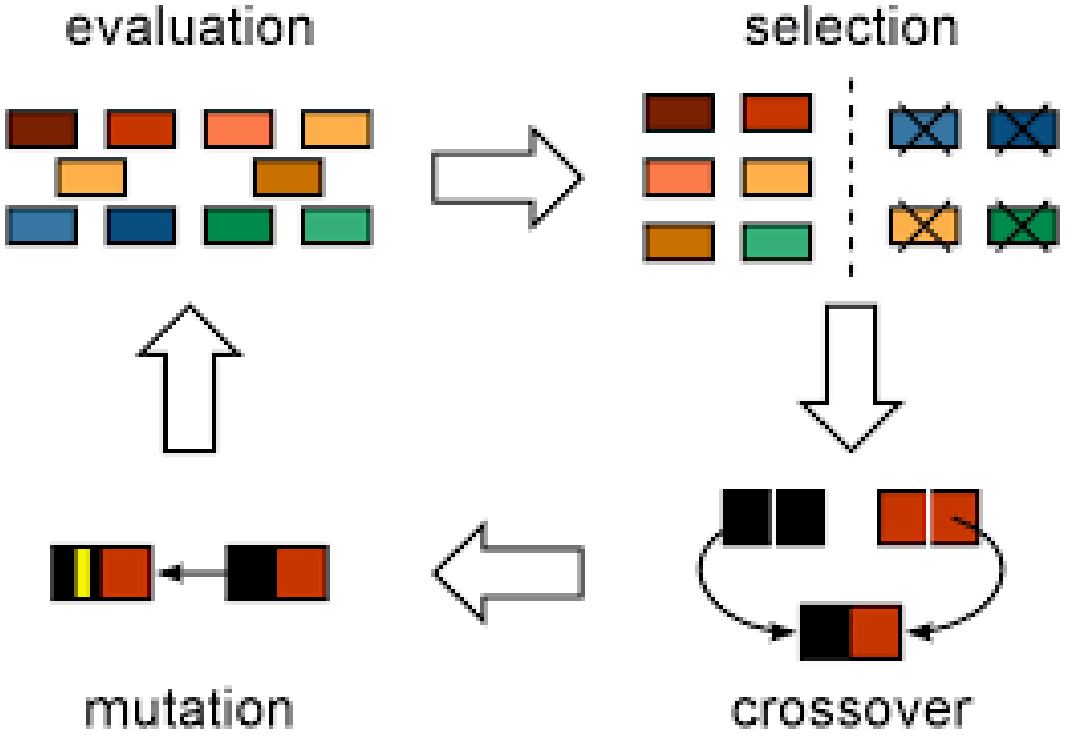
# Genetic algorithms

- 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

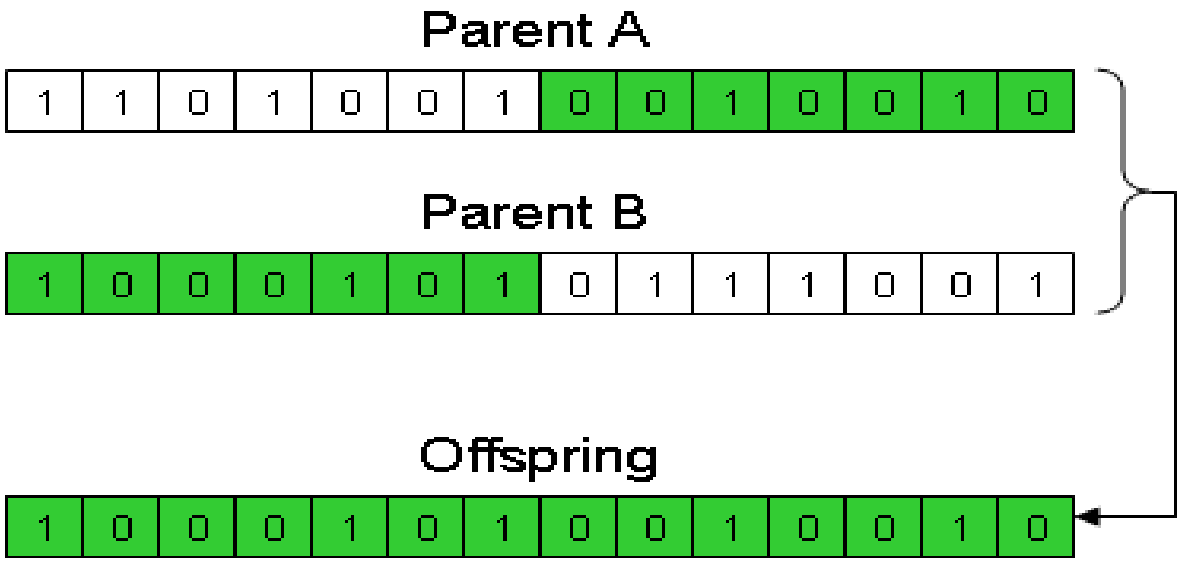


# Genetic algorithms

- 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

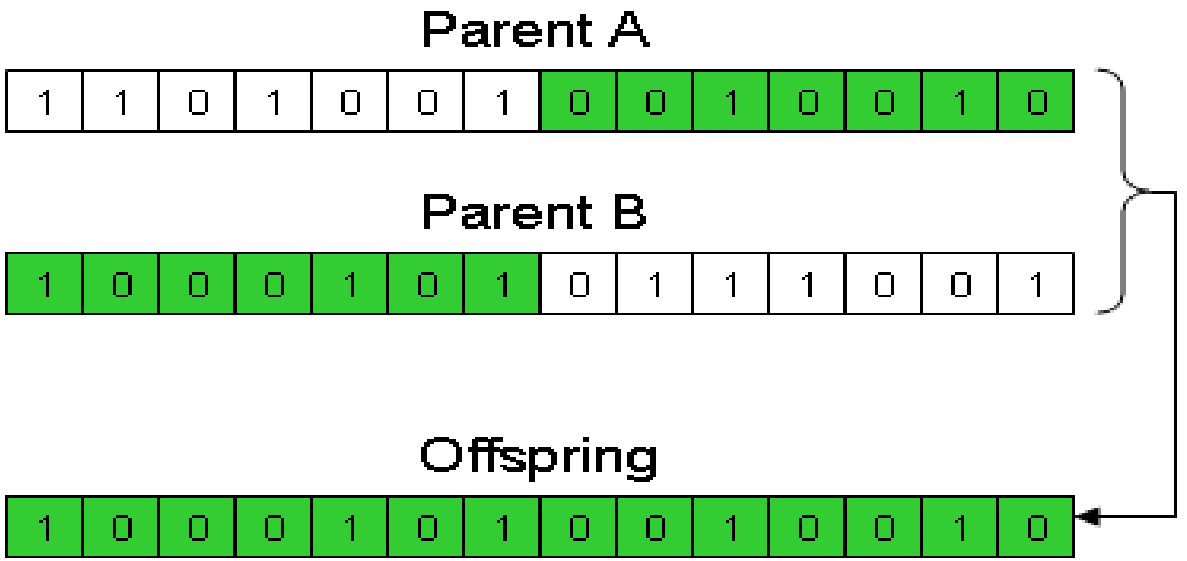
# Crossover

- 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 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, ...

# Questions?

British Hedgehog Preservation Society