

Practical RL

Episode 1, spring'19

Intro to Reinforcement Learning

Why should you care



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? Fixed a bug in seminar? 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

$$\begin{aligned} & (x, y) \\ & \text{[banner,page], ctr} \\ & a_\theta(x) \rightarrow y \\ & \text{linear / tree / NN} \\ & L(y, a_\theta(x)) \\ & \text{MSE, crossentropy} \end{aligned}$$

Find:

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

Supervised learning

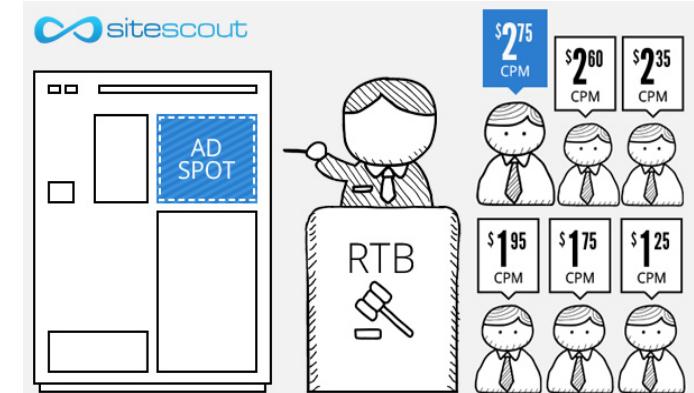
Great... except if we have no reference answers

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

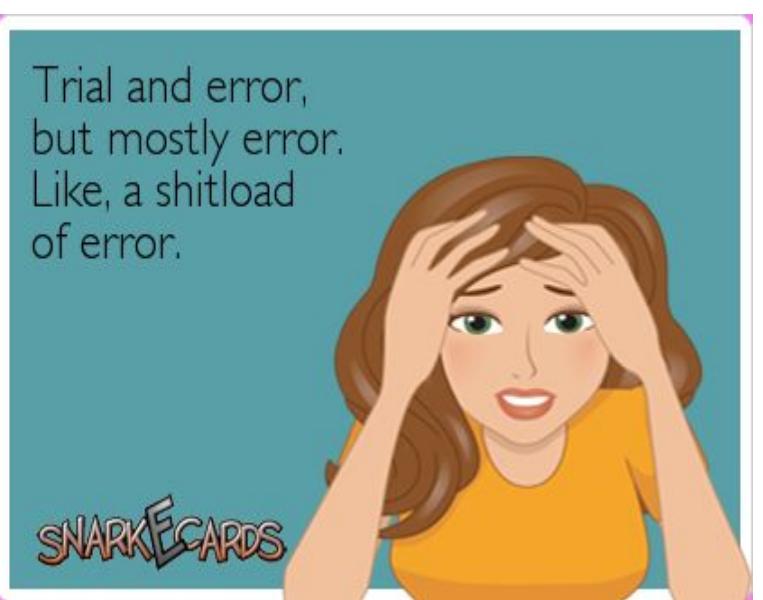


Ideas?

Duct tape approach

Common idea:

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



Giant Death Robot (GDR)

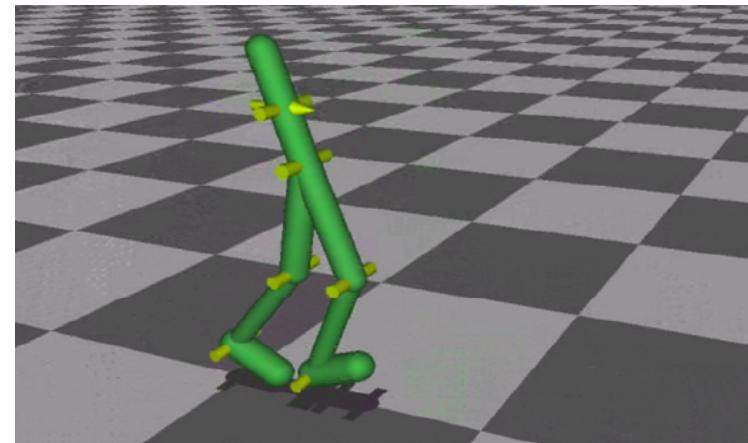
Great... except if we have no reference answers

We have:

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

We want:

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

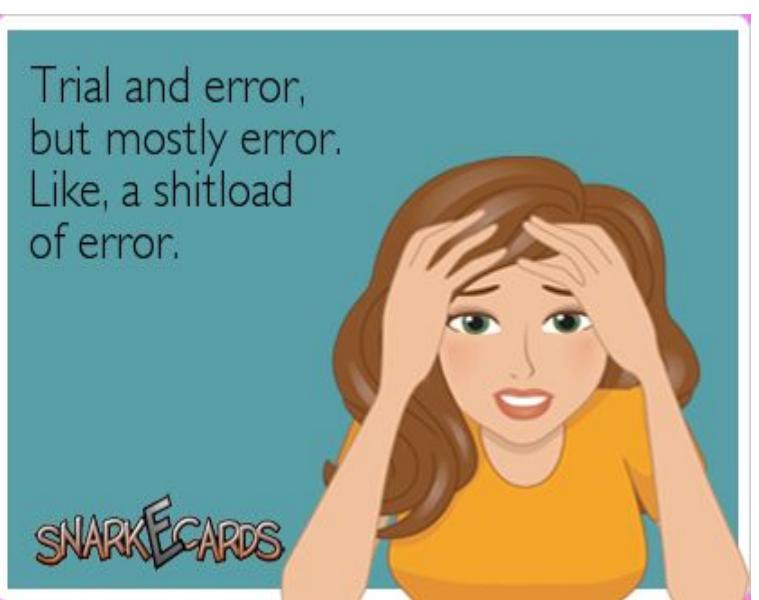
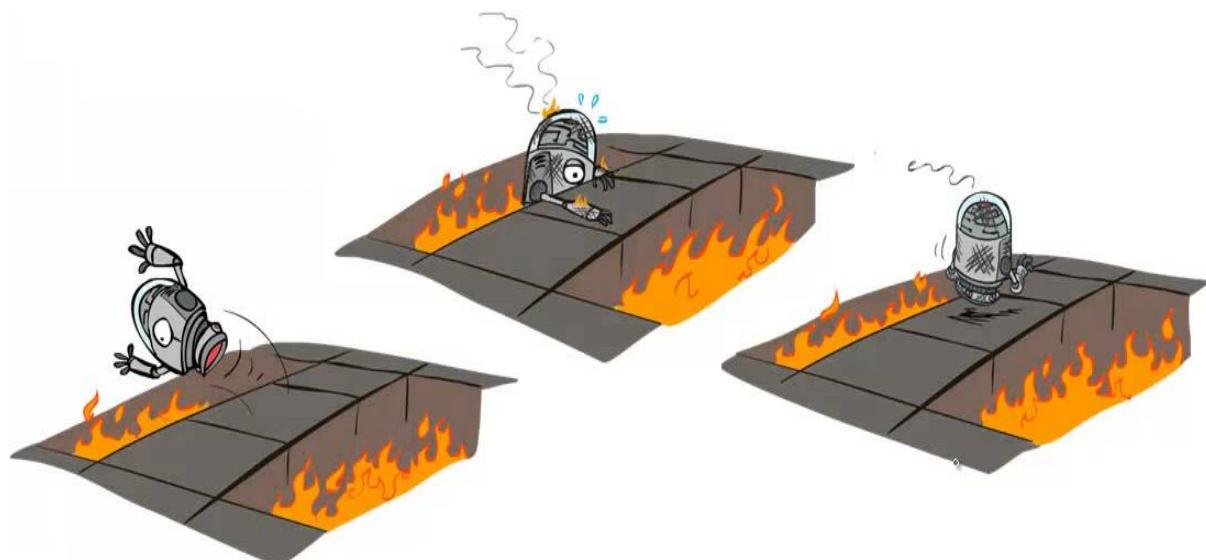


Ideas?

Duct tape approach (again)

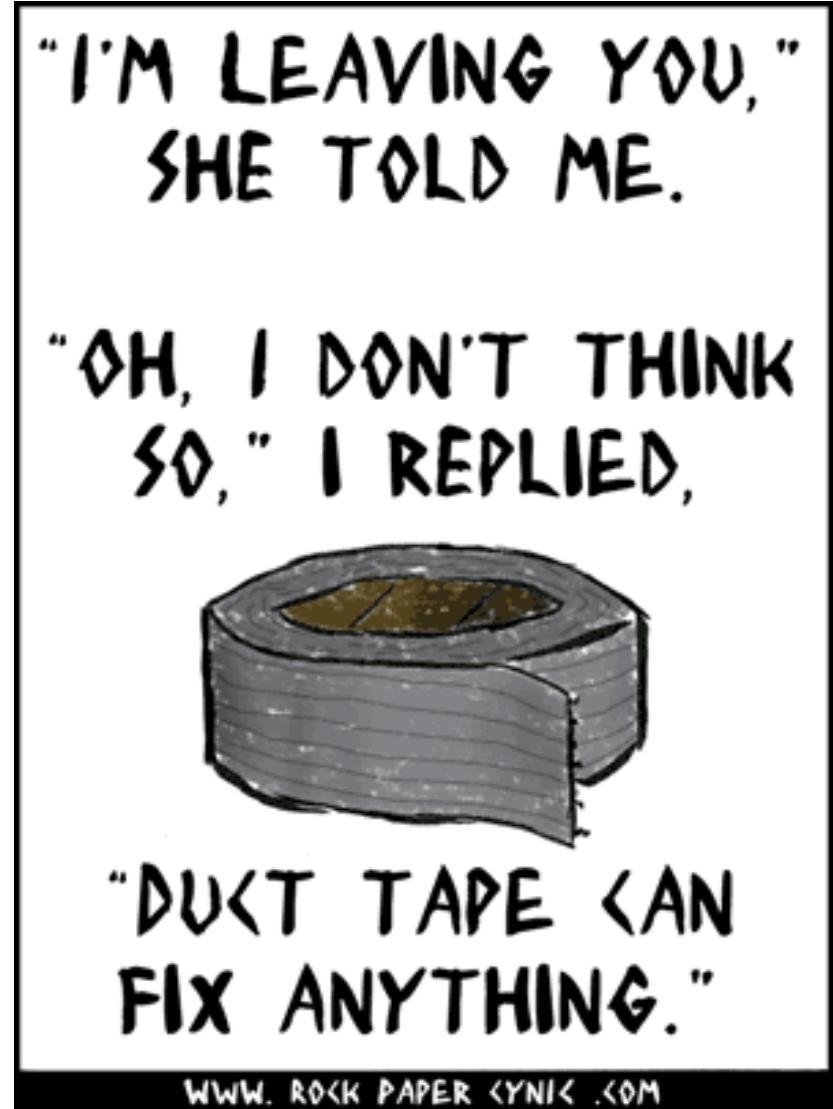
Common idea:

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



SNARK ECARDS

Duct tape approach



Problems

Problem 1:

- What exactly does the “optimal action” mean?

Extract as much
money as you can
right now

vs

Make user happy
so that he would
visit you again

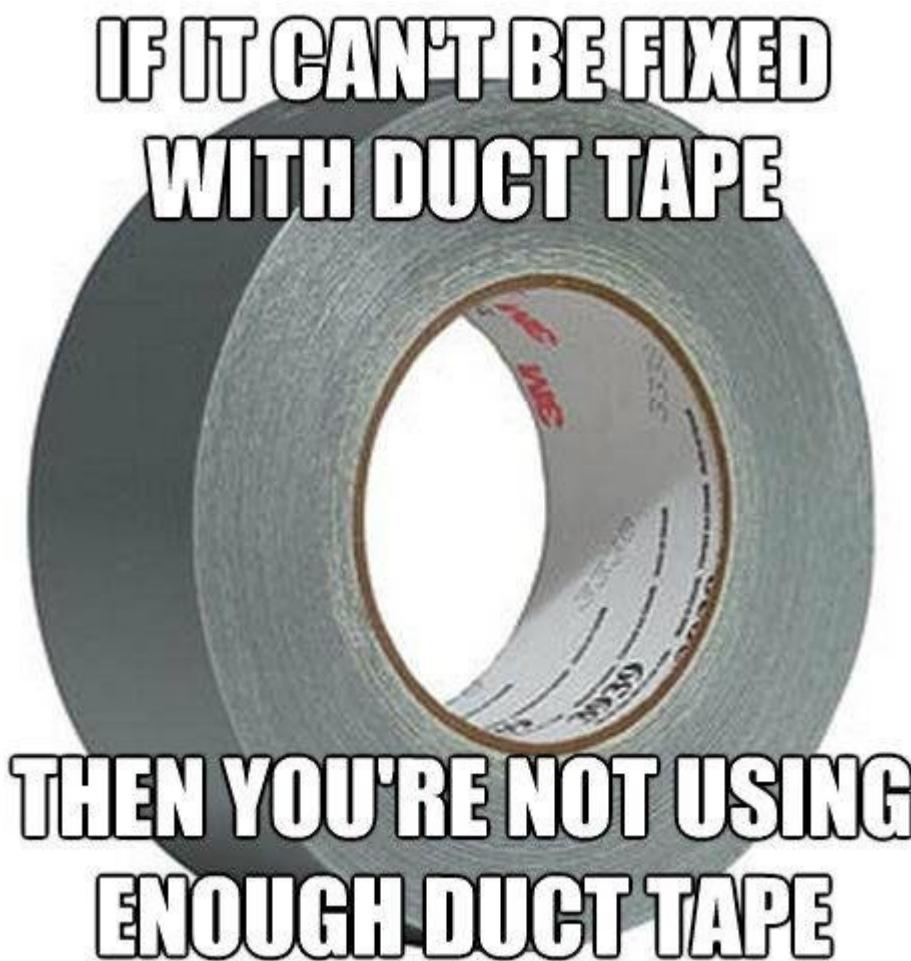
Problems

Problem 2:

- If you always follow the “current optimal” strategy, you may never discover something better.
- If you show the same banner to 100% users, you will never learn how other ads affect them.

Ideas?

Duct tape approach



zipmeme

Reinforcement learning

STAND BACK



**I'M GOING TO TRY
SCIENCE**

What is: bandit



Examples:

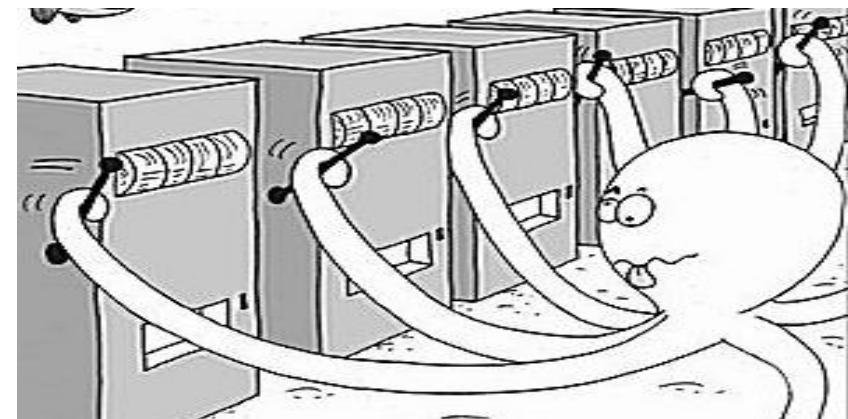
- banner ads (RTB)
- recommendations
- medical treatment

What is: bandit



Examples:

- banner ads (RTB)
- recommendations
- medical treatment



What is: bandit

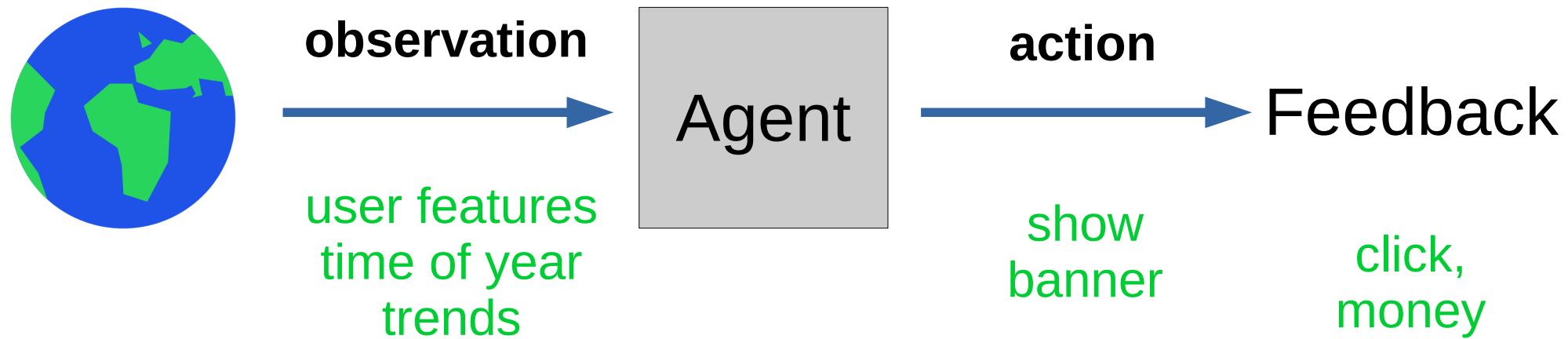


Examples:

- banner ads (RTB)
- recommendations
- medical treatment

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

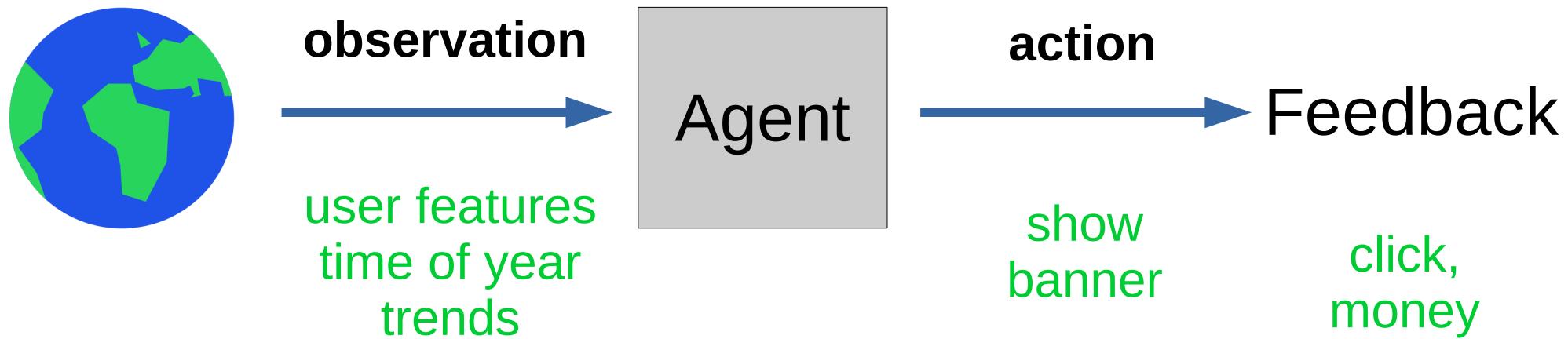
What is: bandit



Examples:

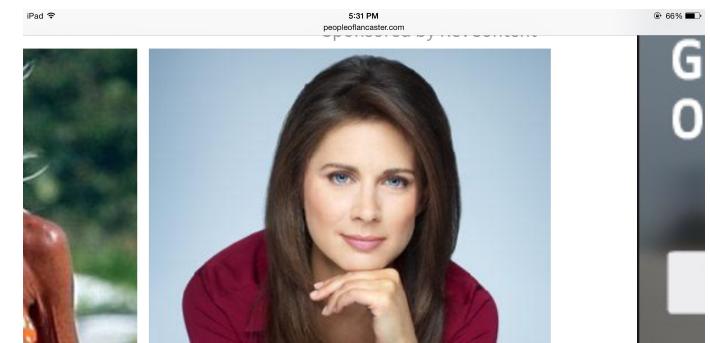
- banner ads (RTB)
- recommendations
- medical treatment

What is: bandit



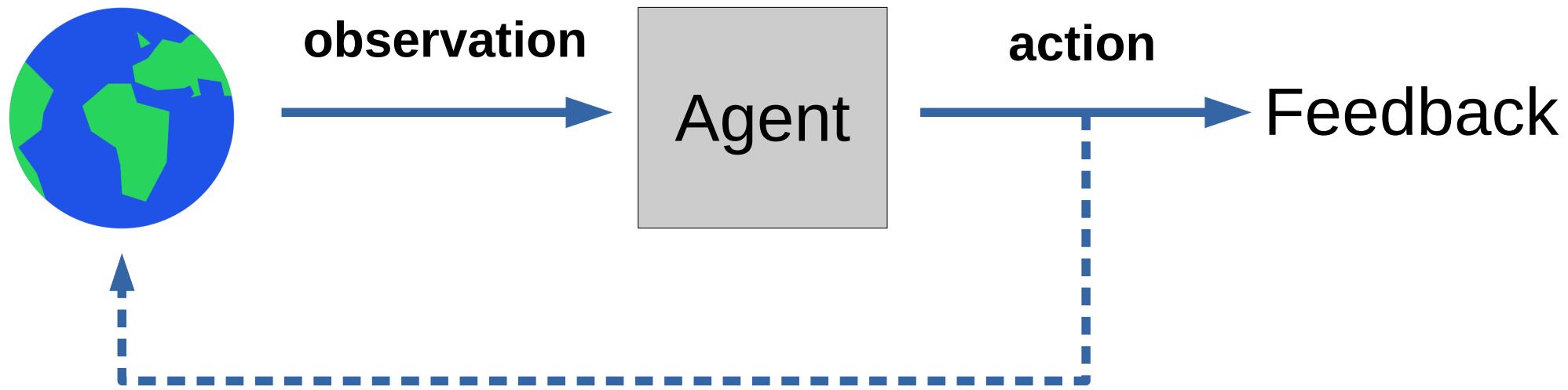
Q: You're Yandex/Google/Youtube.
There's a kind of banners that would have great click rates: the “clickbait”.

Is it a good idea to show clickbait?



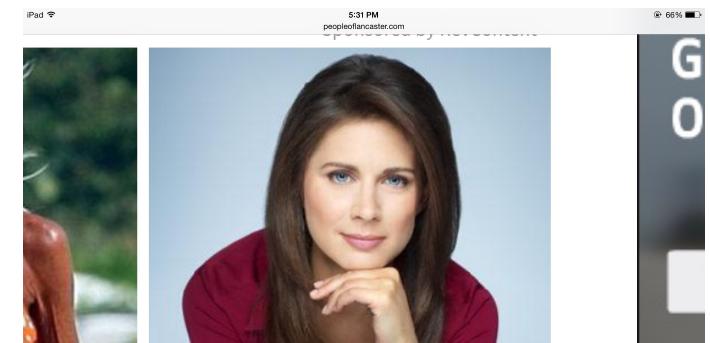
Who 23 Celebrities You Would Never Guess Are Actually Black
Age 0

What is: bandit



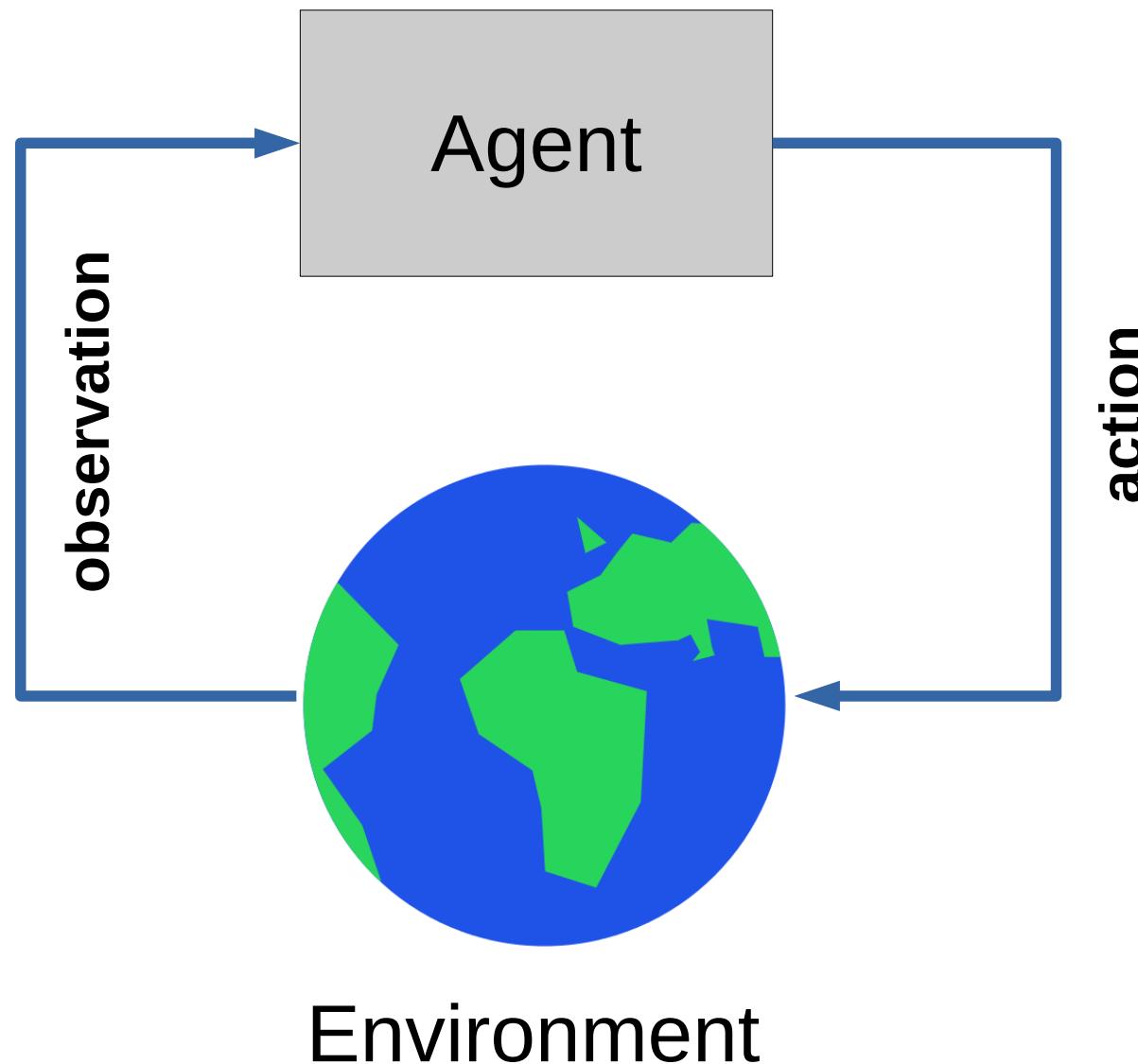
Q: You're Yandex/Google/Youtube.
There's a kind of banners that would have great click rates: the “clickbait”.

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

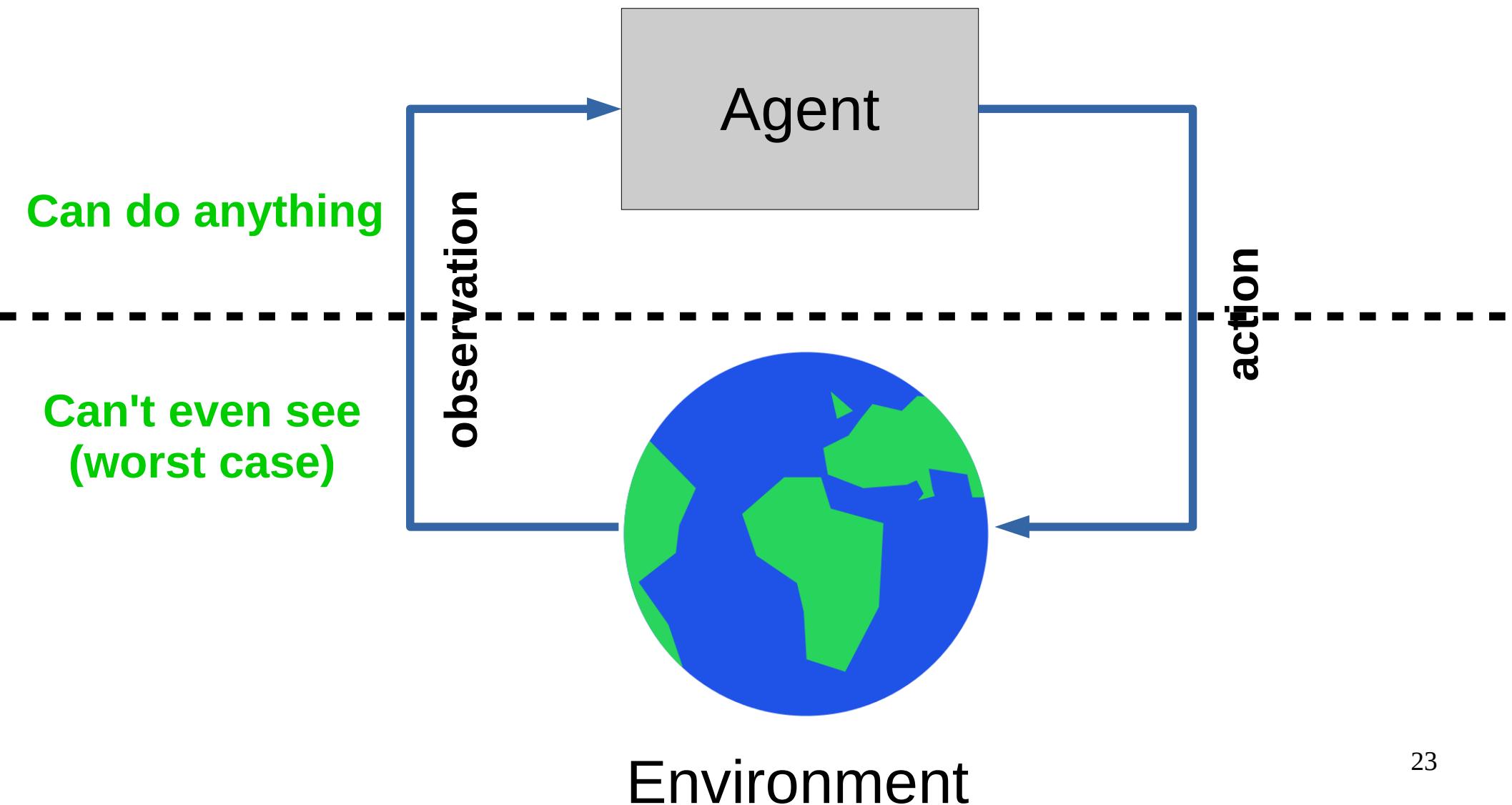


Who 23 Celebrities You
Age Would Never Guess Are
Actually Black

What is: decision process

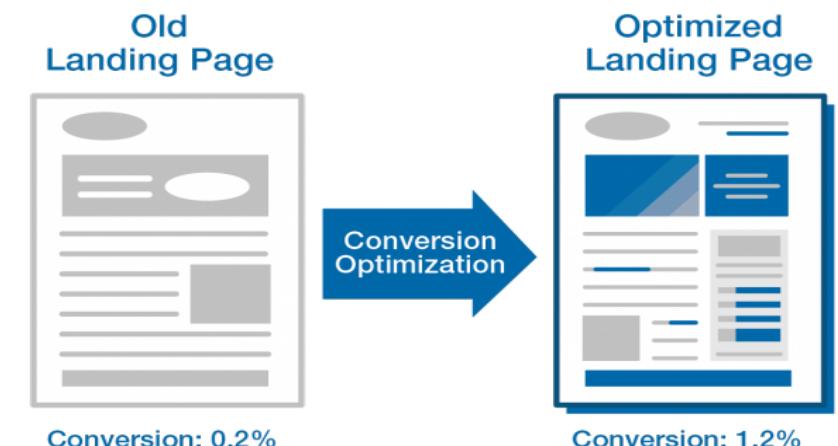
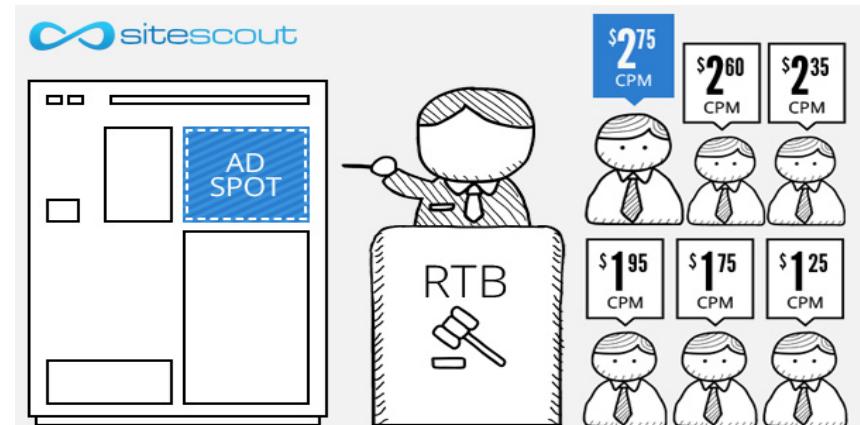


What is: decision process

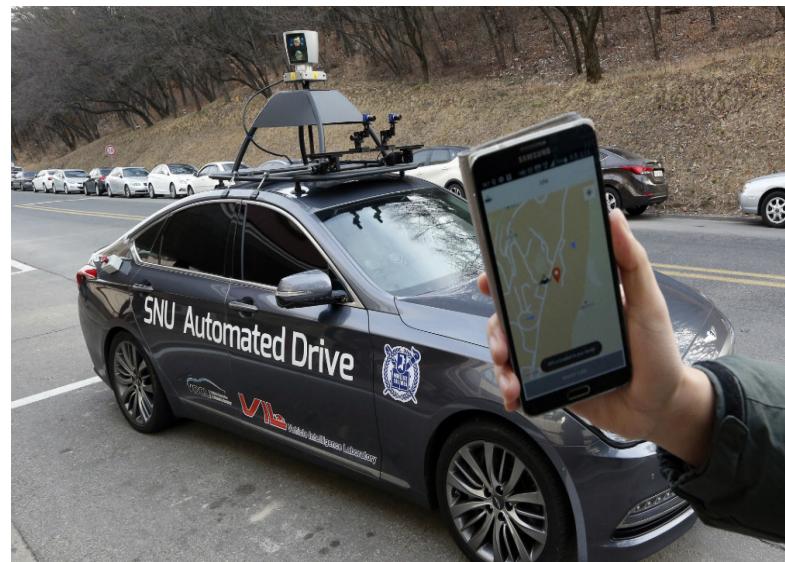


Reality check: web

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

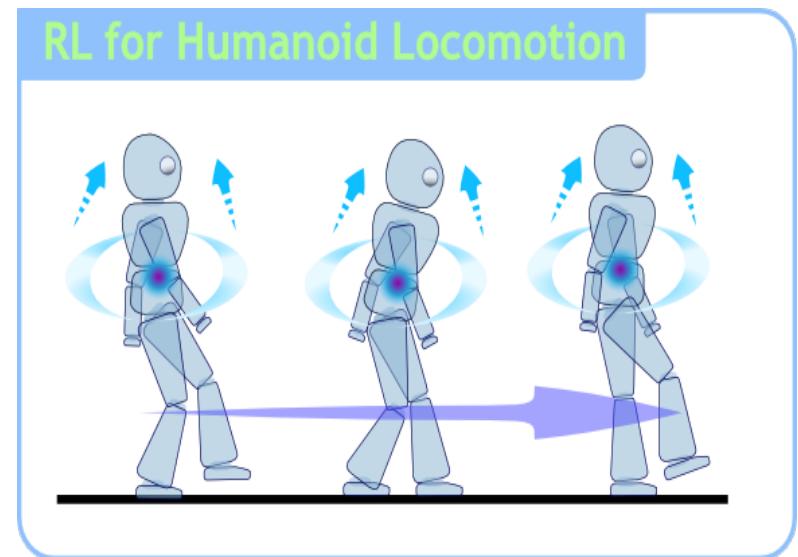


Reality check: dynamic systems

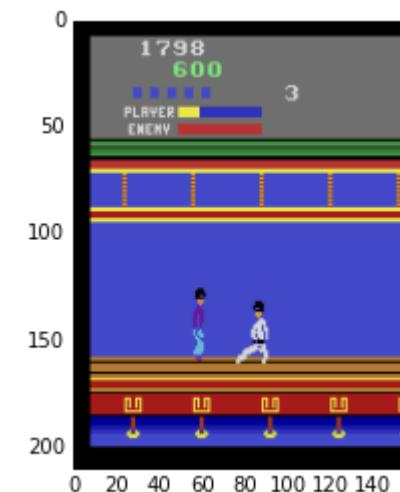
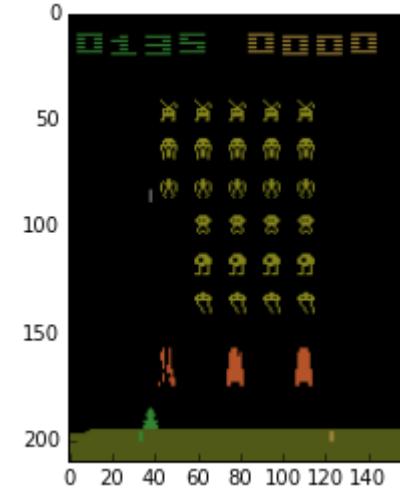
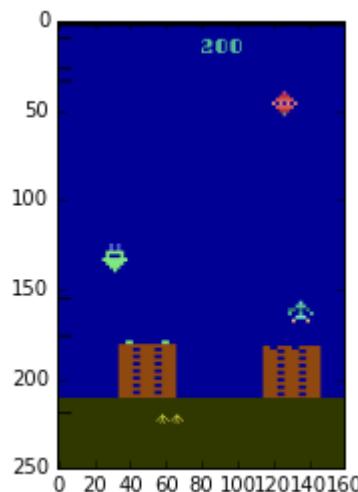


Reality check: dynamic systems

- **Cases:**
 - Robots
 - Self-driving vehicles
 - Pilot assistant
 - More robots!
- **Example**
 - Observation: sensor feed
 - Action: voltage sent to motors
 - Feedback: how far did it move forward before falling

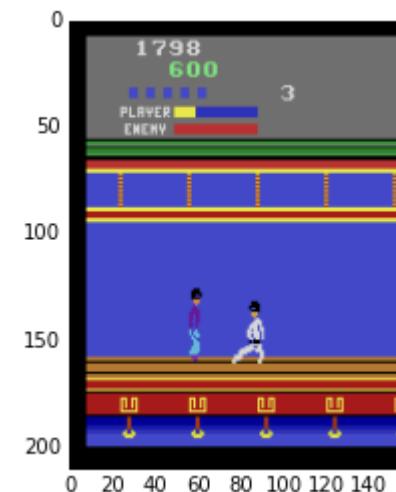
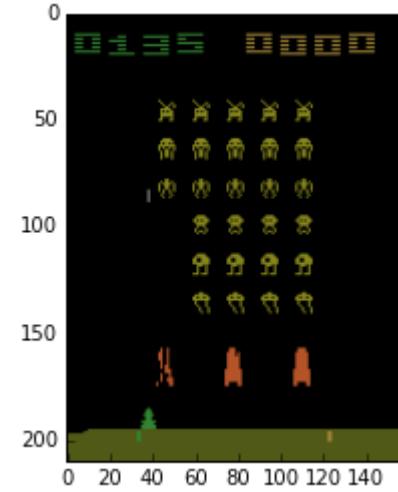
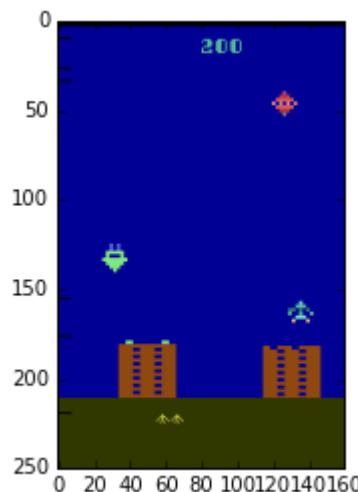


Reality check: videogames



- Q: What are observations, actions and feedback?

Reality check: videogames



- Q: What are observations, actions and feedback?

Other use cases

- Personalized medical treatment



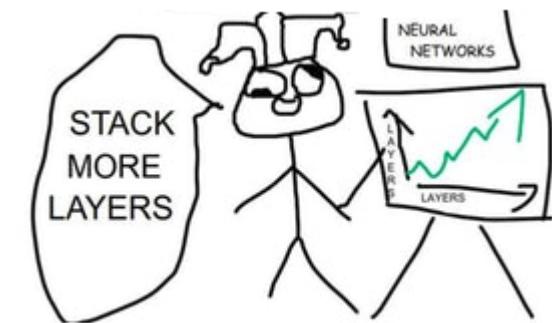
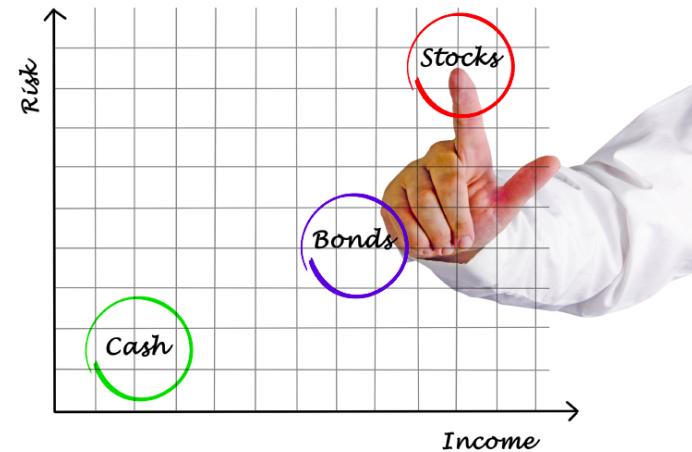
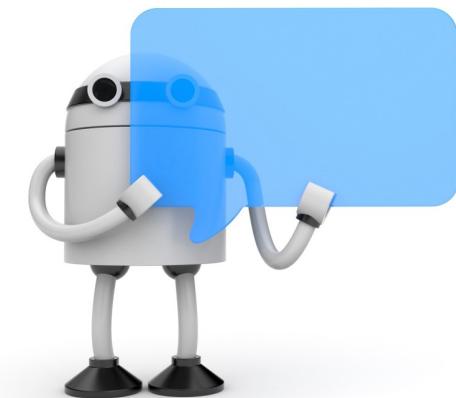
- Even more games (Go, chess, etc)



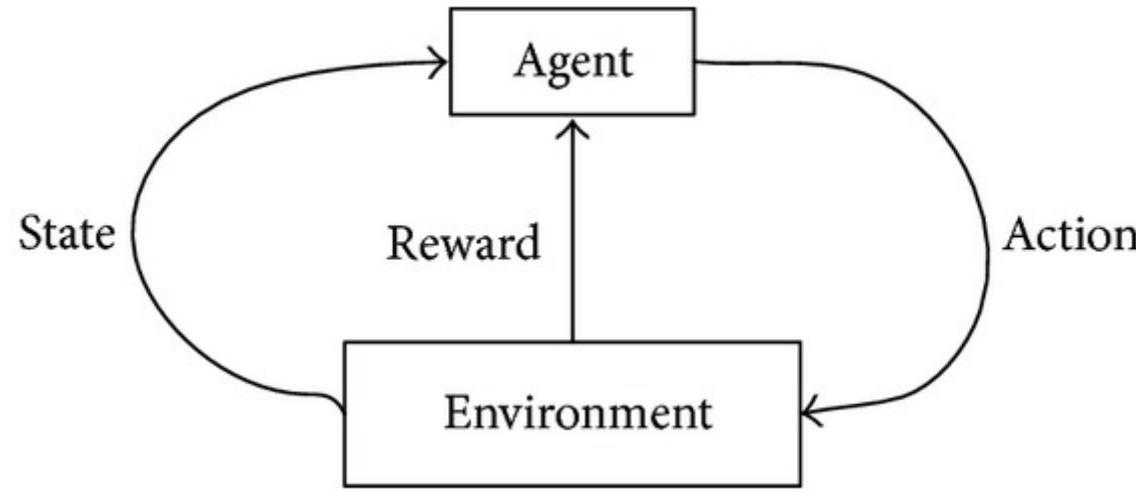
- Q: What are observations, actions and feedback?

Other use cases

- Conversation systems
 - learning to make user happy
- Quantitative finance
 - portfolio management
- Deep learning
 - optimizing non-differentiable loss
 - finding optimal architecture



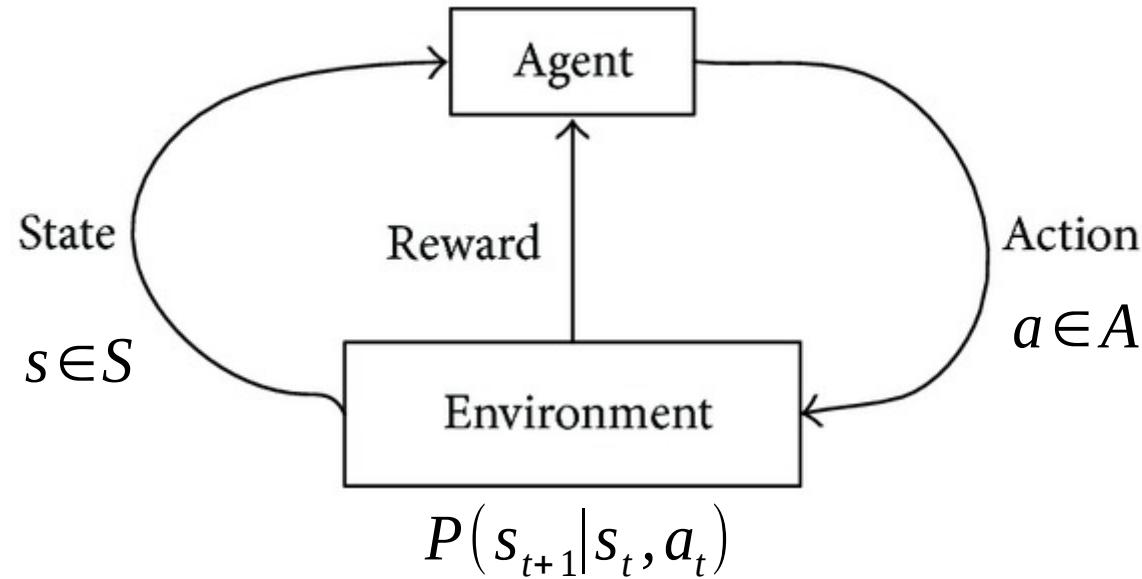
The MDP formalism



Markov Decision Process

- Environment states: $s \in S$
- Agent actions: $a \in A$
- Rewards $r \in \mathbb{R}$
- Dynamics: $P(s_{t+1} | s_t, a_t)$

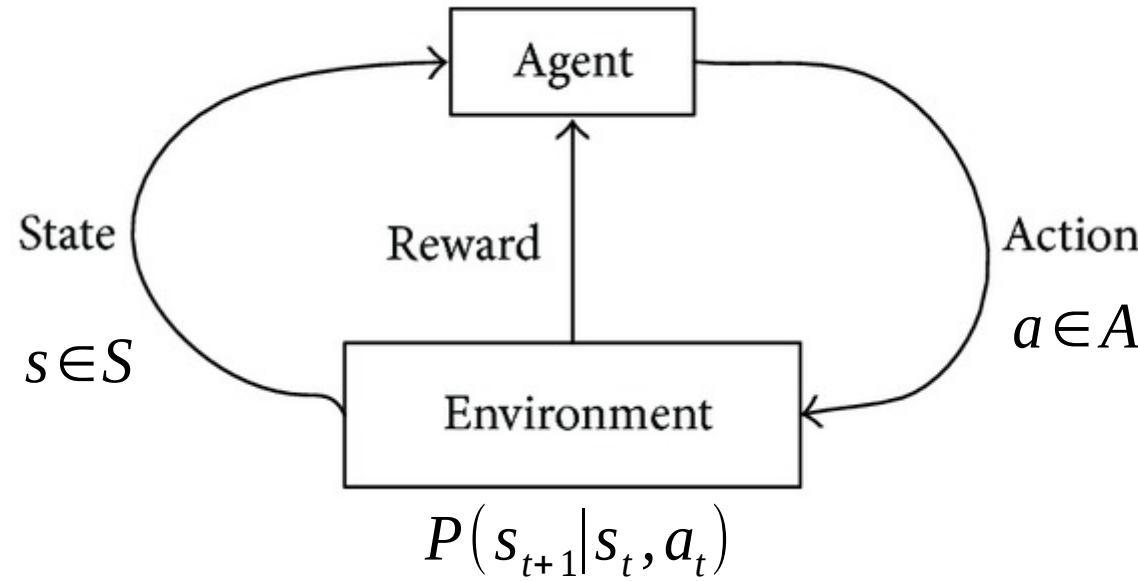
The MDP formalism



Markov Decision Process
Markov assumption

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}) = P(s_{t+1}|s_t, a_t)$$

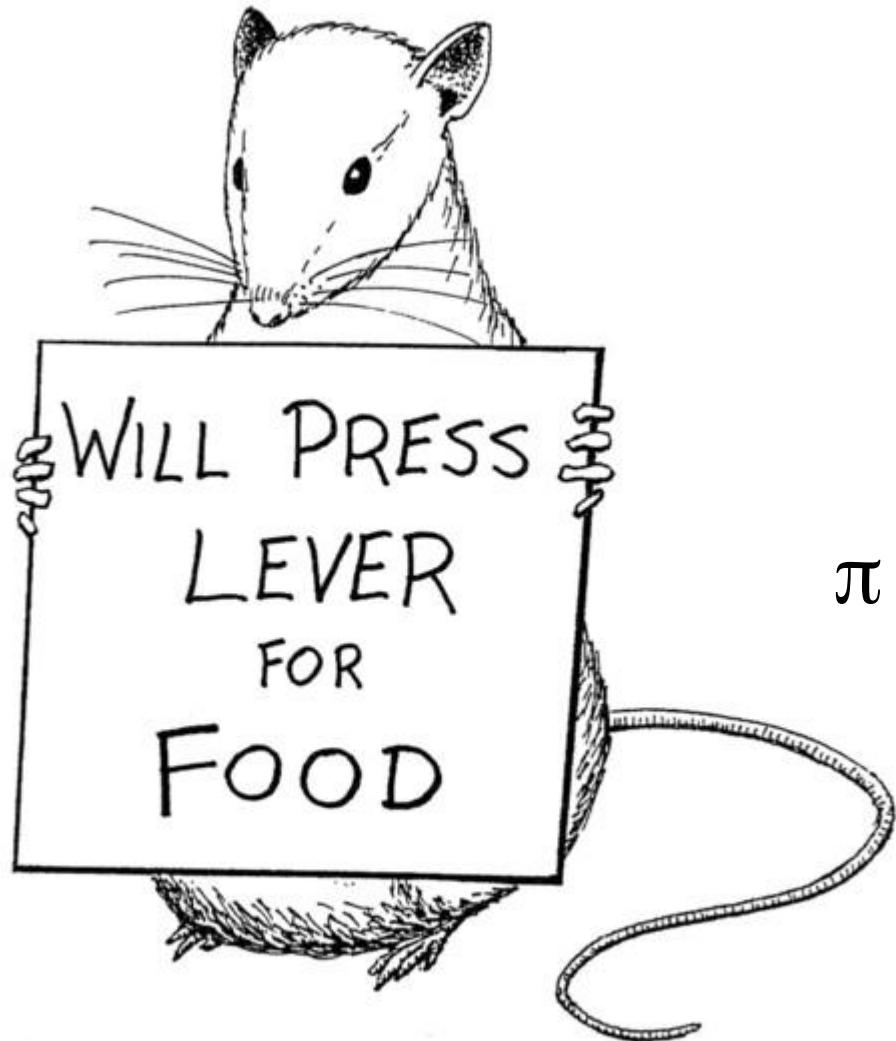
The MDP formalism



Markov Decision Process
Markov assumption

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}) = P(s_{t+1}|s_t, a_t)$$

Total reward



Total reward for session:

$$R = \sum_t r_t$$

Agent's policy:

$$\pi(a|s) = P(\text{take action } a \text{ in state } s)$$

Problem: find policy with highest reward:

$$\pi(a|s) : E_{\pi}[R] \rightarrow \max$$

Objective

The easy way:

$E_{\pi} R$ is an expected sum of rewards
that agent with policy π earns per session

Objective

The easy way:

$E_{\pi} R$ is an expected sum of rewards
that agent with policy π earns per session

The hard way:

$$E \quad E \quad E \quad \dots \quad E \quad [r_0 + r_1 + r_2 + \dots + r_T]$$
$$s_0 \sim p(s_0), a_0 \sim \pi(a|s_0), s_1, r_0 \sim P(s', r | s, a) \quad s_T, r_T \sim P(s', r | s_{T-1}, a_{t-1})$$

How do we solve it?

General idea:

Play a few sessions

Update your policy

Repeat

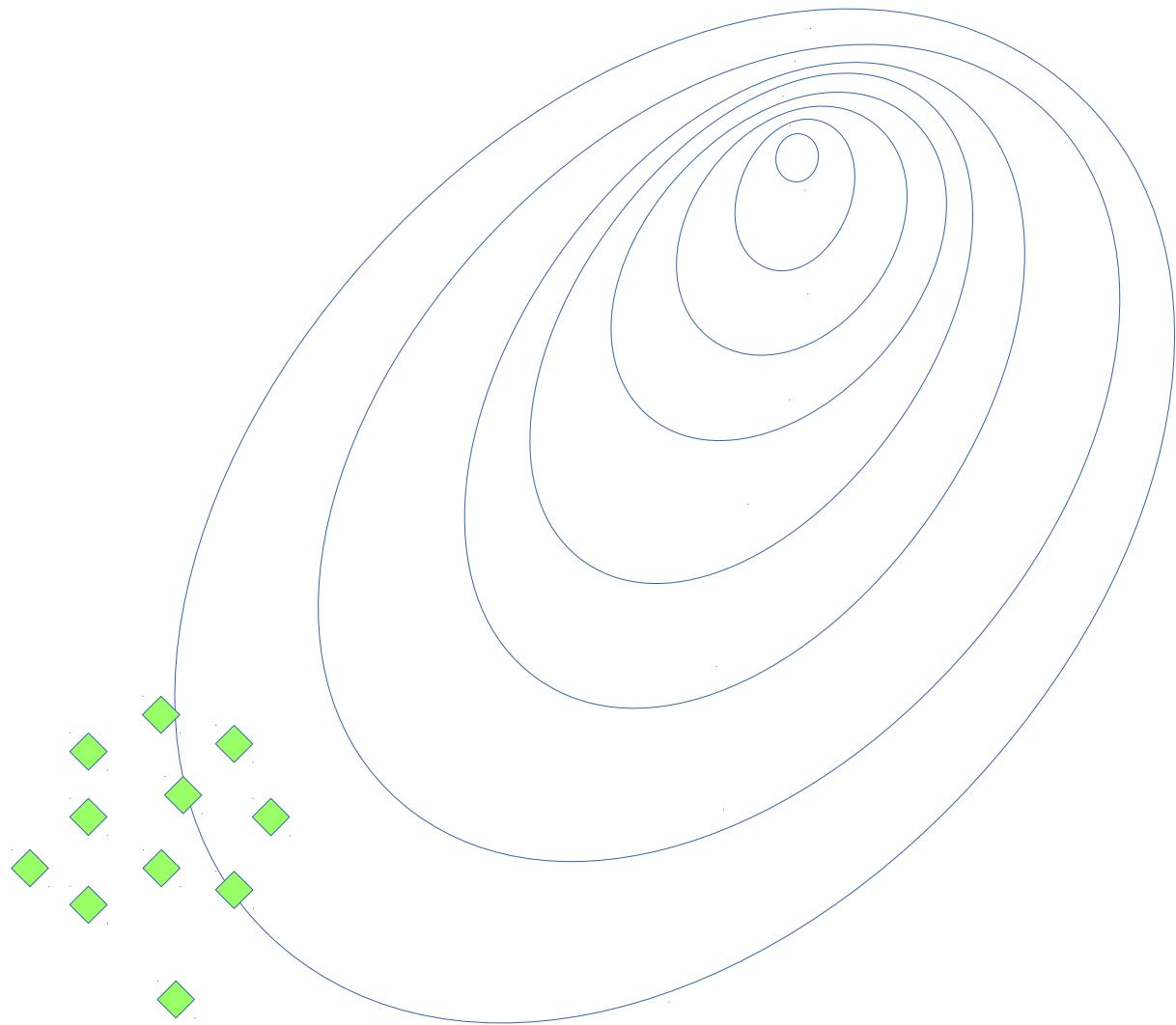
Crossentropy method

Initialize policy

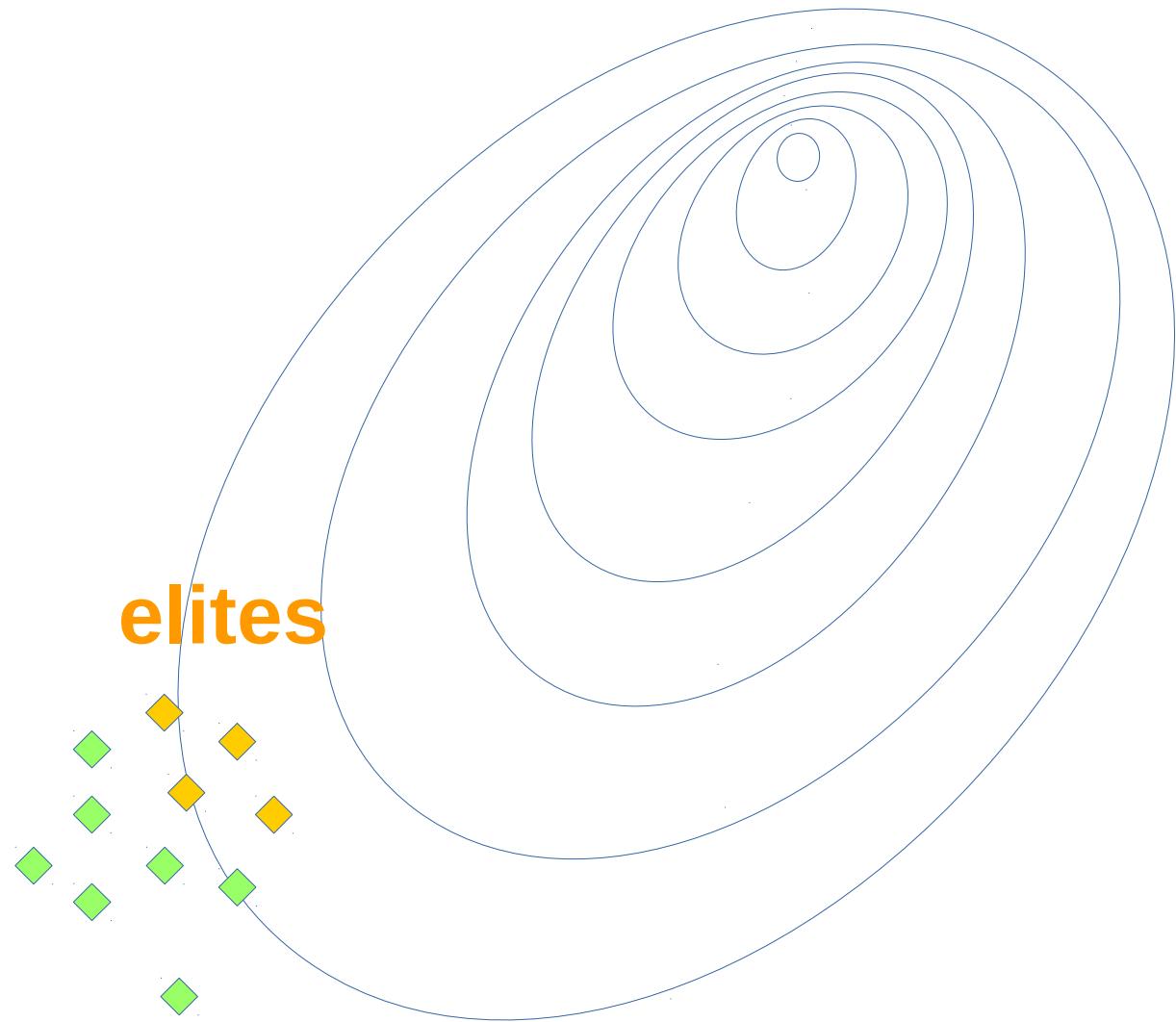
Repeat:

- Sample $N[100]$ sessions
- Pick $M[25]$ best sessions, called **elite** sessions
- Change policy so that it prioritizes actions from elite sessions

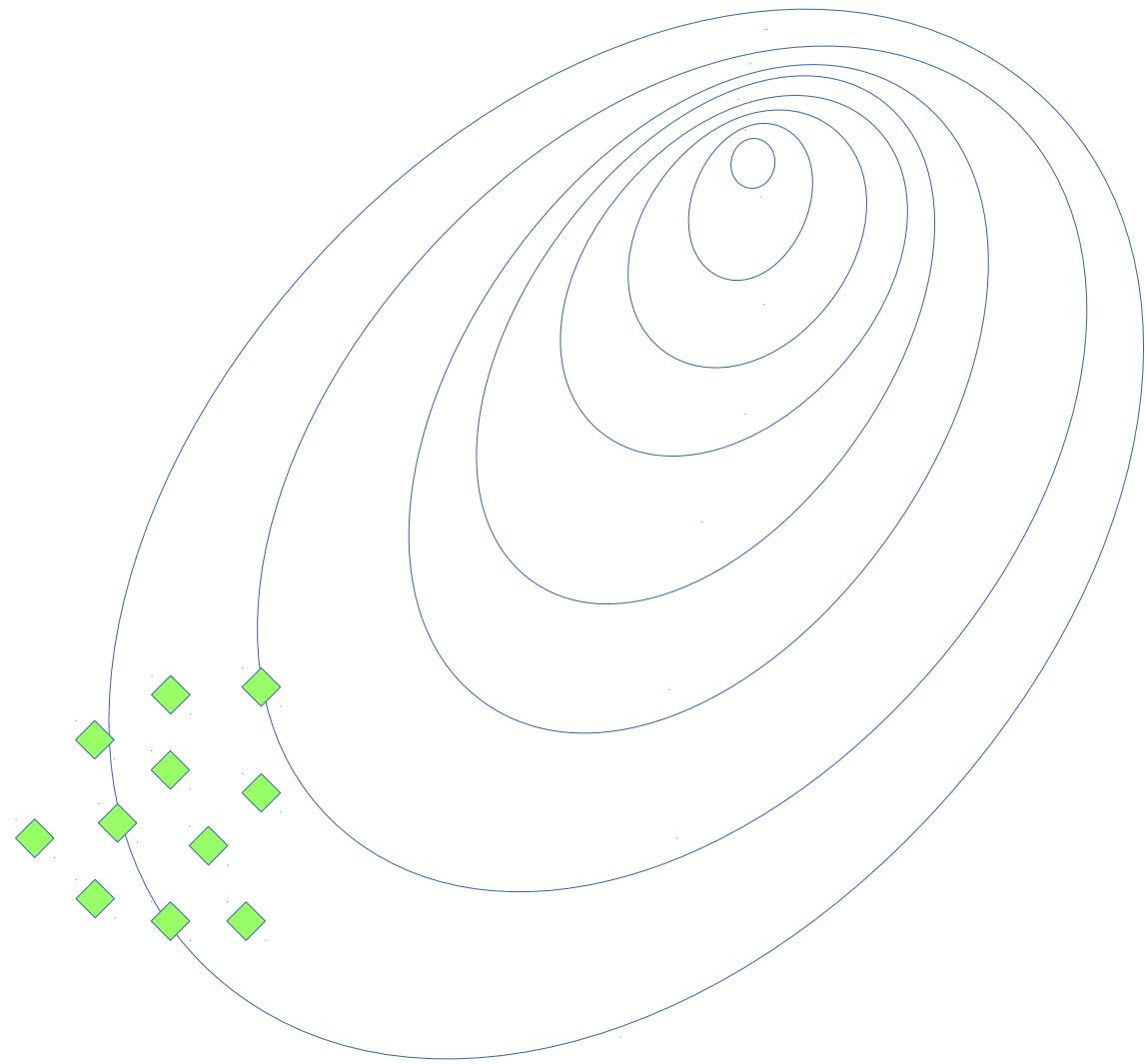
Step-by-step view



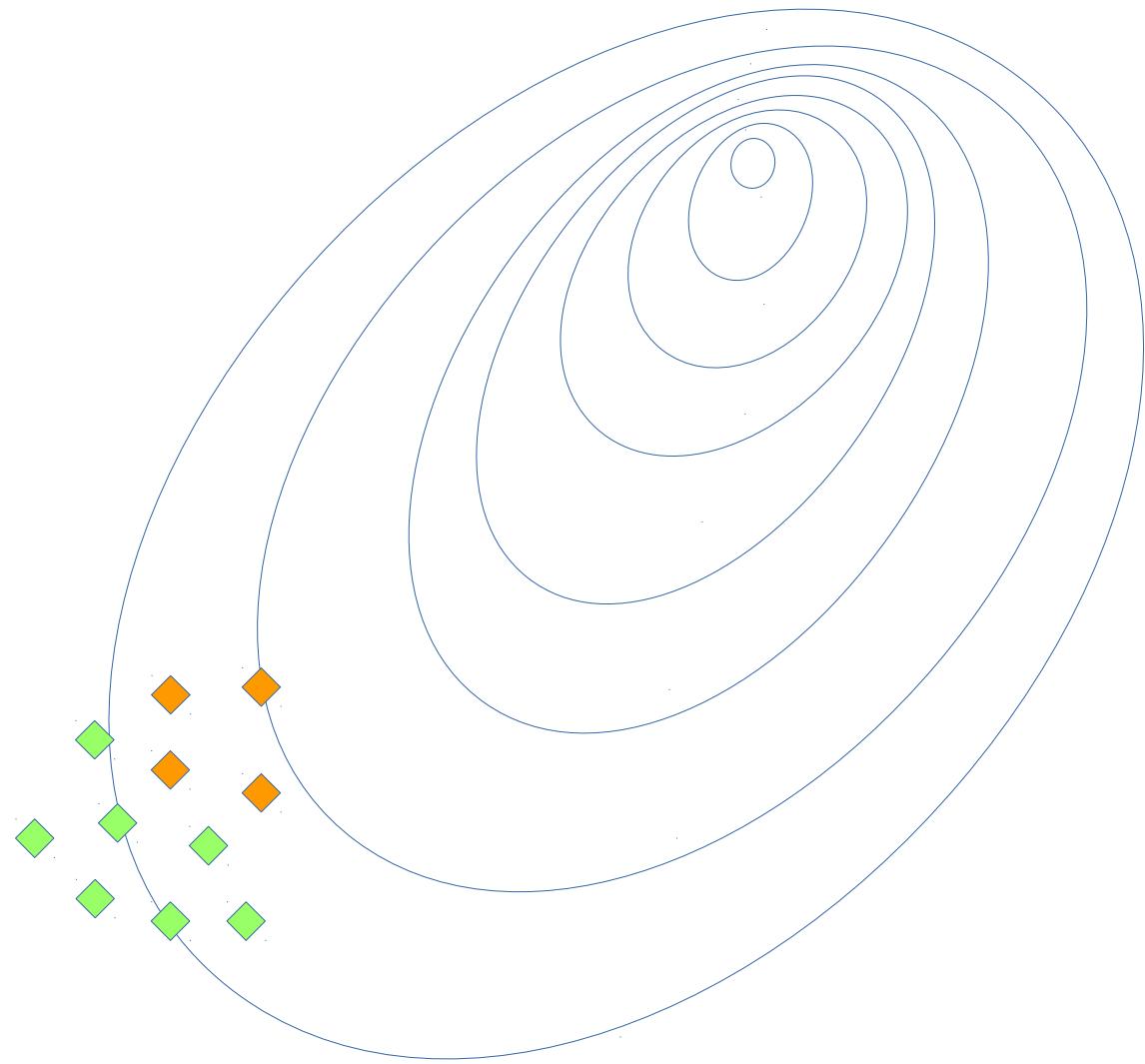
Step-by-step view



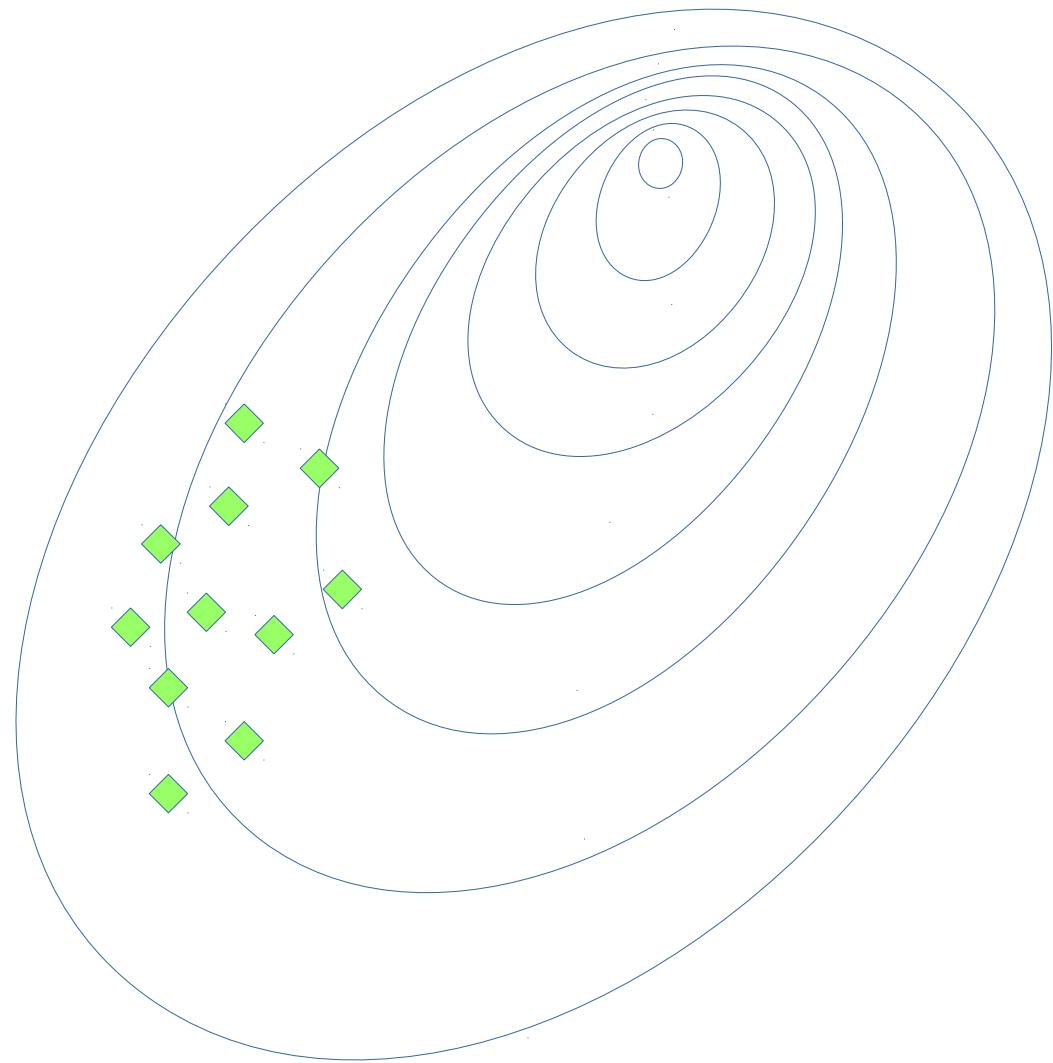
Step-by-step view



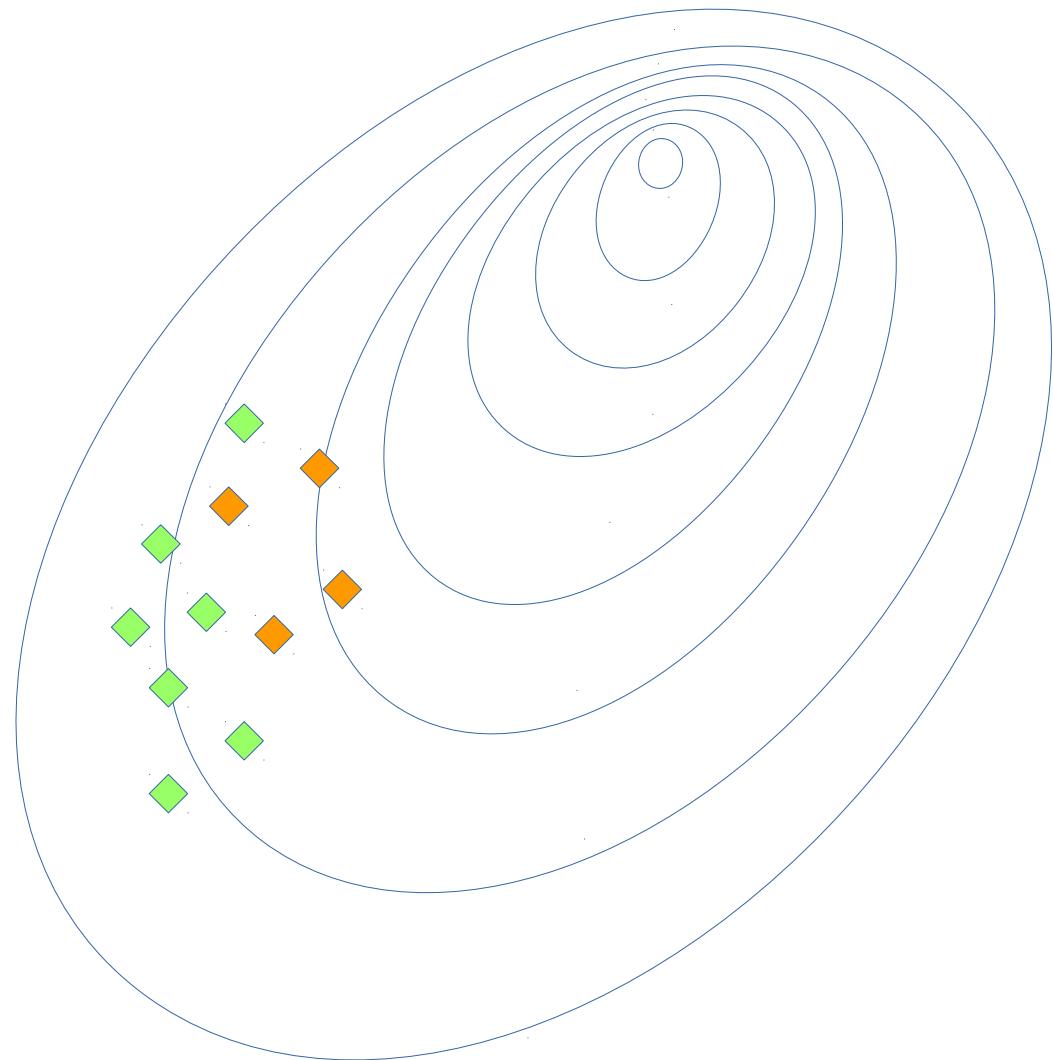
Step-by-step view



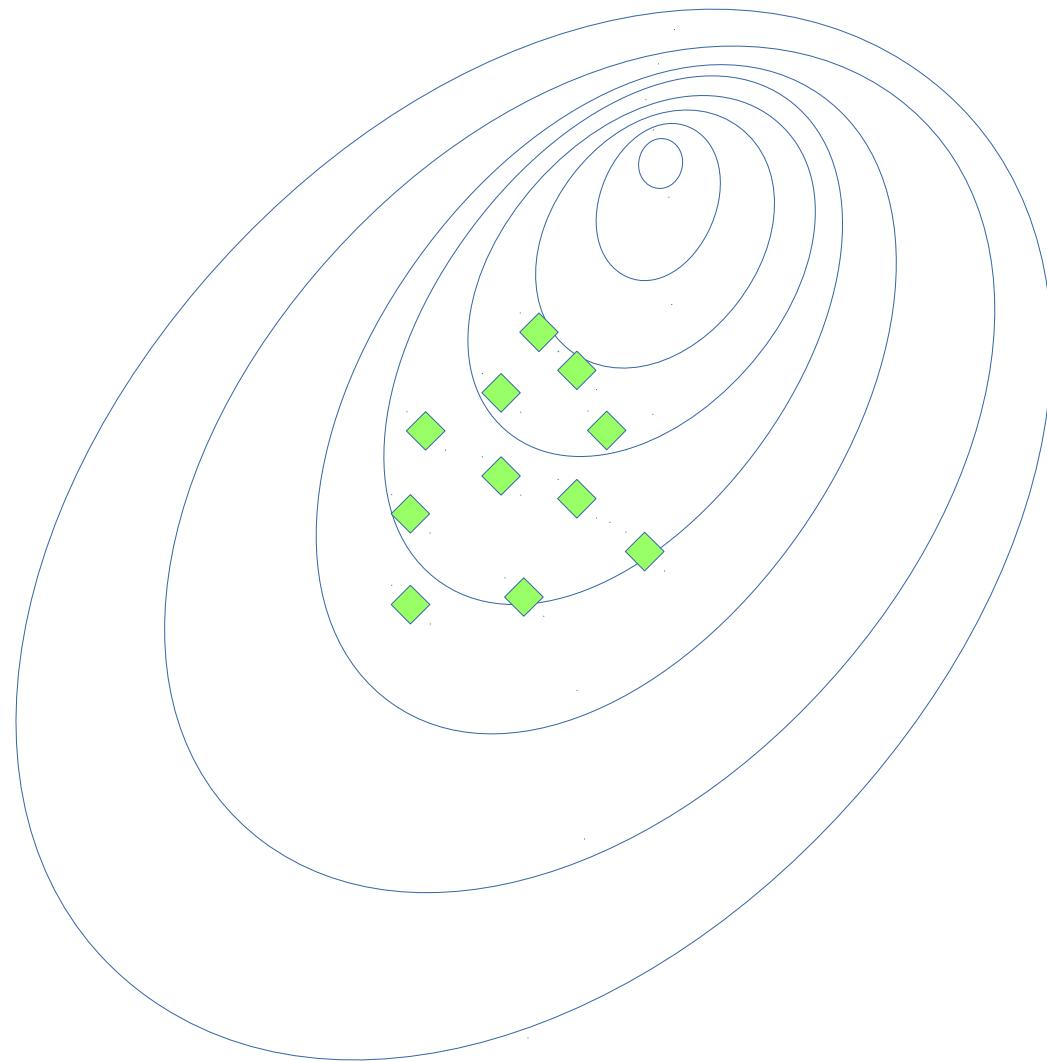
Step-by-step view



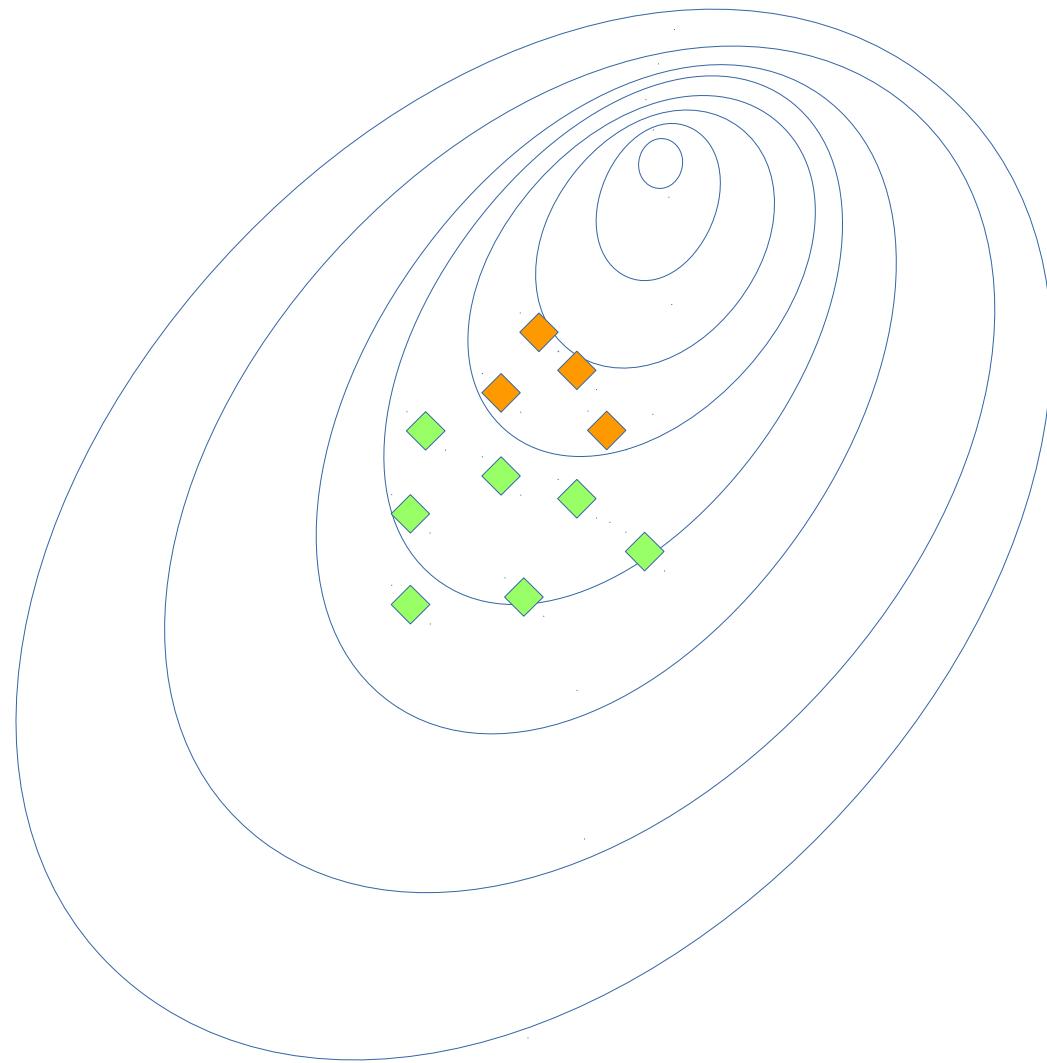
Step-by-step view



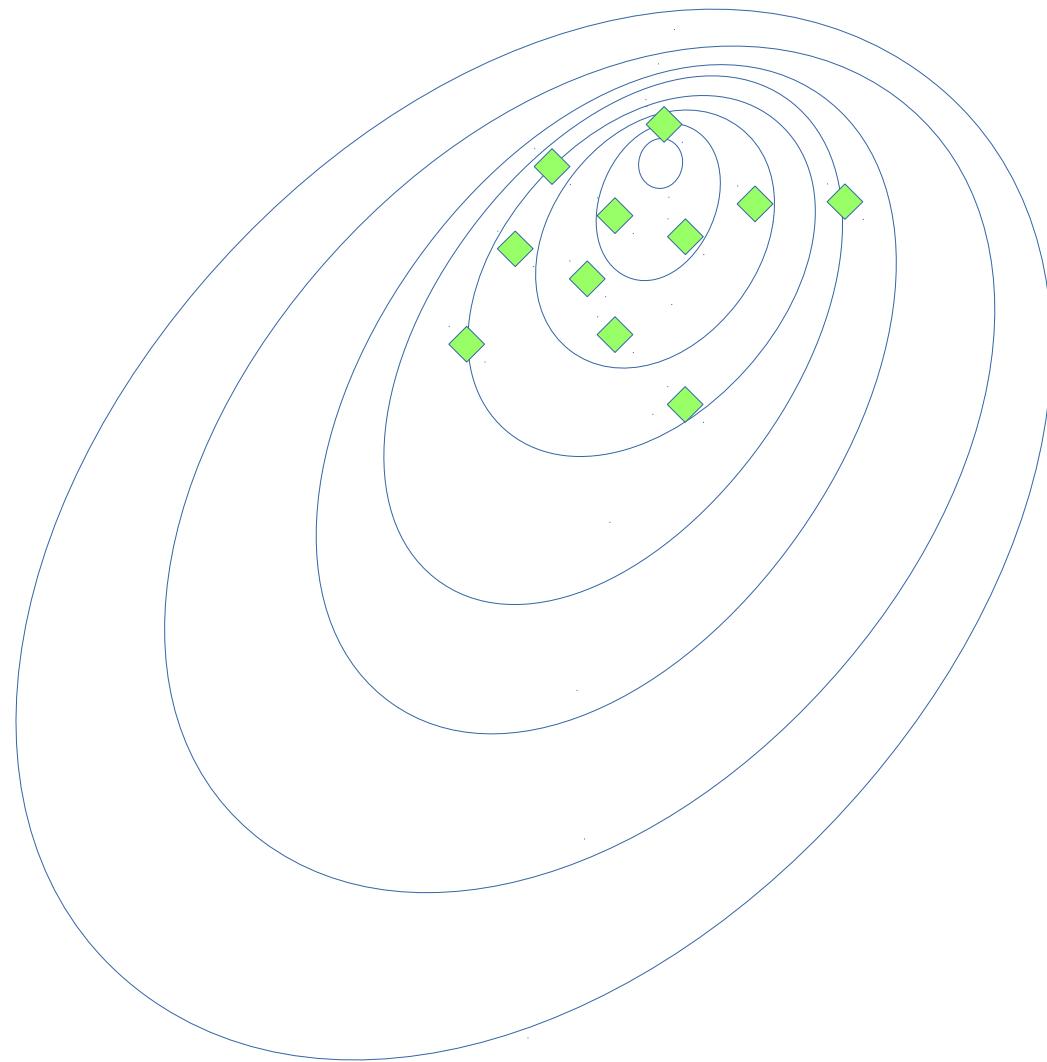
Step-by-step view



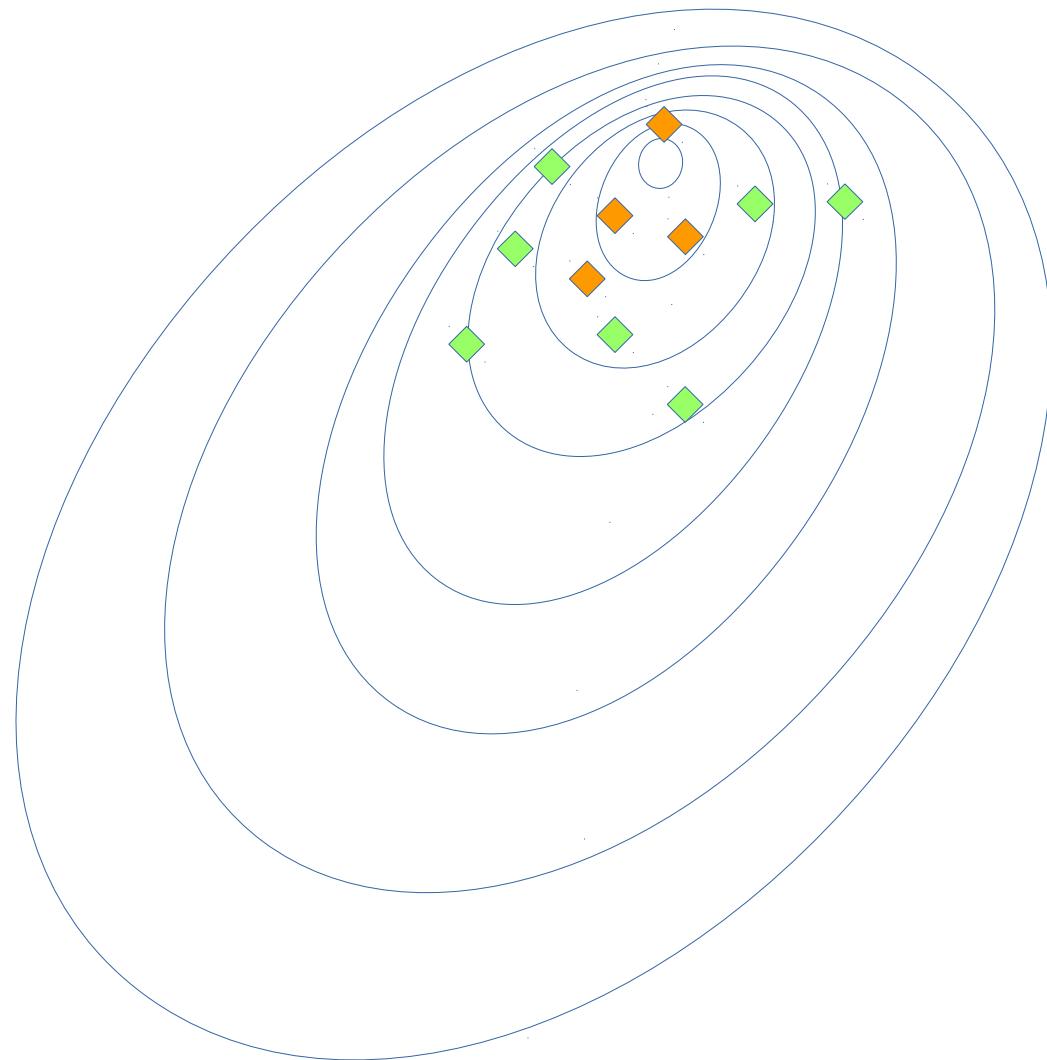
Step-by-step view



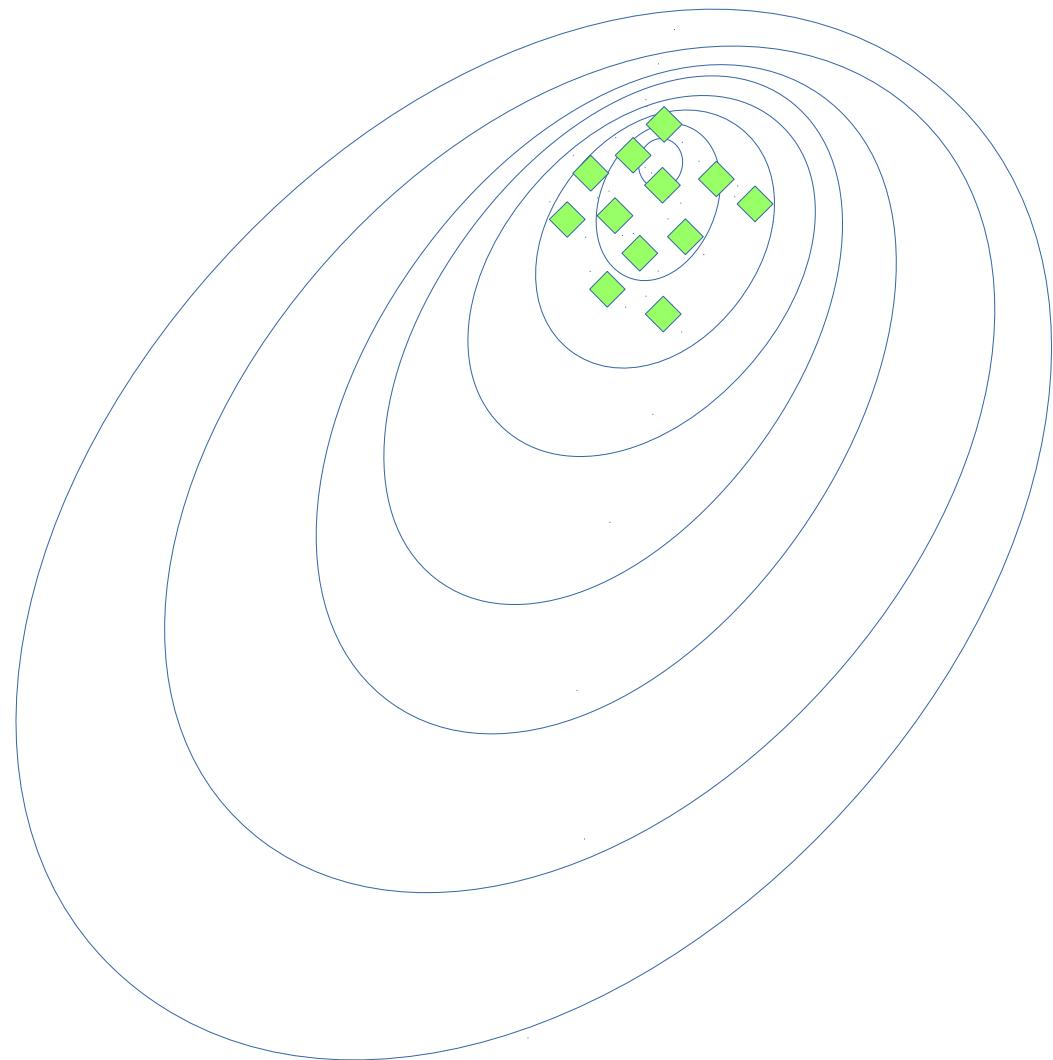
Step-by-step view



Step-by-step view



Step-by-step view



Tabular crossentropy method

- Policy is a matrix

$$\pi(a|s) = A_{s,a}$$

- Sample N games with that policy
- Get M best sessions (elites)

$$Elite = [(s_0, a_0), (s_1, a_1), (s_2, a_2), \dots, (s_k, a_k)]$$

Tabular crossentropy method

- Policy is a matrix

$$\pi(a|s) = A_{s,a}$$

- Sample N games with that policy
- Take M best sessions (elites)
- Aggregate by states

$$\pi(a|s) = \frac{\sum_{s_t, a_t \in Elite} [s_t = s][a_t = a]}{\sum_{s_t, a_t \in Elite} [s_t = s]}$$

Tabular crossentropy method

- Policy is a matrix

$$\pi(a|s) = A_{s,a}$$

- Sample N games with that policy
- Take M best sessions (elite)
- Aggregate by states

$$\pi(a|s) = \frac{\text{took } a \text{ at } s}{\text{was at } s}$$

In M best games

Grim reality

If your environment has infinite/large state space



Approximate crossentropy method

- Policy is approximated
 - Neural network predicts $\pi_w(a|s)$ given s
 - Linear model / Random Forest / ...

Can't set $\pi(a|s)$ explicitly

All state-action pairs from M best sessions

$$Elite = [(s_0, a_0), (s_1, a_1), (s_2, a_2), \dots, (s_k, a_k)]$$

Approximate crossentropy method

Neural network predicts $\pi_W(a|s)$ given s

All state-action pairs from M best sessions

$$Elite = [(s_0, a_0), (s_1, a_1), (s_2, a_2), \dots, (s_k, a_k)]$$

Maximize likelihood of actions in “best” games

$$\pi = \operatorname{argmax}_{\pi} \sum_{s_i, a_i \in Elite} \log \pi(a_i | s_i)$$

Approximate crossentropy method

- Initialize NN weights $W_0 \leftarrow random$
- Loop:
 - Sample N sessions
 - $Elite = [(s_0, a_0), (s_1, a_1), (s_2, a_2), \dots, (s_k, a_k)]$
 - $W_{i+1} = W_i + \alpha \nabla [\sum_{s_i, a_i \in Elite} \log \pi_{W_i}(a_i | s_i)]$

Approximate crossentropy method

- Initialize NN weights **nn = MLPClassifier(...)**
- Loop:
 - Sample N sessions
 - $Elite = [(s_0, a_0), (s_1, a_1), (s_2, a_2), \dots, (s_k, a_k)]$
 - **nn.fit(elite_states,elite_actions)**

Continuous action spaces

- Continuous state space
- Model $\pi_W(a|s) = N(\mu(s), \sigma^2)$
 - Mu(s) is neural network output
 - Sigma is a parameter or yet another network output
- Loop:
 - Sample N sessions
 - elite = take M best sessions and concatenate
 - $W_{i+1} = W_i + \alpha \nabla \left[\sum_{s_i, a_i \in Elite} \log \pi_{W_i}(a_i | s_i) \right]$

What changed?

Approximate crossentropy method

- Initialize NN weights **nn = MLPRegressor(...)**
- Loop:
 - Sample N sessions
 - $Elite = [(s_0, a_0), (s_1, a_1), (s_2, a_2), \dots, (s_k, a_k)]$
 - **nn.fit(elite_states,elite_actions)**

Almost nothing!

Tricks

- Remember sessions from 3-5 past iterations
 - Threshold and use all of them when training
 - May converge slower if env is easy to solve.
- Regularize with entropy
 - to prevent premature convergence.
- Parallelize sampling
- Use RNNs if partially-observable (later)





**Q: How is RL different
from supervised learning?**

What-what learning?

Supervised learning

- Learning to approximate reference answers
- Needs correct answers
- Model does not affect the input data

Reinforcement learning

- Learning optimal strategy by trial and error
- Needs feedback on agent's own actions
- Agent can affect it's own observations



What-what learning?

Unsupervised learning

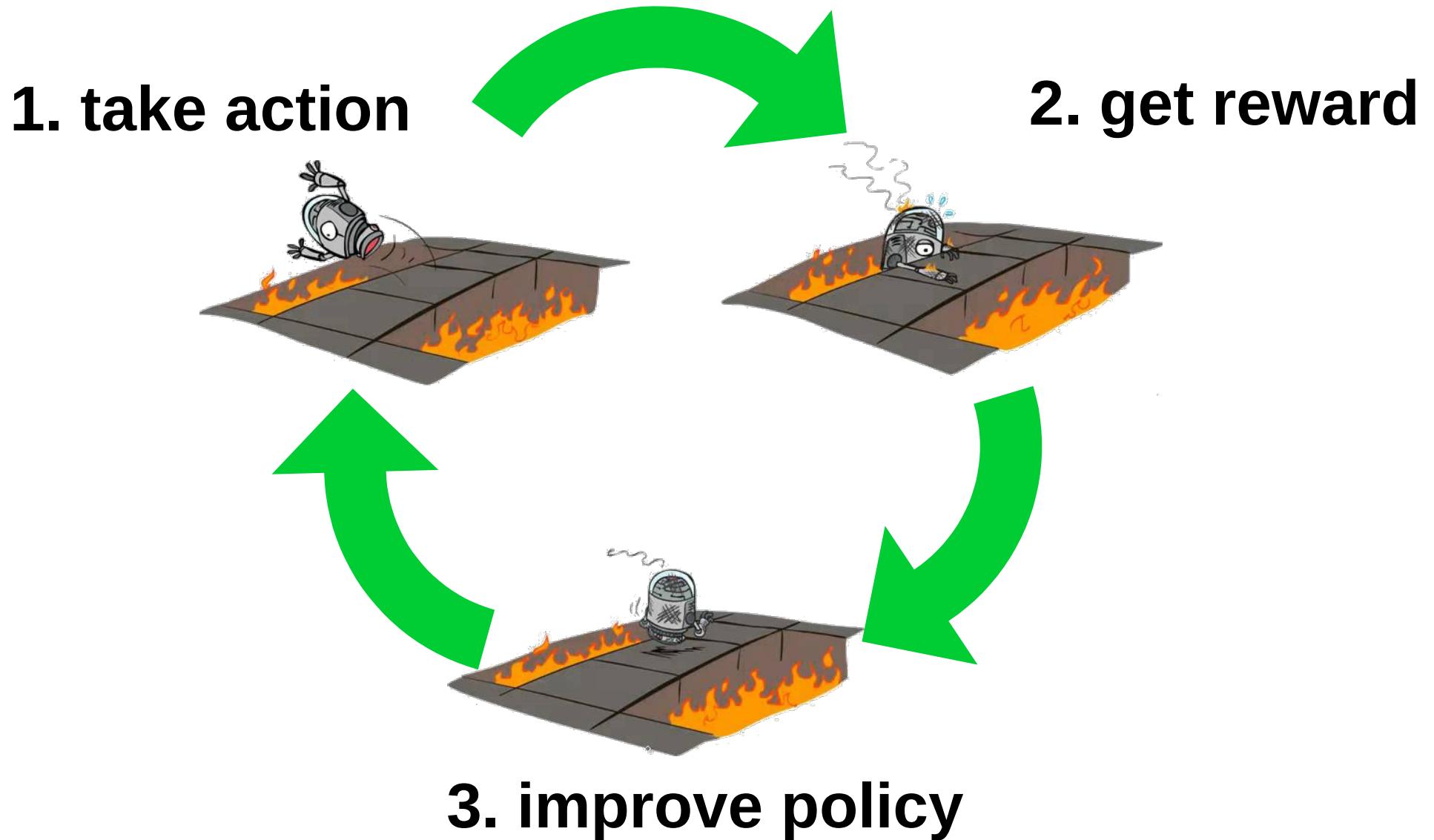
- Learning underlying data structure
- No feedback required
- Model does not affect the input data

Reinforcement learning

- Learning optimal strategy by trial and error
- Needs feedback on agent's own actions
- Agent can affect its own observations



Reinforcement learning is easy!



Reinforcement learning is challenging!

How to explore?

1. take action

How not to break anything when acting?

Continuous actions?

How to define policy model?

When not to use RL?

take supervised data into account?

did my algorithm overfit to simulation?

What if there's many actions?

How to infer actions from game image?



Which actions caused reward?

Sparse rewards?



Can agent cheat reward?

2. get reward

Which rewards are easier to learn?

What if there are multiple agents?

What if observations are incomplete?

How do I formulate my problem for RL?



best way to learn from limited data?

3. improve policy

How do I know if it converged or not?

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
 - Papers from '16 to '19
- **Almost no** planning (aka model-based RL)
- **Brief** on monte-carlo RL and bandits