

Reinforcement learning week 1

Why should you care



Yandex
Data Factory

LAMBDA 



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

Find:

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

[banner,page], ctr

linear/ tree / NN

MSE, crossentropy



Supervised learning



Supervised learning

Great... except if we have no reference answers



Great... except if we have no reference answers

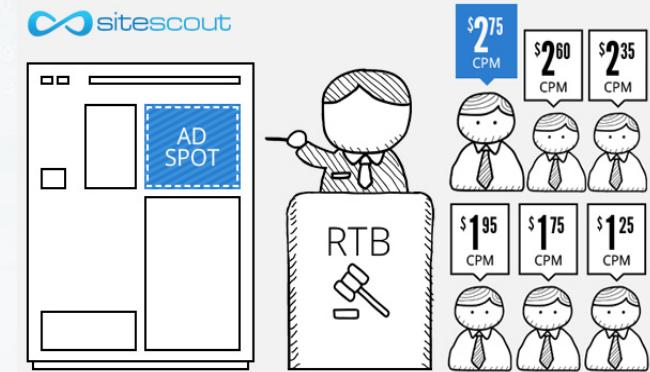


Online Ads

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



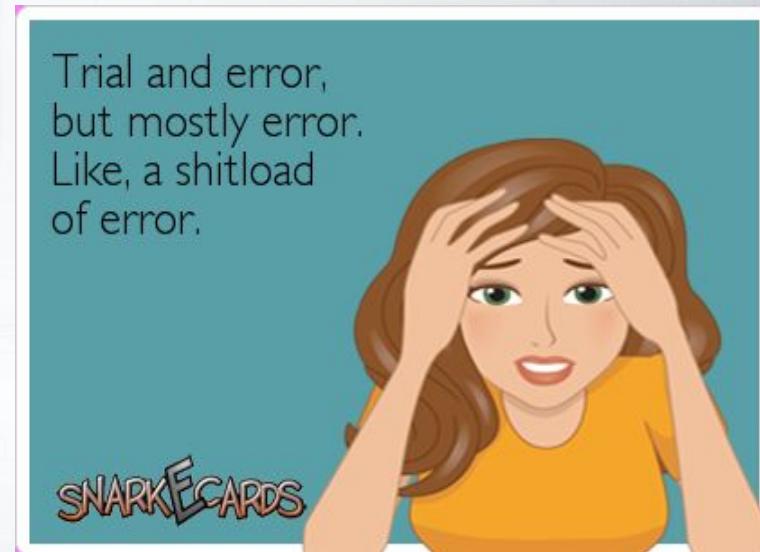
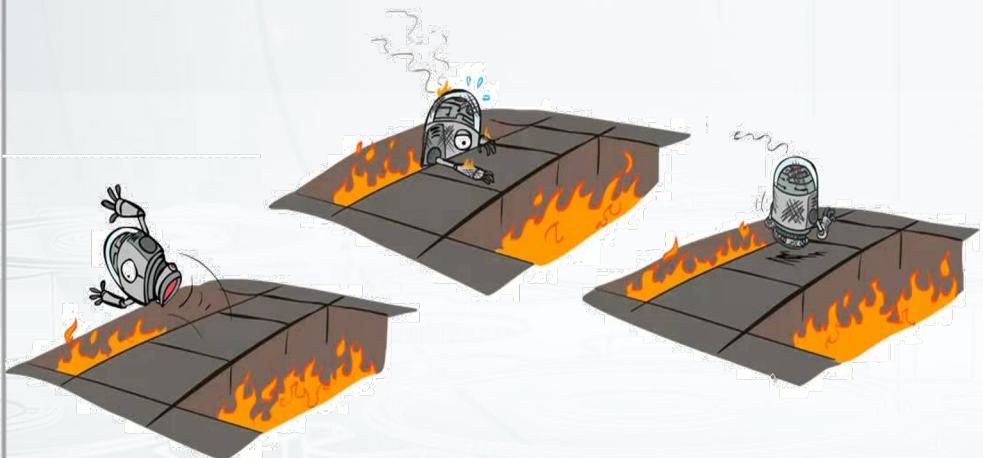
Ideas?



Duct tape approach

Common idea:

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



Advertone, <http://www.pvsm.ru/digital/68355>



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

IF IN CAN'T BE FIXED E
WITH DUCT TAPE



THEN YOU'RE NOT USING
ENOUGH DUCT TAPE



Reinforcement learning

STAND BACK



**I'M GOING TO TRY
SCIENCE**



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?

1,00000000000
0.50000000000
0.00000000000
-0.50000000000
-1,00000000000

▲ 60; 0.8178592143
▲ 80; 0.8945486389
◆ 60; 0.7883362066
■ 30; 0.6499716055
■ 30; 0.6398330060
◆ 30; 0.4322583897
■ 120; 0.5853979786
■ 120; 0.374393697
■ 120; 0.374393697

Reinforcement learning

360; 0.00000000000
300; 0.00000000000
240; 0.00000000000
180; 0.00000000000
120; 0.00000000000
60; 0.00000000000
0; 0.00000000000

Learning optimal strategy by trial and error

Needs feedback on agent's own actions

Model does not affect the input data

Agent can affect it's own observations



What-what learning?

▲ 60; 0.8178592143
▼ 80; 0.8945486389
◆ 60; 0.7883362086
■ 30; 0.6499716055 120; 0.8953970386

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 it's own observations



What is: bandit



observation

Agent

action

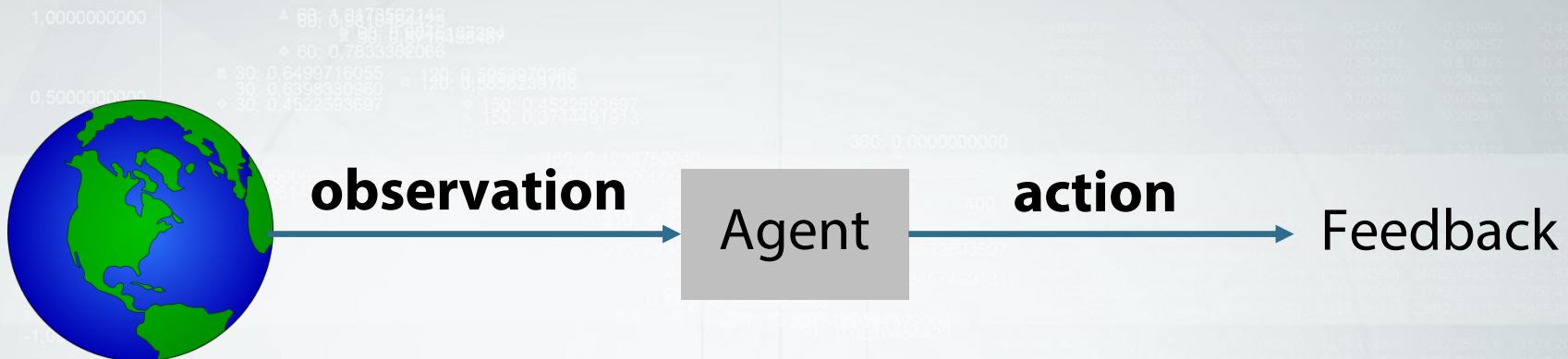
Feedback

Examples:

- banner ads (RTB)
- recommendations
- medical treatment

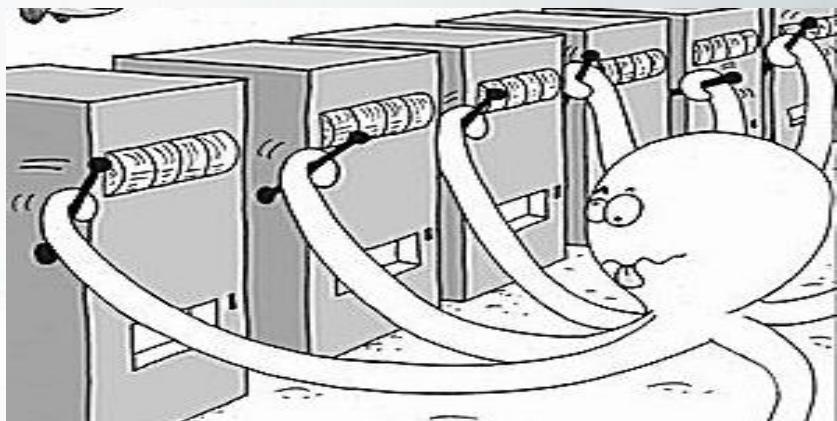


What is: bandit



Examples:

- banner ads (RTB)
- recommendations
- medical treatment



What is: bandit



observation

Agent

action

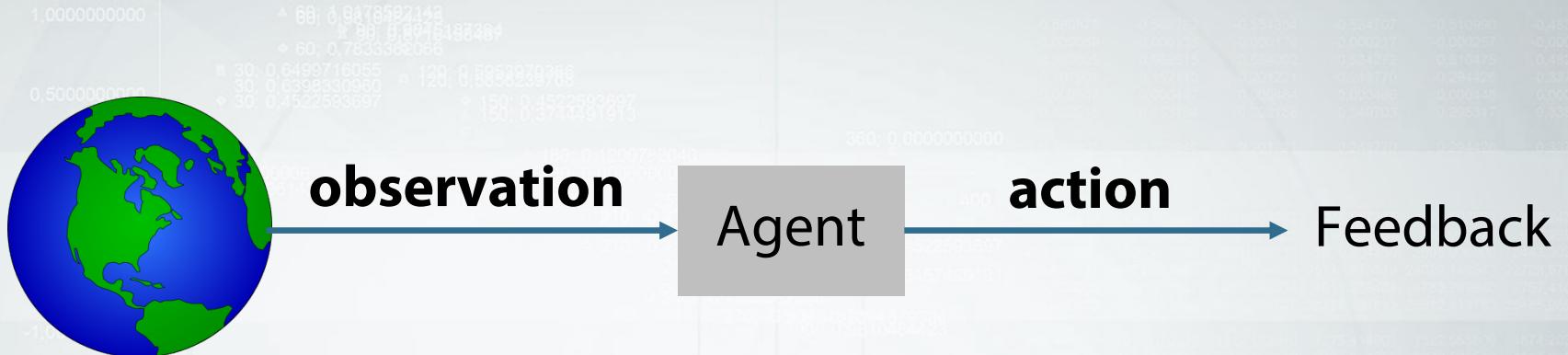
Feedback

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



observation

user features
time of year
trends

Agent

action

show
banner

Feedback
click,
money

Examples:

- banner ads (RTB)
- recommendations
- medical treatment



What is: bandit



observation

user features
time of year
trends

Agent

action

show
banner

Feedback

click,
money

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

Is it a good idea to show clickbait?



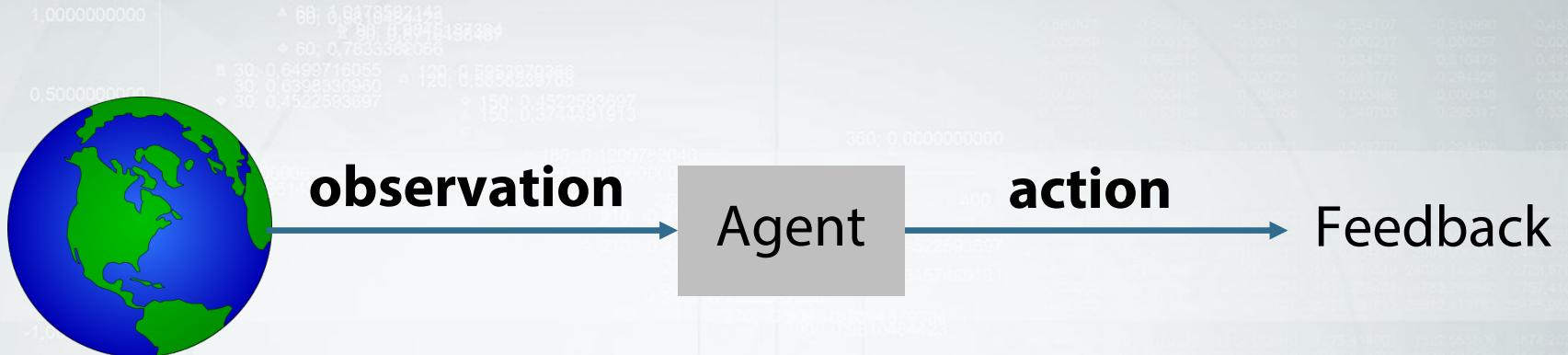
Who

Age

23 Celebrities You
Would Never Guess Are
Actually Black



What is: bandit



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

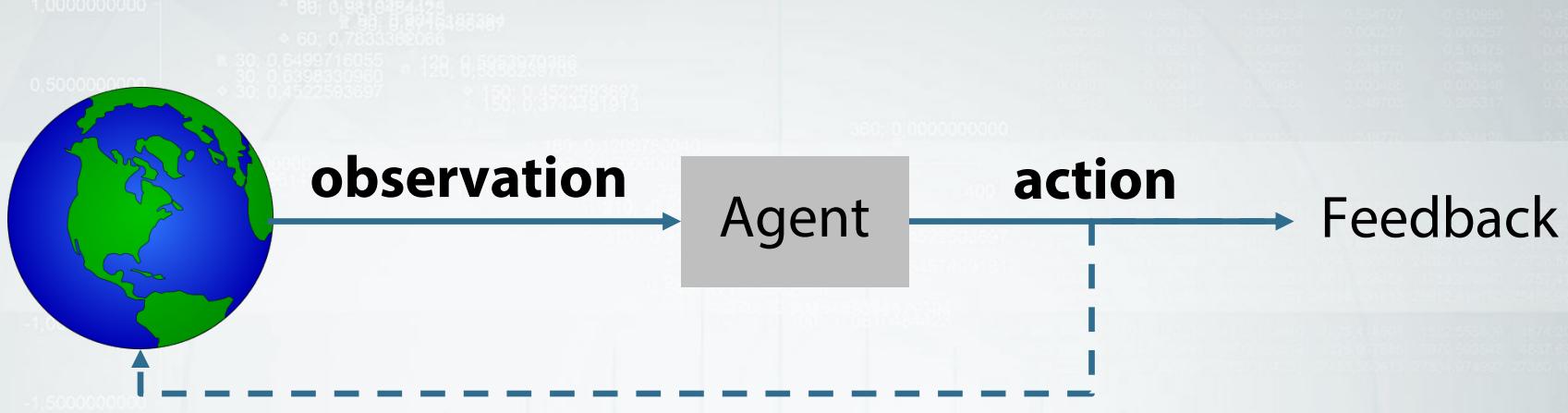
Is it a good idea to show clickbait?



Who 23 Celebrities You
Age Would Never Guess Are
Actually Black



What is: bandit



Q: You're Google/Yandex/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



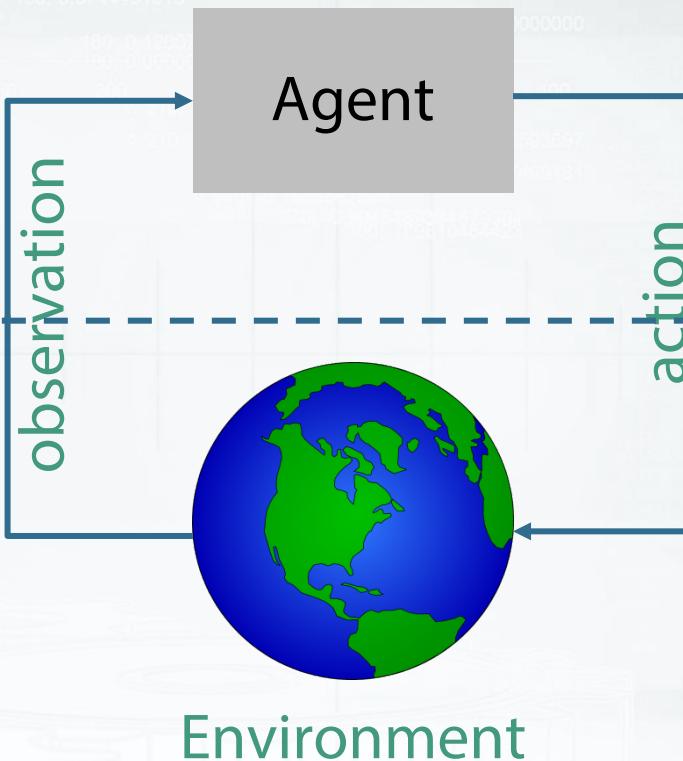
What is: decision process

1,500,000,000
1,000,000,000
500,000,000
0,000,000,000
-500,000,000

- ▲ 60; 0.9178592143
- ▲ 80; 0.8945486389
- ◆ 60; 0.7833362086
- 30; 0.6499716055
- 30; 0.6398330960
- ◆ 30; 0.4522583697
- 120; 0.5856329786
- 120; 0.3744391913
- ◆ 150; 0.4522583697
- 150; 0.3744391913

Can do anything

Can't even see
(worst case)



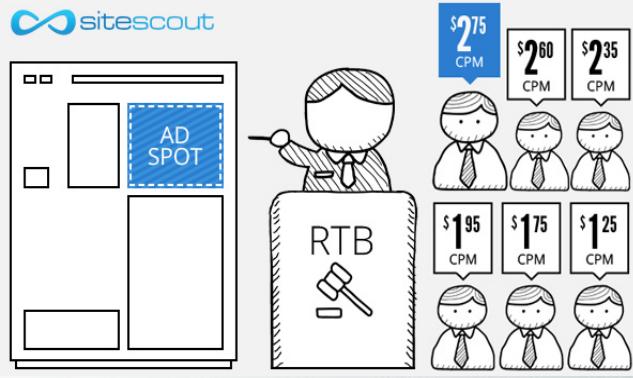
Reality check: web

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?



Old
Landing Page



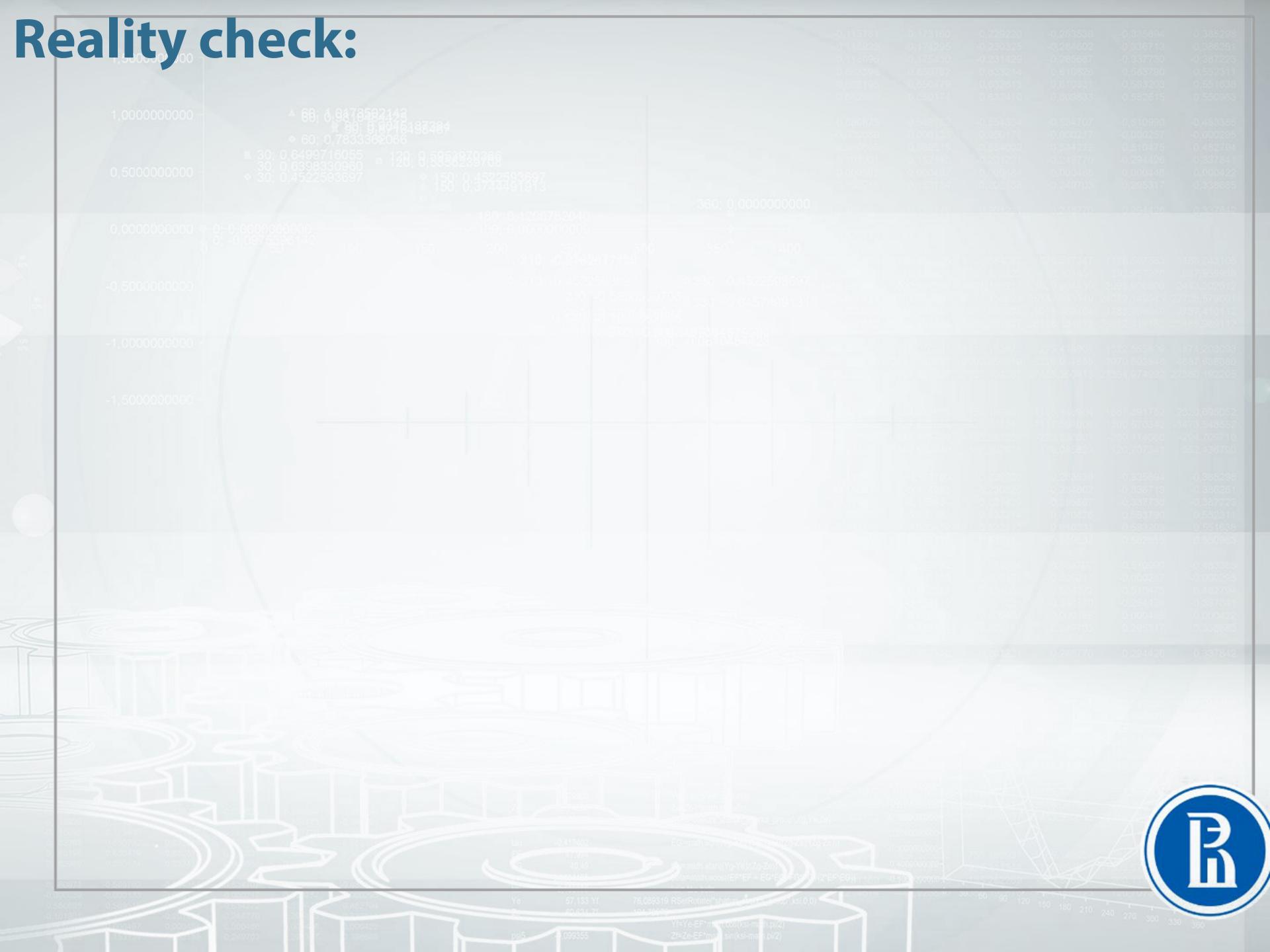
Conversion: 0.2%

Optimized
Landing Page



Conversion: 1.2%

Reality check:



Reality check: dynamic systems



Reality check: dynamic systems



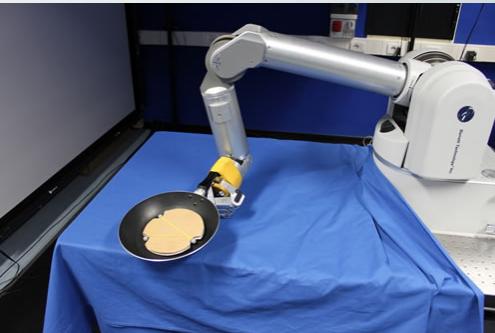
Reality check: dynamic systems

Cases:

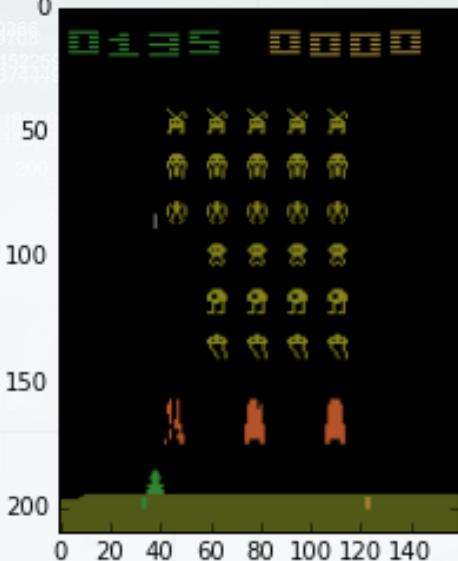
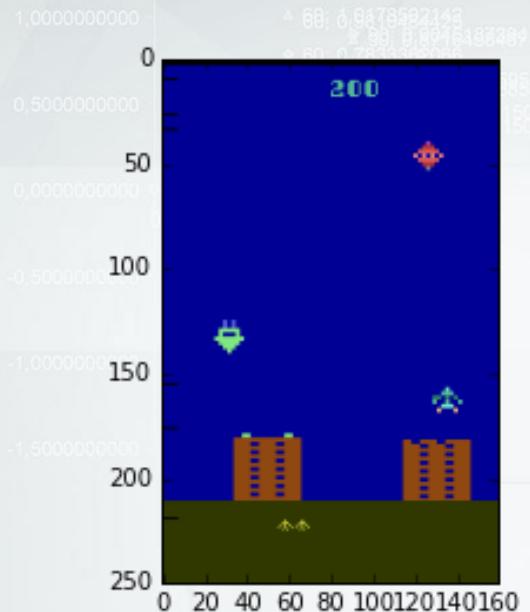
- Robots
- Self-driving vehicles
- Pilot assistant
- More robots!

Example:

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



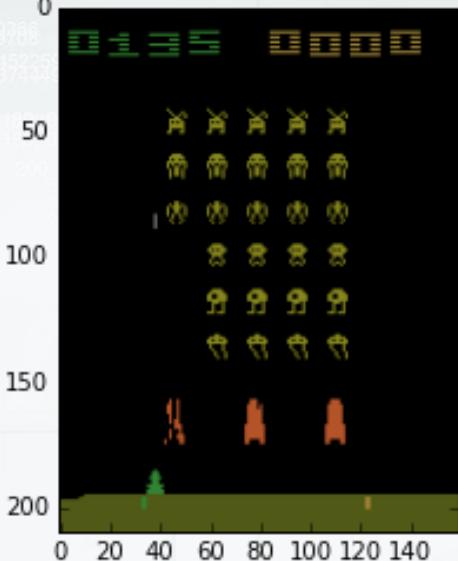
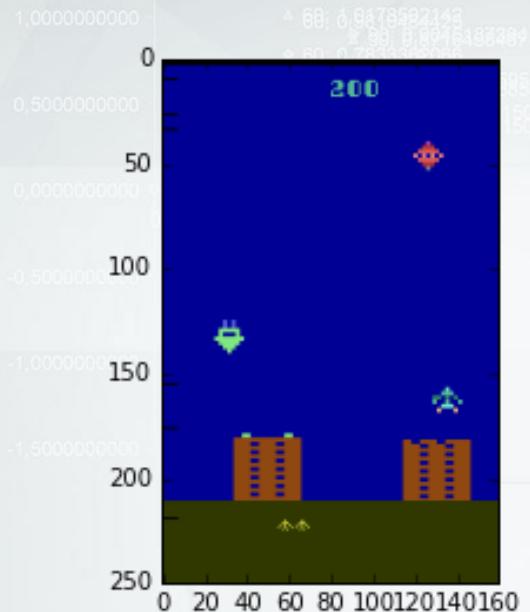
Reality check: videogames



Q: What are observations, actions and feedback?



Reality check: videogames



Q: What are observations, actions and feedback?



Other use cases

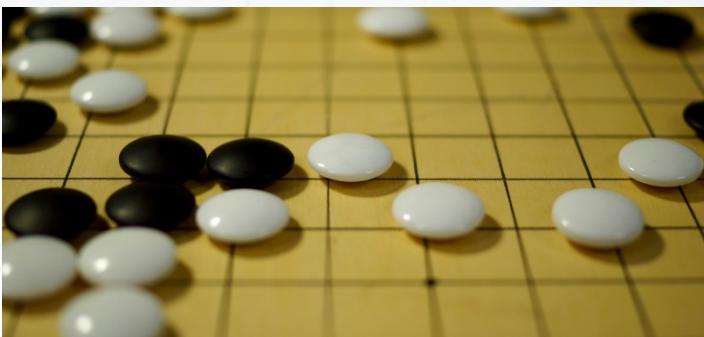
1.0000000000
0.5000000000
0.0000000000
-0.5000000000
-1.0000000000
-1.5000000000

- ▲ 60; 0.8178592142
- ▲ 60; 0.8945486304
- ◆ 60; 0.7883388686
- 30; 0.6499716055
- 30; 0.6398330960
- ◆ 30; 0.4522583697

Personalized medical treatment



Even more games (Go, chess, etc)



Q: What are observations, actions and feedback?

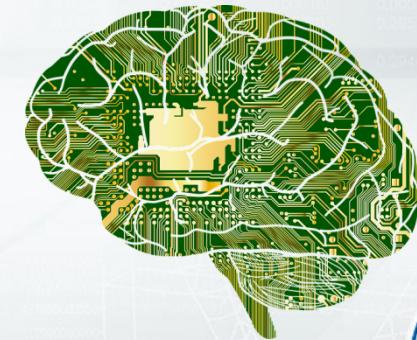


Other use cases



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



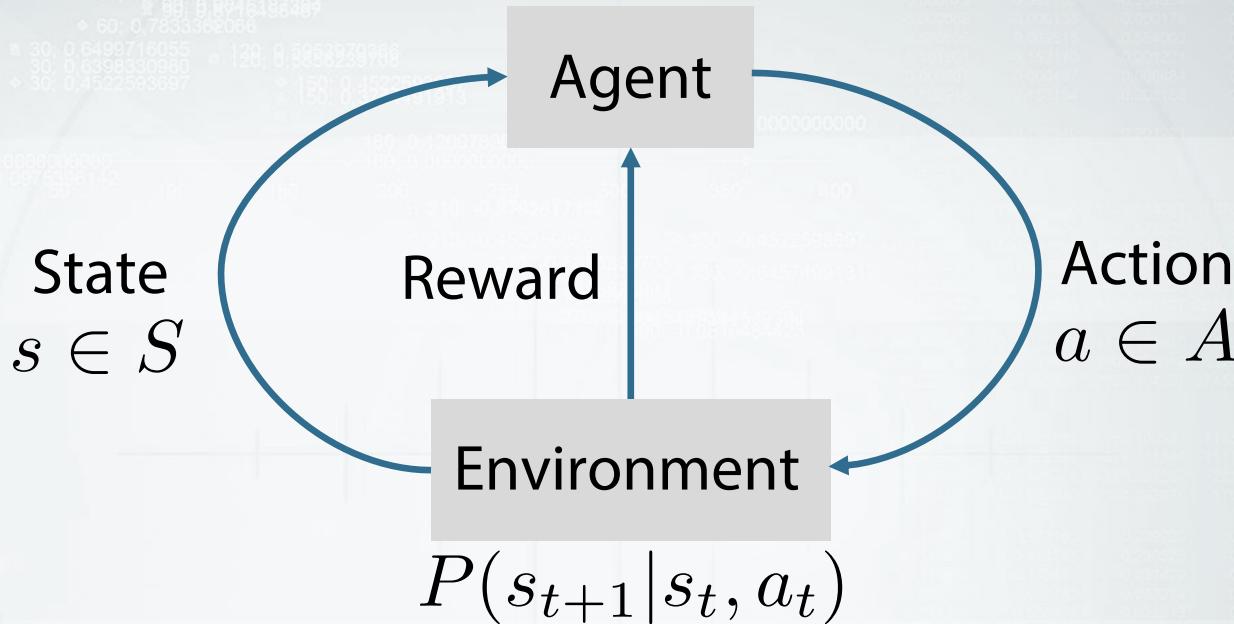
The MDP formalism



Markov Decision Process



The MDP formalism



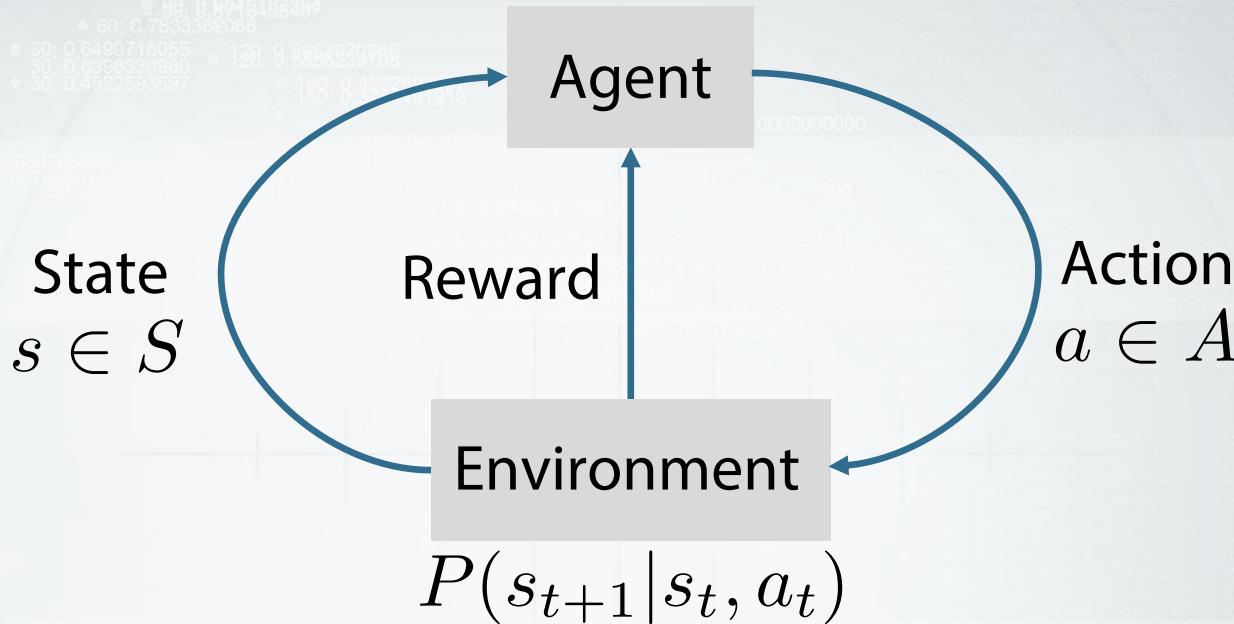
Markov Decision Process

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

The hard way:

$$E_{s_0 \sim p(s_0), a_0 \sim \pi(a|s_0), s_1 r_0 \sim P(s', r|s, a), \dots, s_n, r_n \sim \pi(a|s_{n-1})} [r_0 + r_1 + \dots + r_n]$$



How do we solve it?

Policy = probability of

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



Tabular crossentropy method

- Policy is a matrix

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

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

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



Tabular crossentropy method

- Policy is a matrix

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

- Sample N games with that policy



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

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



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



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



Smoothing

- If you were in some state only once, you only take this action now.
- Apply smoothing

$$\pi(a|s) = \frac{[\text{took } a \text{ at } s] + \lambda}{[\text{was at } s] + \lambda \cdot N_{actions}}$$

In M best games

Alternative idea: smooth updates:

$$\pi_{i+1}(a|s) = \alpha \cdot \pi_{opt} + (1 - \alpha)\pi_i(a|s)$$



Stochastic MDPs

- If there's randomness in environment, algorithm will prefer “lucky” sessions.

- Training on lucky sessions is no good

- **Solution:** sample action for each state and run several simulations with these state-action pairs. Average the results.



Grim reality

1.0000000000
0.5000000000
0.0000000000
-0.5000000000
-1.0000000000
-1.5000000000

▲ 60; 0.8178592142
▲ 60; 0.8945486309
◆ 60; 0.7833362066

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



Approximate crossentropy method

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

All state-action pairs from M best sessions

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

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

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

All state-action pairs from M best sessions

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

Maximize likelihood of actions in “best” games



Approximate crossentropy method

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

All state-action pairs from M best sessions

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

Maximize likelihood of actions in “best” games conveniently,

`nn.fit(elite_states,elite_actions)`



Approximate crossentropy method



Approximate crossentropy method

- Initialize NN weights $W_0 \leftarrow \text{random}$

- Loop:

- Sample N sessions
- elite = take M best sessions and concatenate

$$W_{i+1} = W_i + \alpha \nabla \left[\sum_{s_i, a_i \in \text{Elite}} \log \pi_{W_i}(a_i | s_i) \right]$$



Approximate

- Initialize NN weights $W_0 \leftarrow \text{random}$

• Loop:

- Sample N sessions
- elite = take M best sessions and concatenate

$$W_{i+1} = W_i + \alpha \nabla \left[\sum_{s_i, a_i \in \text{Elite}} \log \pi_{W_i}(a_i | s_i) \right]$$



Approximate (deep) version

- Initialize NN weights $W_0 \leftarrow \text{random}$

`model = MLPClassifier()`

- Loop:

- Sample N sessions
 - elite = take M best sessions and concatenate

$$W_{i+1} = W_i + \alpha \nabla \left[\sum_{s_i, a_i \in \text{Elite}} \log \pi_{W_i}(a_i | s_i) \right]$$

`model.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
 - $W_{i+1} = W_i + \alpha \nabla [\sum_{s_i, a_i \in Elite} \log \pi_{W_i}(a_i|s_i)]$

What changed?



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
 - $W_{i+1} = W_i + \alpha \nabla [\sum_{s_i, a_i \in Elite} \log \pi_{W_i}(a_i|s_i)]$



Continuous action spaces

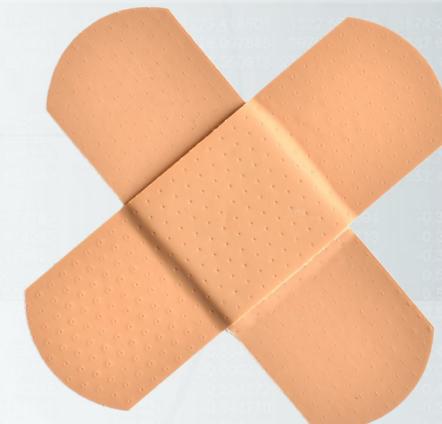
- Continuous state space model = `MLPRegressor()`
 - Model $\pi_W(a|s) = N(\mu(s), \sigma^2)$
 - Mu(s) is neural network output
 - Sigma is a parameter or yet another network output
 - Loop:
 - Sample N sessions
 - elite = take M best sessions and concatenate
 - $W_{i+1} = W_i + \alpha \nabla [\sum_{s_i,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

Nothing!

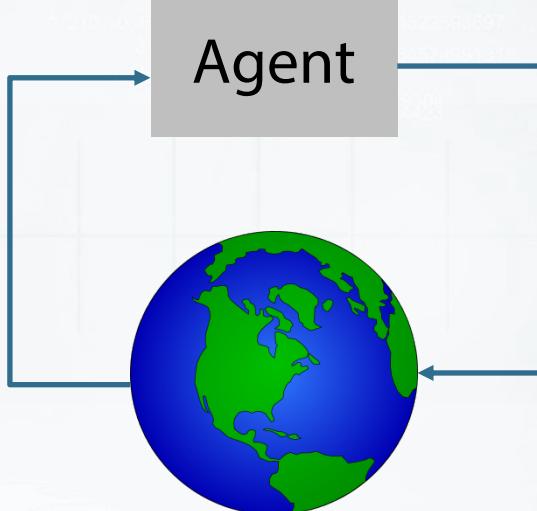


Tricks

- 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)



Black box optimization setup



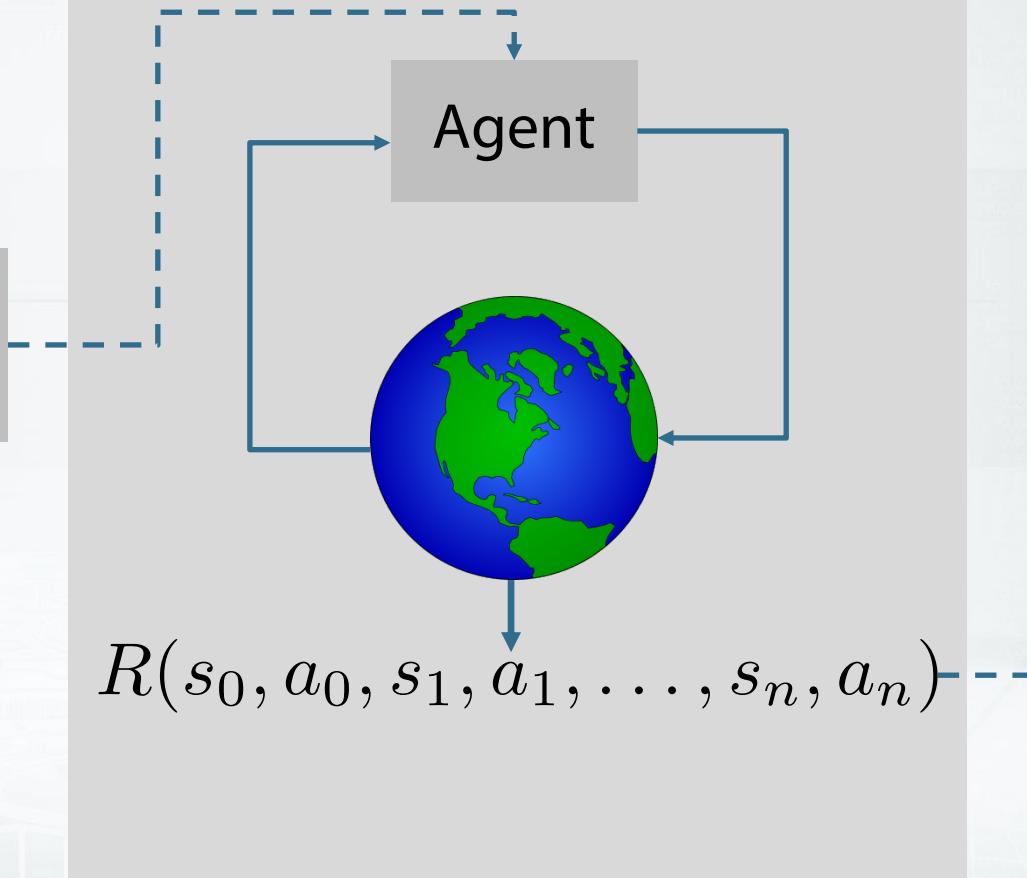
$$R(s_0, a_0, s_1, a_1, \dots, s_n, a_n)$$



Black box optimization setup



Policy
params



Black box optimization setup

Policy
params

black
box

R



Evolution Strategies

- Introduce distribution over weights

$$\theta \sim N(\theta|\mu, \sigma^2)$$

- Maximize expected reward

$$J = \underset{N(\theta|\mu, \sigma^2)}{E} R$$



Evolution Strategies

- Introduce distribution over weights

$$\theta \sim N(\theta|\mu, \sigma^2)$$

← other $P(\theta)$
will work as well

- Maximize expected reward

$$J = \underset{N(\theta|\mu, \sigma^2)}{E} R$$



Evolution Strategies

- Introduce distribution over weights

$$\theta \sim N(\theta|\mu, \sigma^2)$$

- Maximize expected reward



Evolution Strategies

- Introduce distribution over weights

$$\theta \sim N(\theta|\mu, \sigma^2)$$

- Maximize expected reward

$$J = \underset{\theta \sim N(\theta|\mu, \sigma^2)}{E} \underset{\tau(\theta)=s, a, s', a', \dots}{E} R(\tau)$$



Evolution Strategies



Δ 60; 0.8178592143
 \times 80; 0.8945486389
 \diamond 60; 0.7833362066
 \blacksquare 30; 0.6499716055
 \blacksquare 30; 0.6398330960
 \diamond 30; 0.4522583697

\star 120; 0.5853970366
 \star 120; 0.4522583697
 \star 120; 0.3744491973

-0.115781	0.173160	-0.229220	-0.263598	-0.335694	0.385298
-0.115933	0.174295	-0.230325	-0.264602	-0.336713	0.386261
-0.114933	0.177430	-0.231422	-0.265687	-0.337730	0.387223
0.115594	0.150787	0.653244	0.610526	0.563780	0.552311
0.115710	0.150479	0.653313	0.610331	0.563023	0.551638
0.115781	0.150174	0.652240	0.609333	0.562515	0.550963

-0.030173	-0.557762	-0.554354	-0.534707	-0.510990	-0.483385
-0.030203	-0.550120	-0.550178	-0.500217	-0.000257	-0.000285
-0.030203	-0.550116	-0.550262	-0.534715	0.510475	0.482784
-0.030173	-0.550177	-0.550231	-0.248770	-0.294026	0.233784
-0.030173	-0.550177	-0.550231	-0.248770	-0.294026	0.233784
-0.030097	-0.550487	-0.550484	0.000486	0.000448	0.000422

360; 0.0000000000	360; 0.0000000000	360; 0.0000000000	360; 0.0000000000	360; 0.0000000000	360; 0.0000000000
-------------------	-------------------	-------------------	-------------------	-------------------	-------------------

180; 0.1209782940

0; -0.0000000000

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375326342

0; -0.0375

Evolution Strategies

- Expected reward (using math. expectation)

$$J = \mathbb{E}_{\theta \sim N(\theta|\mu, \sigma^2)} \mathbb{E}_{\tau(\theta)=s, a, s', a', \dots} R(\tau)$$

Q: How can we estimate J in practice?
for large/infinite state space



Evolution Strategies

- Expected reward (using math. expectation)

$$J = \mathbb{E}_{\theta \sim N(\theta|\mu, \sigma^2)} \mathbb{E}_{\tau(\theta)=s, a, s', a', \dots} R(\tau)$$



Evolution Strategies

- Expected reward (using math. expectation)

$$J = \underset{\theta \sim N(\theta|\mu, \sigma^2)}{E} \underset{\tau(\theta)=s, a, s', a', \dots}{E} R(\tau)$$

- Approximate with sampling

$$\nabla J \approx \frac{1}{N} \sum_{i=0}^N \sum_{\tau_i=s, a, s', a', \dots} R(\tau_i)$$



Sample θ from $\theta \sim N(\theta|\mu, \sigma^2)$



Evolution Strategies

- Expected reward (using math. expectation)

- Approximate with sampling

$$\nabla J \approx \frac{1}{N} \sum_{i=0}^N \sum_{\tau_i=s,a,s',a',\dots} R(\tau_i)$$



Sample θ from $\theta \sim N(\theta|\mu, \sigma^2)$



Evolution Strategies

- Expected reward (using math. expectation)

$$J = \mathbb{E}_{N(\theta|\mu, \sigma^2)} \mathbb{E}_{\tau_i=s, a, s', a', \dots} R(s, a, s', a', \dots)$$

- Approximate with sampling

$$\nabla J \approx \frac{1}{N} \sum_{i=0}^N \sum_{\tau_i=s, a, s', a', \dots} R(\tau_i)$$

sample full episodes

Sample θ from $\theta \sim N(\theta|\mu, \sigma^2)$



Evolution Strategies

- Expected reward (using math. expectation)

Evolution Strategies

- Expected reward (using math. expectation)

$$J = \int N(\theta|\mu, \sigma^2) \cdot \int P(\tau|\theta)R(\tau)d\tau d\theta$$

- What we need

$$\nabla J = \frac{\partial J}{\partial}$$



Q: you compute ∇J w.r.t. what?



Evolution Strategies

- Expected reward (using math. expectation)

$$J = \int N(\theta|\mu, \sigma^2) \cdot \int P(\tau|\theta)R(\tau)d\tau d\theta$$

- What we need



Evolution Strategies

- Expected reward (using math. expectation)

$$J = \int N(\theta|\mu, \sigma^2) \cdot \int P(\tau|\theta)R(\tau)d\tau d\theta$$

- What we need

$$\nabla J = \frac{\partial J}{\partial \mu, \partial \sigma^2}$$



Gradient w.r.t. distribution parameters



Evolution Strategies

- Expected reward (using math. expectation)

$$J = \int N(\theta|\mu, \sigma^2) \cdot \int P(\tau|\theta)R(\tau)d\tau d\theta$$

- What we need



Evolution Strategies

- Expected reward (using math. expectation)

$$J = \int N(\theta|\mu, \sigma^2) \cdot \int P(\tau|\theta)R(\tau)d\tau d\theta$$

- What we need

$$\nabla J = \int \nabla[N(\theta|\mu, \sigma^2)] \cdot \int P(\tau|\theta)R(\tau)d\tau d\theta$$



Evolution Strategies

- Expected reward (using math. expectation)

$$J = \int N(\theta|\mu, \sigma^2) \cdot \int P(\tau|\theta)R(\tau)d\tau d\theta$$

- What we need

$$\nabla J = \int \nabla[N(\theta|\mu, \sigma^2)] \cdot \int P(\tau|\theta)R(\tau)d\tau d\theta$$



Evolution Strategies

- Expected reward (using math. expectation)

$$J = \int N(\theta|\mu, \sigma^2) \cdot \int P(\tau|\theta)R(\tau)d\tau d\theta$$

- What we need

$$\nabla J = \int \nabla[N(\theta|\mu, \sigma^2)] \cdot \int P(\tau|\theta)R(\tau)d\tau d\theta$$



Can't estimate by sampling!

$\nabla N(\theta|\mu, \sigma)$ is not a distribution



Logderivative trick

Simple math

$$\nabla \log f(x) =$$



Logderivative trick

Simple math

$$\nabla \log f(x) = \frac{1}{f(x)} \cdot \nabla f(x)$$

$$f(x) \cdot \nabla \log f(z) = \nabla f(z)$$



Logderivative trick



Logderivative trick

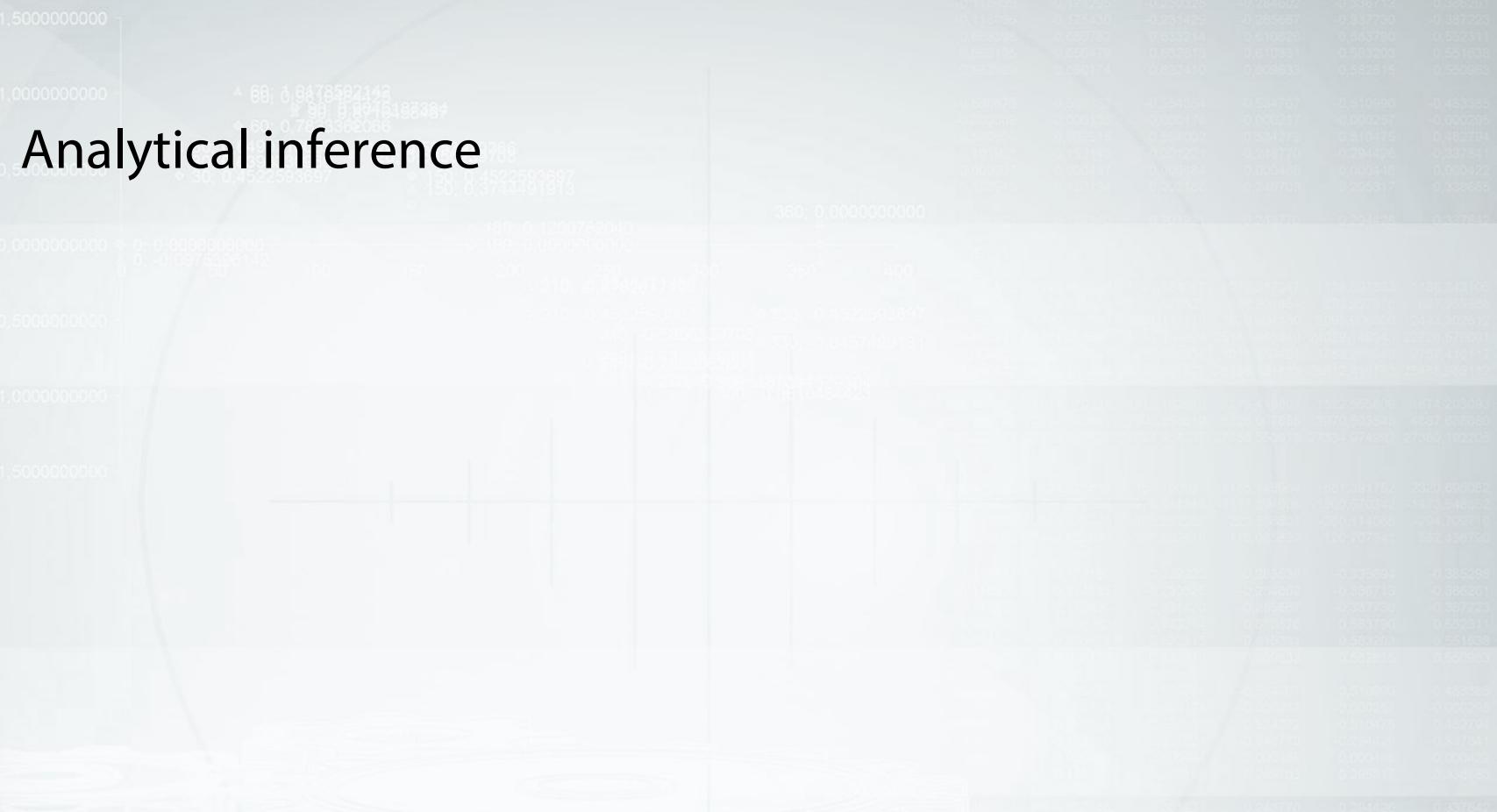
Analytical inference

$$\nabla J = \int \nabla[N(\theta|\mu, \sigma^2)] \cdot \int P(\tau|\theta)R(\tau)d\tau d\theta$$

$$\nabla N(\theta|\mu, \sigma^2) = N(\theta|\mu, \sigma^2) \cdot \nabla \log N(\theta|\mu, \sigma^2)$$



Analytical inference



Evolution Strategies

Analytical inference

$$\nabla J = \int [N(\theta|\mu, \sigma^2) \cdot \nabla \log N(\theta|\mu, \sigma^2)] \cdot \int P(\tau|\theta)R(\tau)d\tau d\theta$$

Q: How can we estimate ∇J in now?



Evolution Strategies

Analytical inference



Evolution Strategies

Analytical inference

$$\nabla J = \underset{\theta \sim N(\theta|\mu, \sigma^2)}{E} \nabla \log N(\theta|\mu, \sigma^2) \cdot \underset{\tau(\theta)=s,a,s',\dots}{E} R(\tau)$$

Sampled estimate

$$\nabla J \approx \frac{1}{N} \sum_{i=0}^N \nabla \log N(\theta|\mu, \sigma^2) \cdot \sum_{s,a \in z_i} R(s, a \dots)$$



Sample θ from $\theta \sim N(\theta|\mu, \sigma^2)$



Evolution Strategies



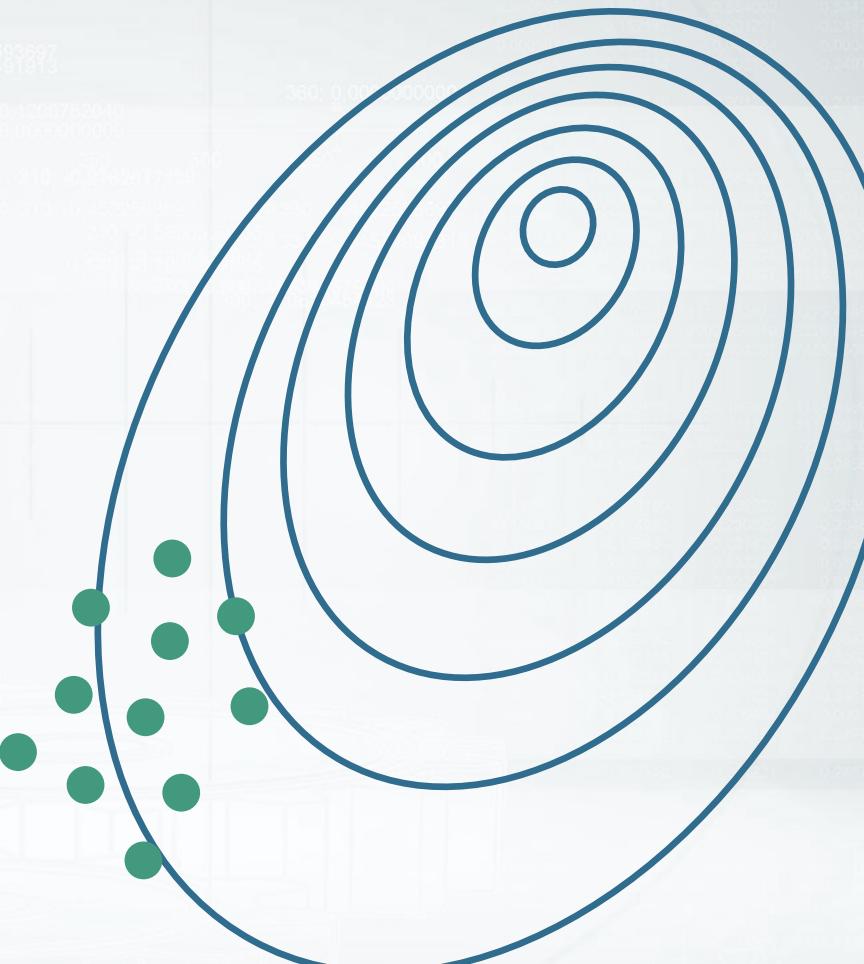
Evolution Strategies

▲ 60; 0.8178592143
▲ 80; 0.8945486309
◆ 60; 0.7833362066
■ 30; 0.6499716055
■ 30; 0.6398330960
◆ 30; 0.4522583697
◆ 120; 0.5853970366
◆ 120; 0.4522583697
◆ 150; 0.4522583697
◆ 150; 0.3744391973



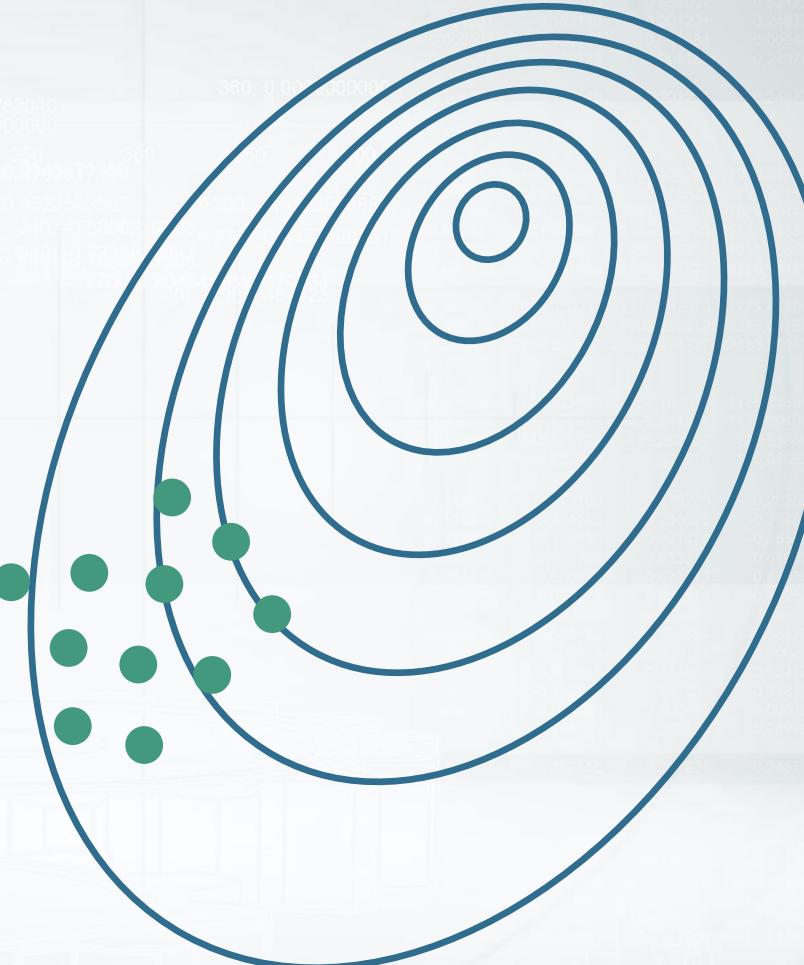
Evolution Strategies

▲ 60; 0.8178592143
▲ 80; 0.8945486309
◆ 60; 0.7833362066
■ 30; 0.6499716055
■ 30; 0.6398330960
◆ 30; 0.4522583697
◆ 120; 0.5853973766
◆ 120; 0.4522583697
◆ 150; 0.4522583697
◆ 150; 0.3744391973



Evolution Strategies

▲ 60; 0.8178592143
▲ 80; 0.8945486309
◆ 60; 0.7833362066
■ 30; 0.6499716055
■ 30; 0.6398330960
◆ 30; 0.4522583697
◆ 120; 0.5853973766
◆ 120; 0.4522583697
◆ 150; 0.4522583697
◆ 150; 0.3743491973



Evolution Strategies

▲ 60; 0.8178592143
▲ 80; 0.8945486309
◆ 60; 0.7833362066
■ 30; 0.6499716055
■ 30; 0.6398330960
◆ 30; 0.4522583697
◆ 120; 0.5853973766
◆ 120; 0.4522583697
◆ 150; 0.4522583697
◆ 150; 0.3743491973



Evolution Strategies



Evolution Strategies

▲ 60; 0.8178592143
▲ 80; 0.8945486309
◆ 60; 0.7833362066
■ 30; 0.6499716055
■ 30; 0.6398330960
◆ 30; 0.4522583697
■ 120; 0.5853973666
■ 120; 0.4522583697
◆ 120; 0.3743491973



Evolution Strategies

Algorithm

1. Initialize μ_0, σ_0^2

2. Forever:

$$\nabla J \approx \frac{1}{N} \sum_{i=0}^N \nabla \log N(\theta | \mu, \sigma^2) \cdot \sum_{s, a, s', \dots \in \tau_i} R(s, a, s', \dots)$$



Evolution Strategies

Algorithm

1. Initialize μ_0, σ_0^2

2. Forever:

$$\nabla J \approx \frac{1}{N} \sum_{i=0}^N \nabla \log N(\theta | \mu, \sigma^2) \cdot \sum_{s, a, s', \dots \in \tau_i} R(s, a, s', \dots)$$

$$\mu = \mu + \alpha \nabla_\mu J$$

$$\sigma^2 = \sigma^2 + \alpha \cdot \nabla_{\sigma^2} J$$



Evolution Strategies

Algorithm

1. Initialize μ_0, σ_0^2

2. Forever:



Evolution Strategies

Algorithm

1. Initialize μ_0, σ_0^2

2. Forever: $\theta = \mu + \sigma \cdot \xi : \xi \sim N(0, 1)$

sample $\theta_i, \tau_i, R_i : R_i = R(\tau_i) = R(\tau(\theta_i))$

$$\mu = \mu + \alpha \cdot \frac{1}{N \cdot \sigma} \cdot \sum_i \xi_i \cdot R_i$$

$$\sigma^2 = \dots$$



Reward baselines

TL;DR normalize reward

$$\operatorname{argmax}_{N(\theta|\mu,\sigma^2)} E R = \operatorname{argmax}_{N(\theta|\mu,\sigma^2)} \frac{R - \text{mean}}{\text{std}}$$

$$A = \frac{R - ER}{Var R}$$



Evolution strategies

Features

- A general black box optimization
- **Easy to implement & scale**



Evolution strategies

Features

- A general black box optimization
- Easy to implement & scale

$$\nabla J \approx \frac{1}{N} \sum_{i=0}^N \nabla \log N(\theta | \mu, \sigma^2) \cdot \sum_{\tau_i=s, a, s', \dots} R(\tau_i)$$

Q: You have 1000 CPUs.

Optimize this formula!



Evolution strategies

Features

- A general black box optimization
- **Easy to implement & scale**

$$\nabla J \approx \frac{1}{N} \sum_{i=0}^N \nabla \log N(\theta | \mu, \sigma^2) \cdot \sum_{\tau_i=s, a, s', \dots} R(\tau_i)$$



Evolution strategies

Features

- A general black box optimization
- **Easy to implement & scale**

$$\nabla J \approx \frac{1}{N} \sum_{i=0}^N \nabla \log N(\theta | \mu, \sigma^2) \cdot \sum_{\tau_i=s,a,s',\dots} R(\tau_i)$$

You can compute every session in parallel



Evolution strategies

Features

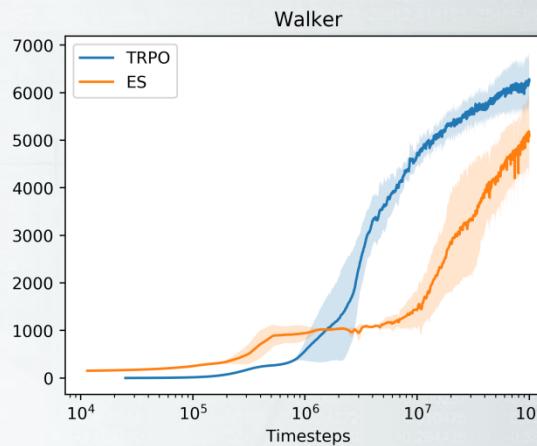
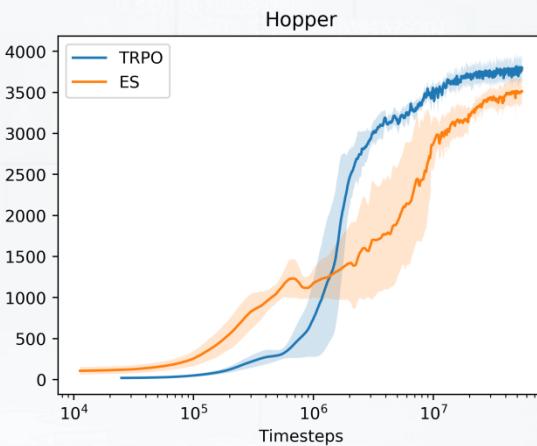
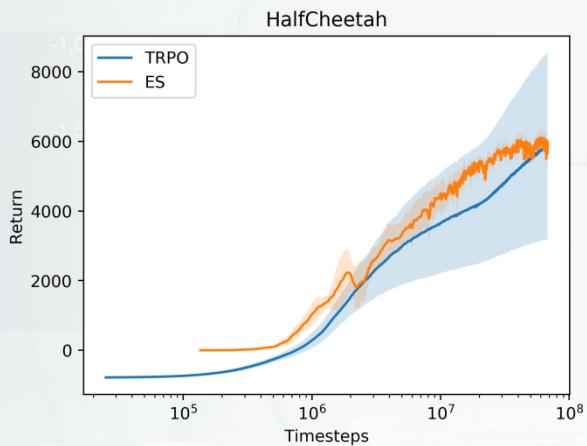
- A general black box optimization



Evolution strategies

Features

- A general black box optimization
- Some results on gym



<https://blog.openai.com/evolution-strategies/>



Common drawback

Both CEM, ES and all similar methods

- Train from full sessions only
- Require a lot of samples



Common drawback

Both CEM, ES and all similar methods

- Train from full sessions only
- Require a lot of samples



Edmund McMillen, Game: Super Meat Boy, https://en.wikipedia.org/wiki/Super_Meat_Boy



Common drawback

Both CEM, ES and all similar methods

- Train from full sessions only
- Require a lot of samples



Edmund McMillen, Game: Super Meat Boy, https://en.wikipedia.org/wiki/Super_Meat_Boy

