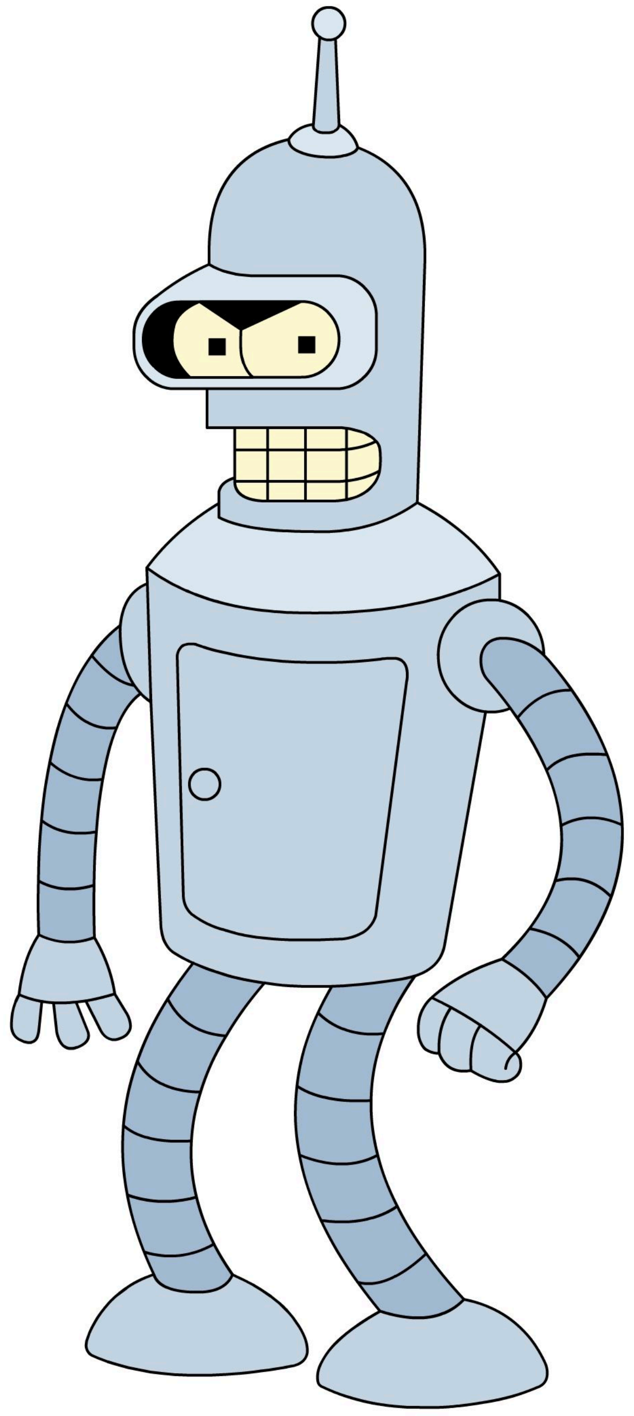
Reinforcement Learning

HSE, autumn - winter 2022

Lecture 1: Intro, CEM

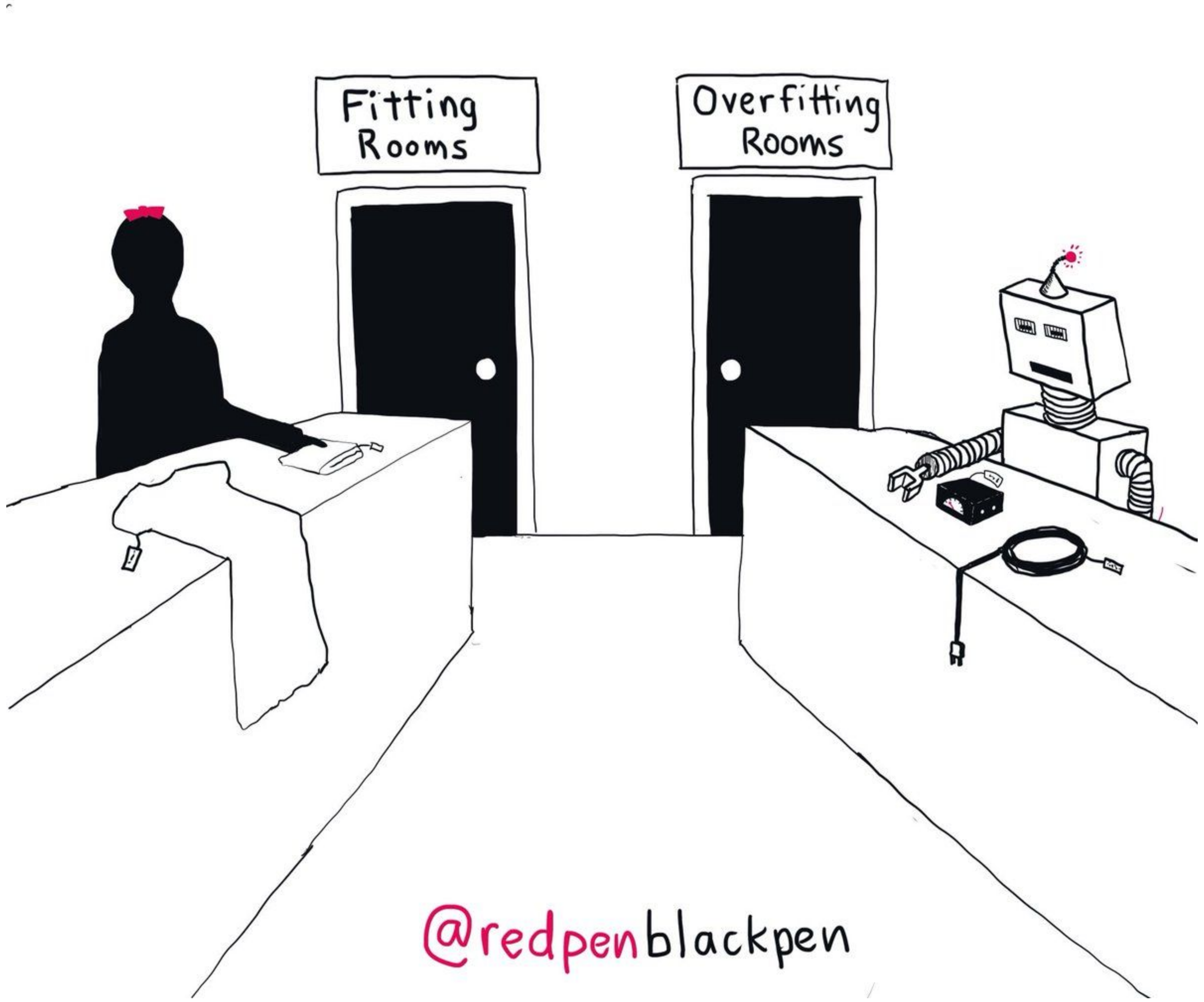


Sergei Laktionov
slaktionov@hse.ru
LinkedIn

Course Staff

Sergei Laktionov
Lectures and practical sections
https://t.me/lsd4math

Arsenty Kambalin
Home assignments and grading
https://t.me/arsentique_s



Source

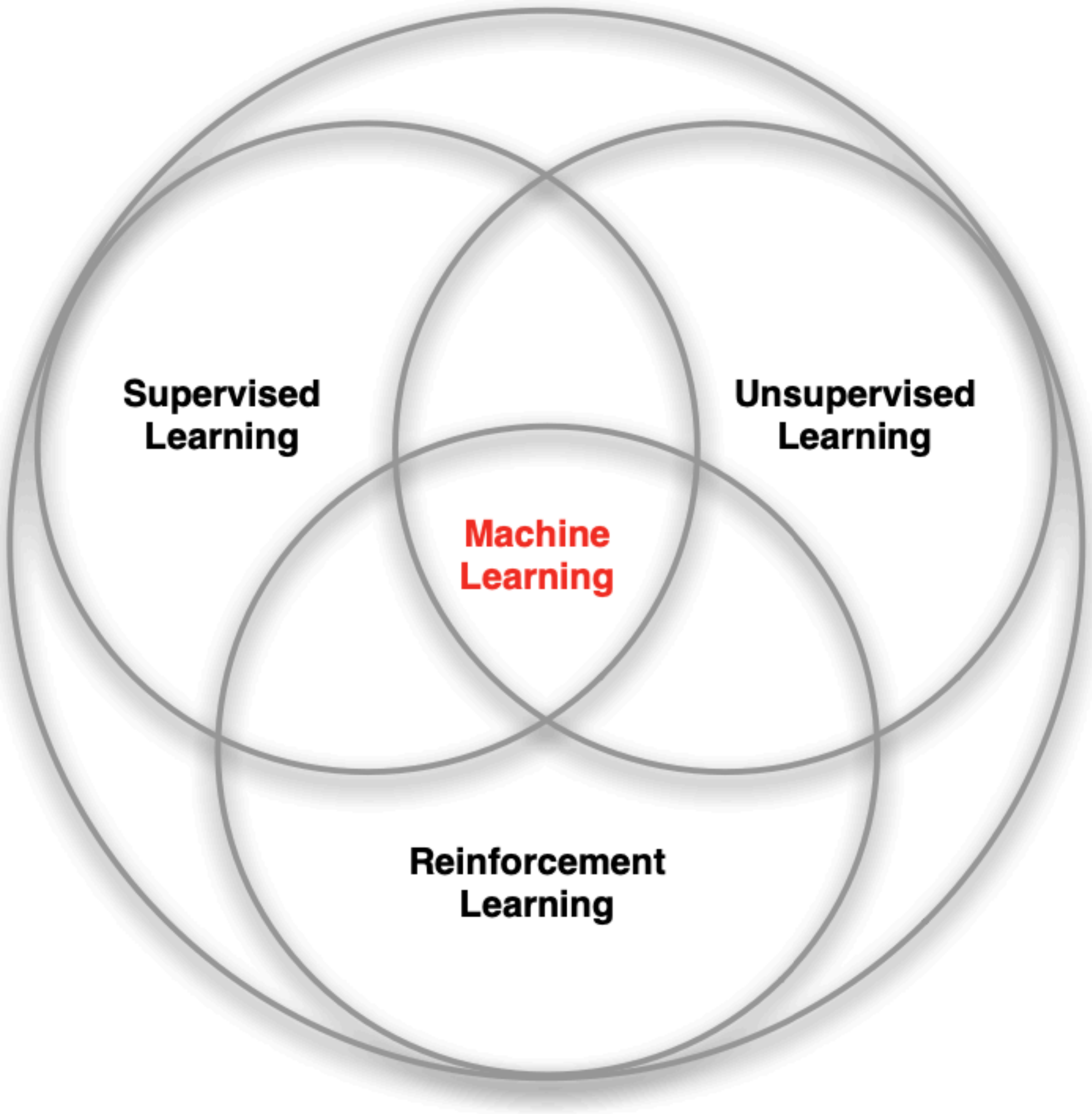
Course Plan

1. 01.11.2022: RL problem statement. Cross-entropy method. (HW1)
2. 08.11.2022: Bellman equations and Dynamic programming. (HW2)
3. 15.11.2022: Model free algorithms. Tabular methods.
4. 22.11.2022: Intro to deep RL. Value-based methods: DQN and modifications (HW3)
5. 29.11.2022: Policy-based methods
6. 06.12.2022: Actor-critic algorithms (HW4)
7. 13.12.2022: Bandits
8. 20.12.2022: Research club :)

Useful Reading

1. http://incompleteideas.net/book/the-book-2nd.html
2. http://rail.eecs.berkeley.edu/deeprlcourse/
3. https://github.com/yandexdataschool/Practical_RL
4. https://www.deepmind.com/learning-resources/reinforcement-learning-lecture-series-2021
5. https://web.stanford.edu/class/cme241/
6. https://github.com/huggingface/deep-rl-class

Recap



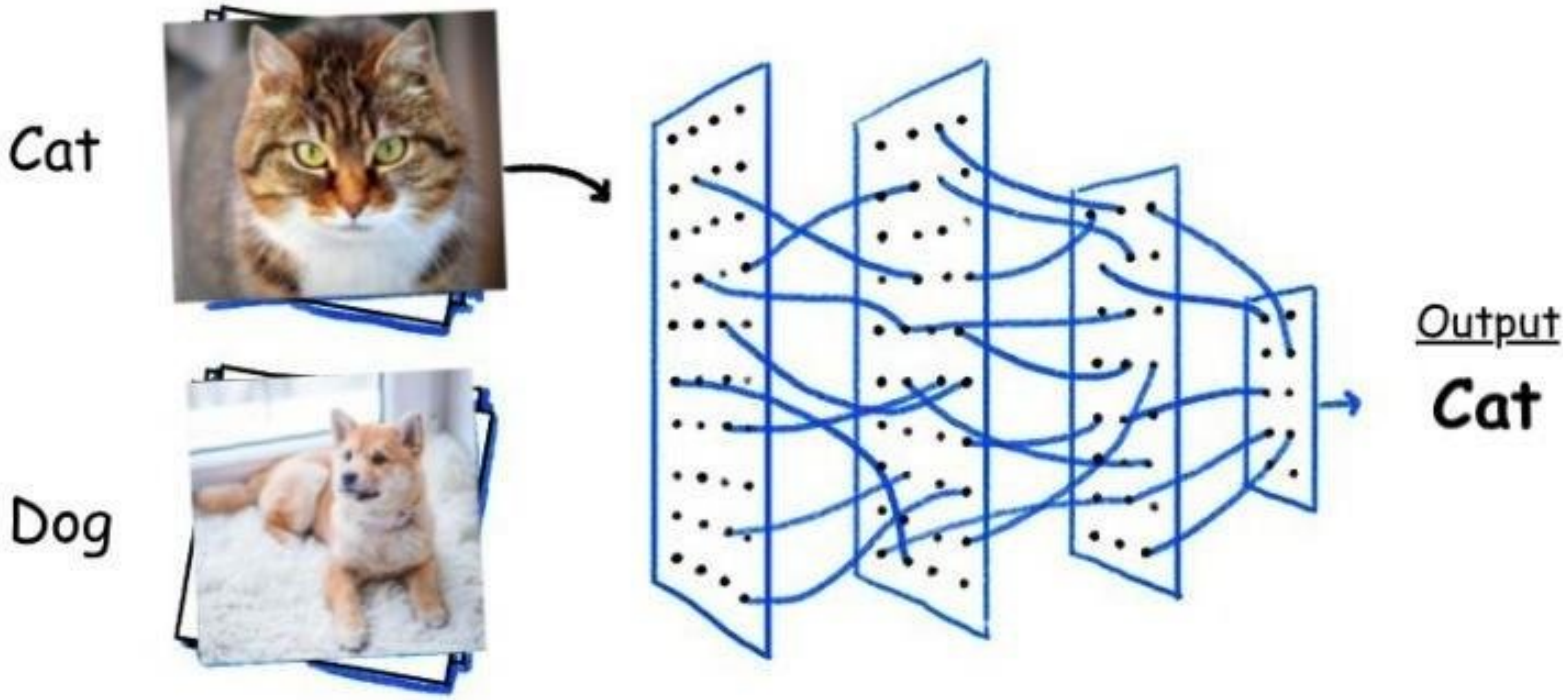
Source

Supervised Learning

Train sample: $(\mathbf{x}_i, y_i) \sim i.i.d.$

Approximation: $f(\mathbf{x}_i) = \hat{y}_i \approx y_i$

Optimization: $\frac{1}{n}\sum_{i=1}^{n} L(f(\mathbf{x}_i), y_i) \to \min_{f \in \mathcal{F}}$



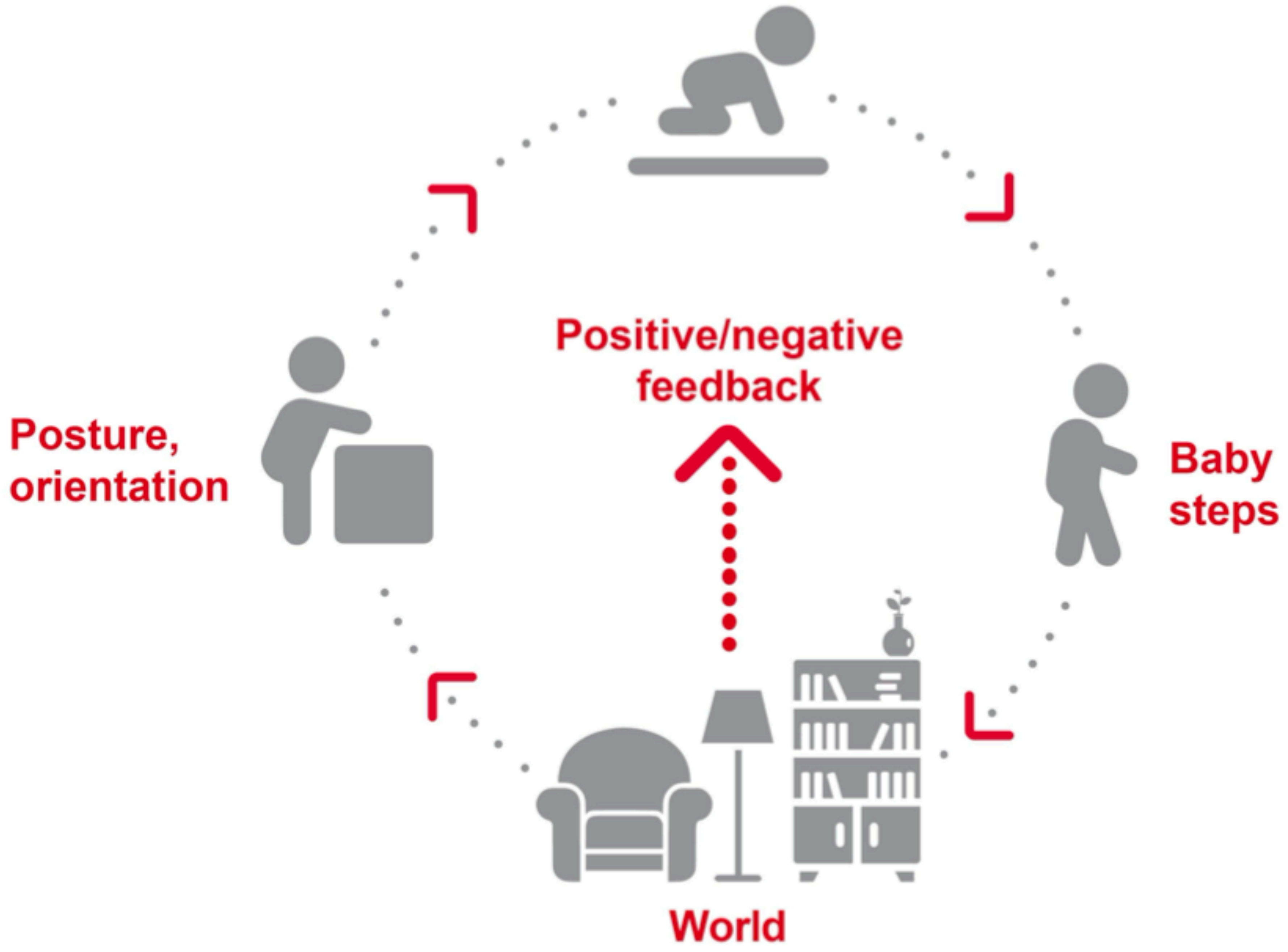
Source

Background for Lecture 1

1. Sutton & Barto 2018, Chapter 1
2. Rao & Jelvis 2022, Chapter 1

Decision Making Process

Baby learning

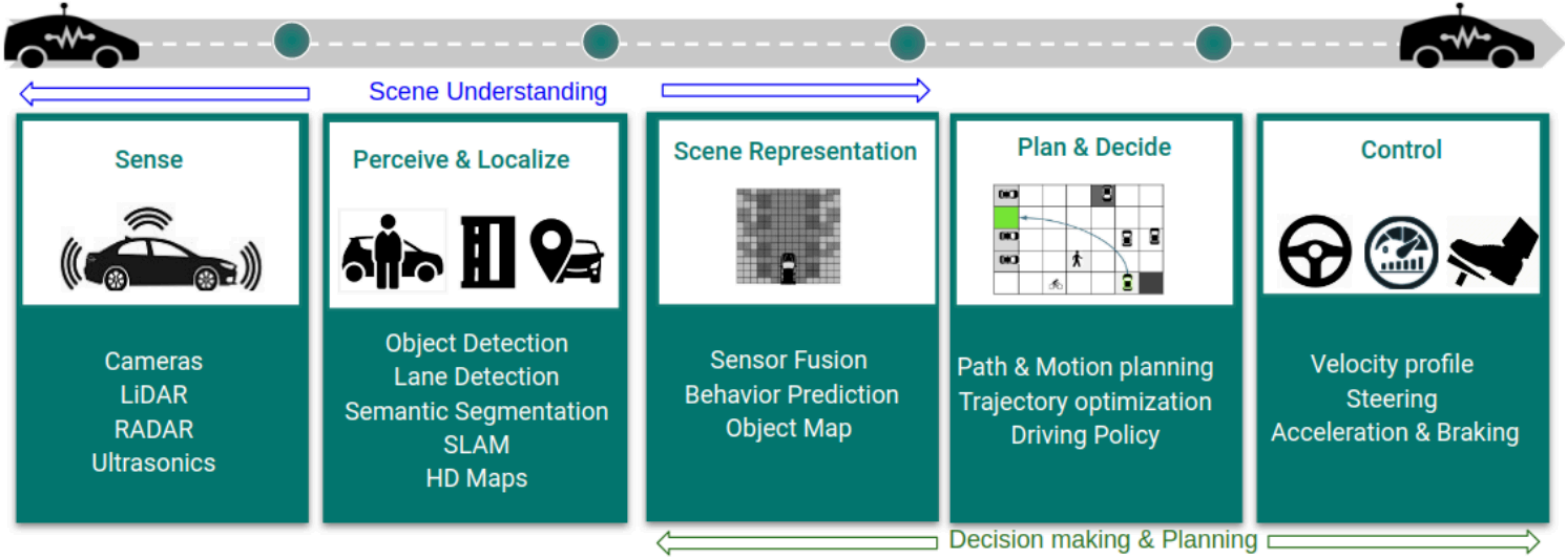


Source

Self-driving car

Source

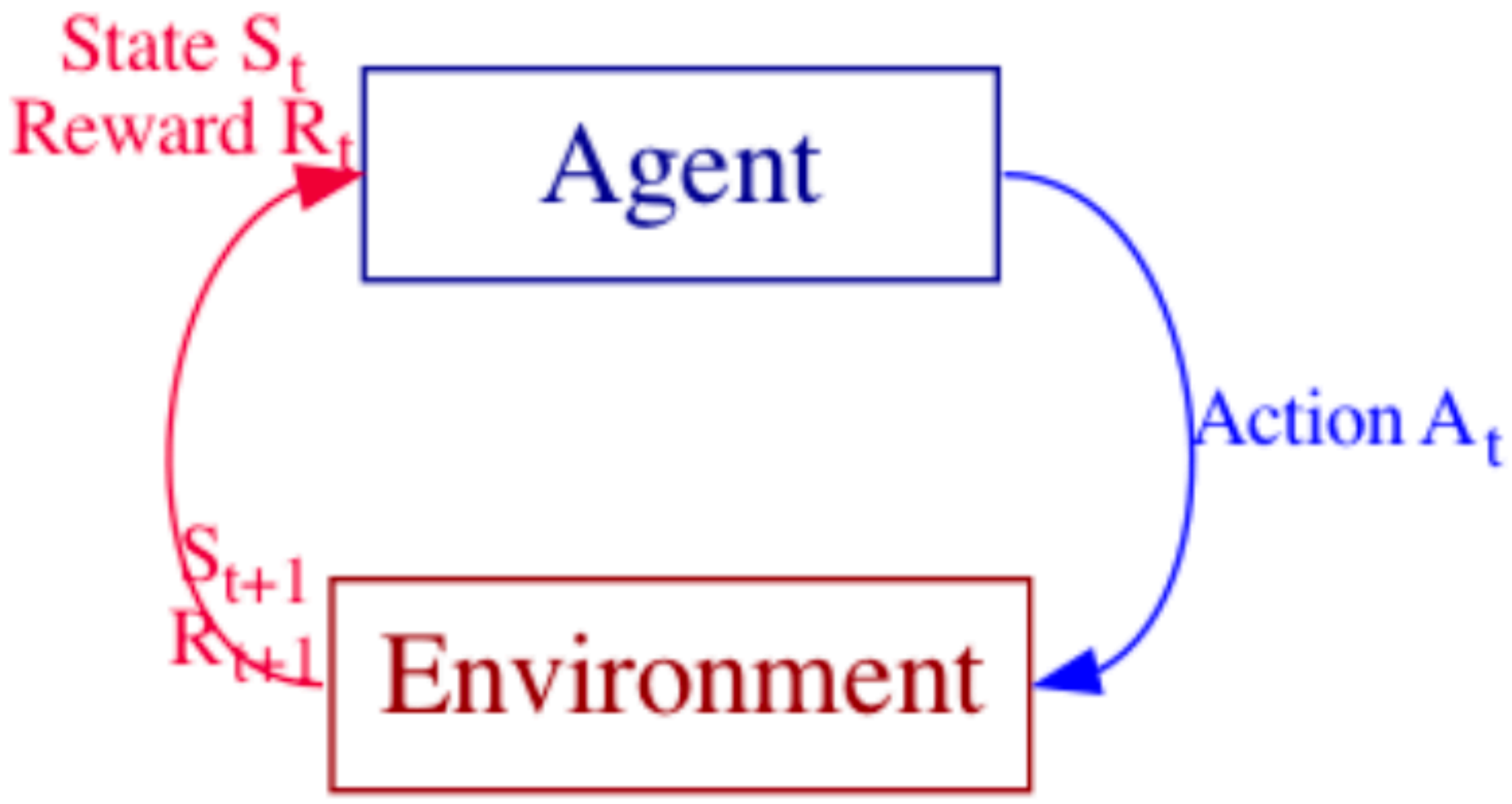
Self-driving Cars



Source

Reinforcement Learning Features

1. Trial-and-error search
2. Delayed rewards
3. Interaction with the environment



Source

Reward hypothesis (Richard Sutton):

"That all of what we mean by goals and purposes can be well thought of as maximization of the expected value of the cumulative sum of a received scalar signal (reward)."

Reinforcement Learning Features

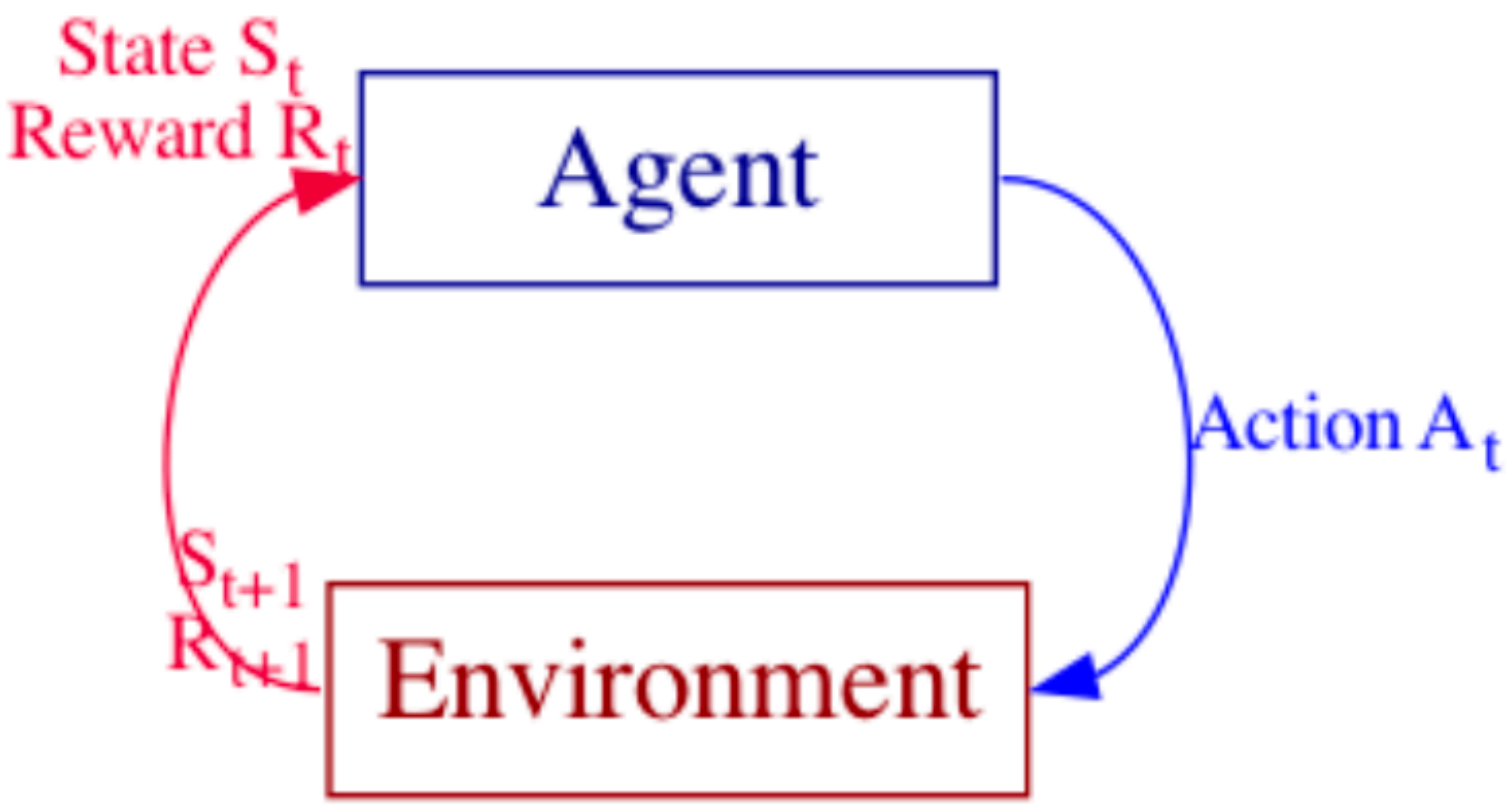
The agent chooses the action by using a **policy** π.

Deterministic policy

$$a := \pi(s)$$

Stochastic policy

$$a \sim \pi(a \mid s)$$



Source

Imitation Learning and Behavioural Cloning

Source

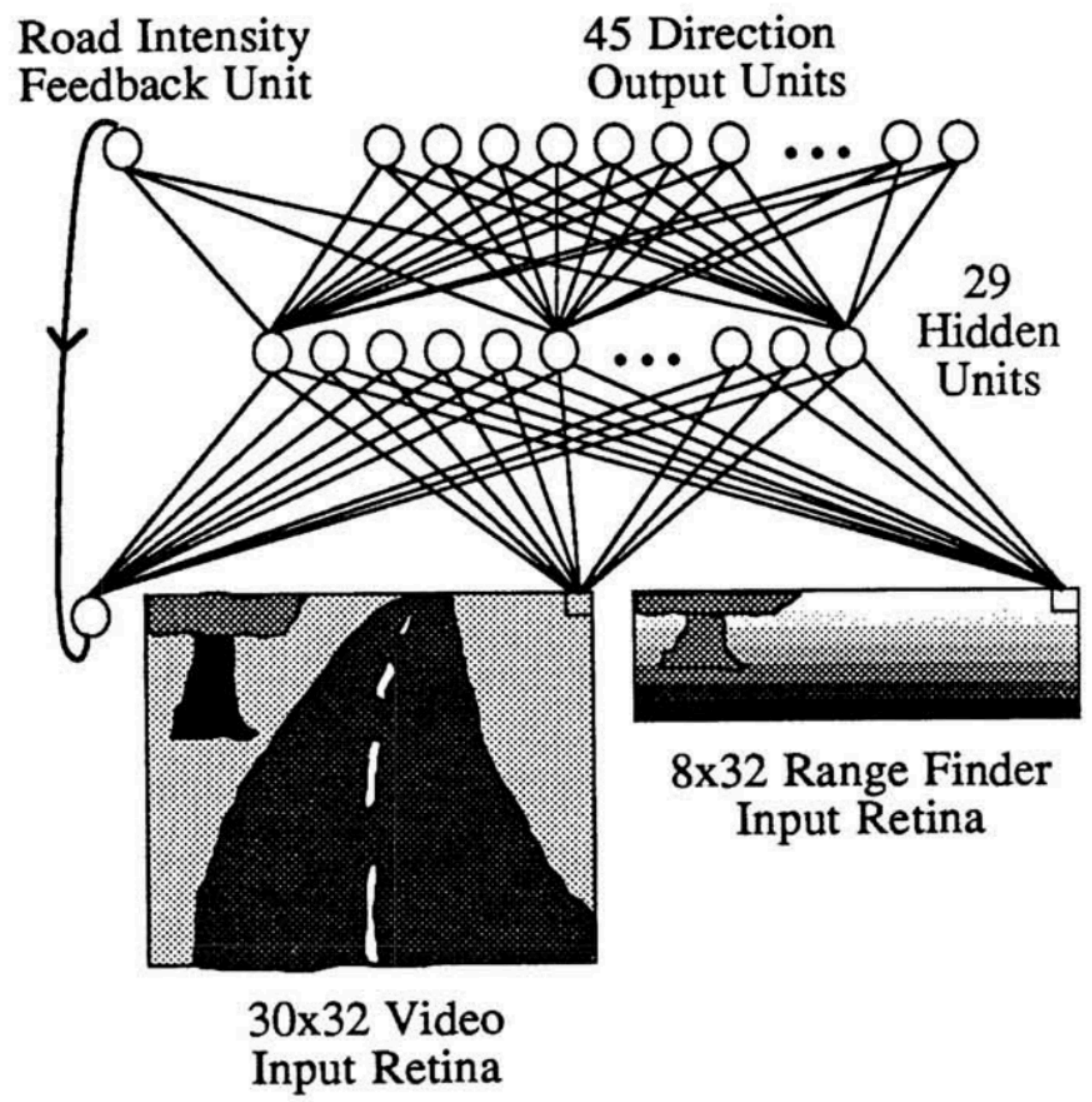
Imitation Learning is useful when it is easier for an expert to demonstrate the desired behaviour rather than to specify a reward function which would generate the same behaviour or to directly learn the policy.

The simplest form of imitation learning is Behavioural Cloning, which focuses on learning the expert’s policy using supervised learning.

ALVINN:
AN AUTONOMOUS LAND VEHICLE IN A NEURAL NETWORK

Dean A. Pomerleau
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Source



Source

Behavioral Cloning for Self-driving Cars

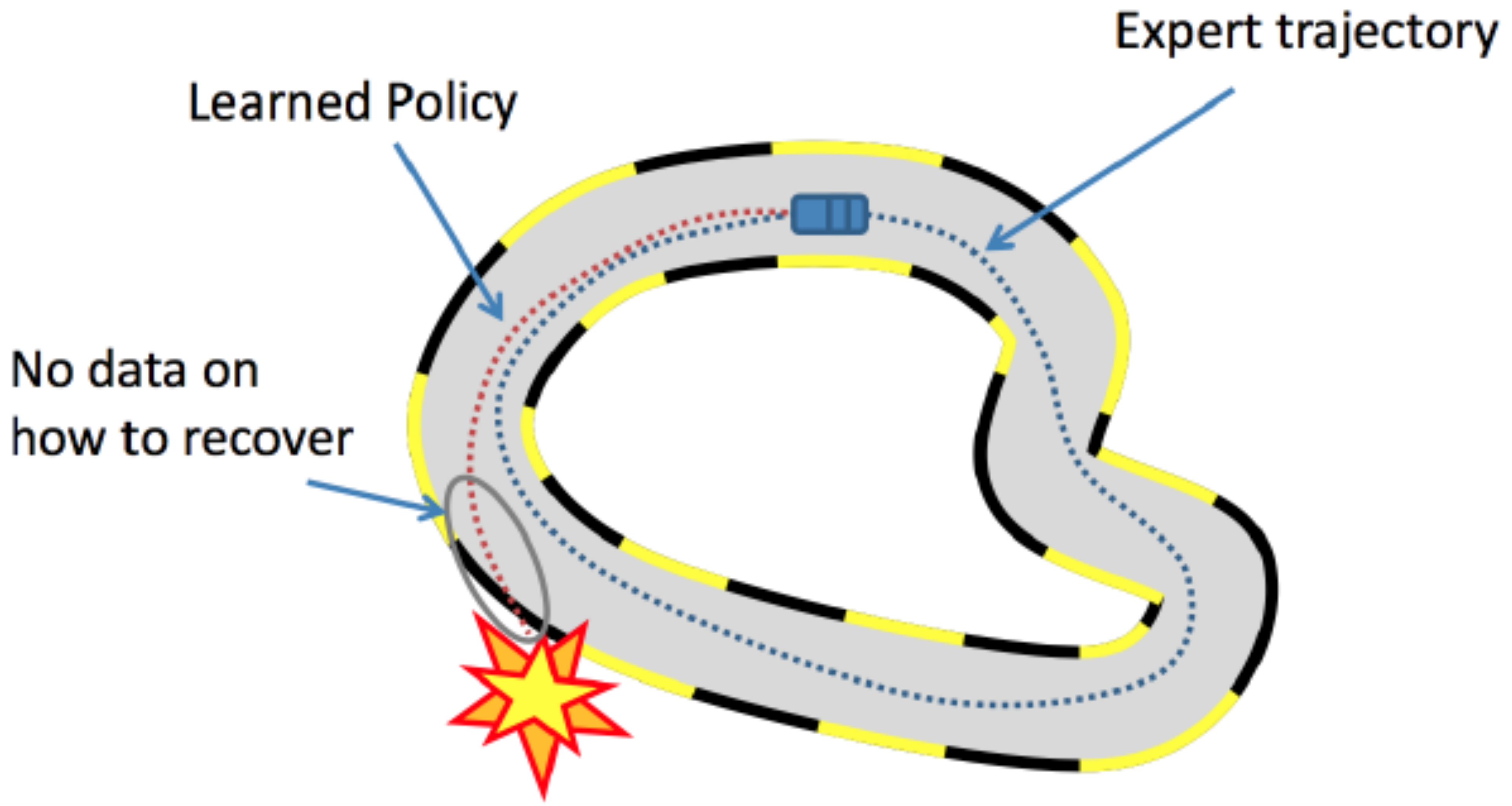
1. Collect demonstrations (trajectories τ^*) from expert

2. Treat the demonstrations as i.i.d. state-action pairs: (s_i, a_i)

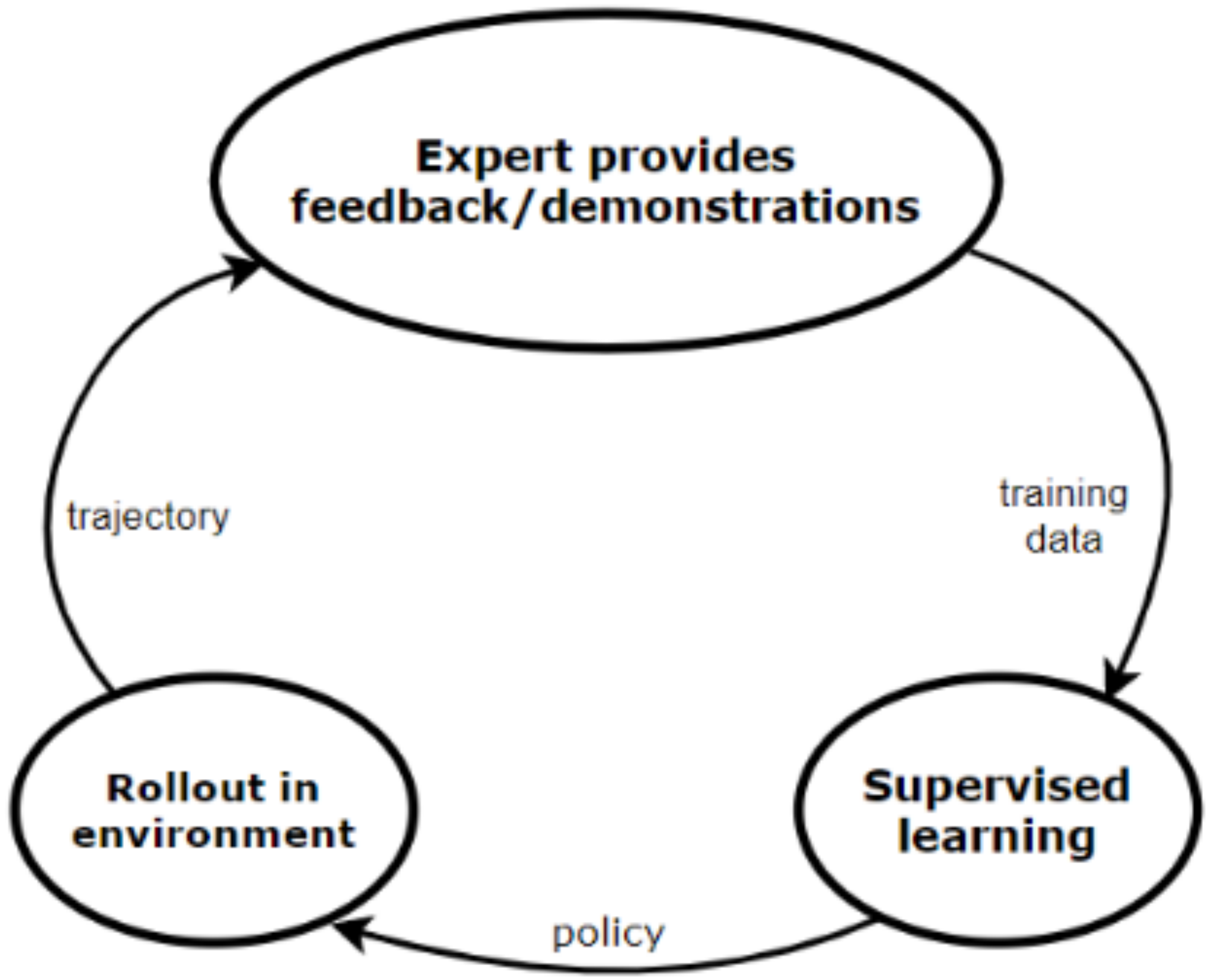
3. Learn the policy in a supervised mode: $\frac{1}{n}\sum_{i=1}^{n} L(\pi(s_i), a_i) \to \min_{\pi \in \mathscr{P}}$

Behavioral Cloning for Self-driving Cars

1. Collect demonstrations (trajectories τ^*) from expert
2. Treat the demonstrations as i.i.d. state-action pairs: (s_i, a_i)
3. Learn the policy in a supervised mode: $\frac{1}{n}\sum_{i=1}^{n} L(\pi(s_i), a_i) \to \min_{\pi \in \mathscr{P}}$



Direct Policy Learning



Source

Initial predictor: π_0

For $m = 1$:

- Collect trajectories τ by rolling out π_{m-1}
- Estimate state distribution P_m using $s \in \tau$
- Collect interactive feedback $\{\pi^*(s) \mid s \in \tau\}$
- Data Aggregation (e.g. Dagger)
  - Train π_m on $P_1 \cup \ldots \cup P_m$
- Policy Aggregation (e.g. SEARN & SMILe)
  - Train π'_m on P_m
  - $\pi_m = \beta\pi'_m + (1-\beta)\pi_{m-1}$

Source

Demonstration

More Examples

Robotics

- Optimal control

Chess, Go

- AlphaGo
- AlphaZero

Quantitative Finance:

- Market making
- Optimal execution
- Portfolio management

- Healthcare
- Industry automation
- Engineering

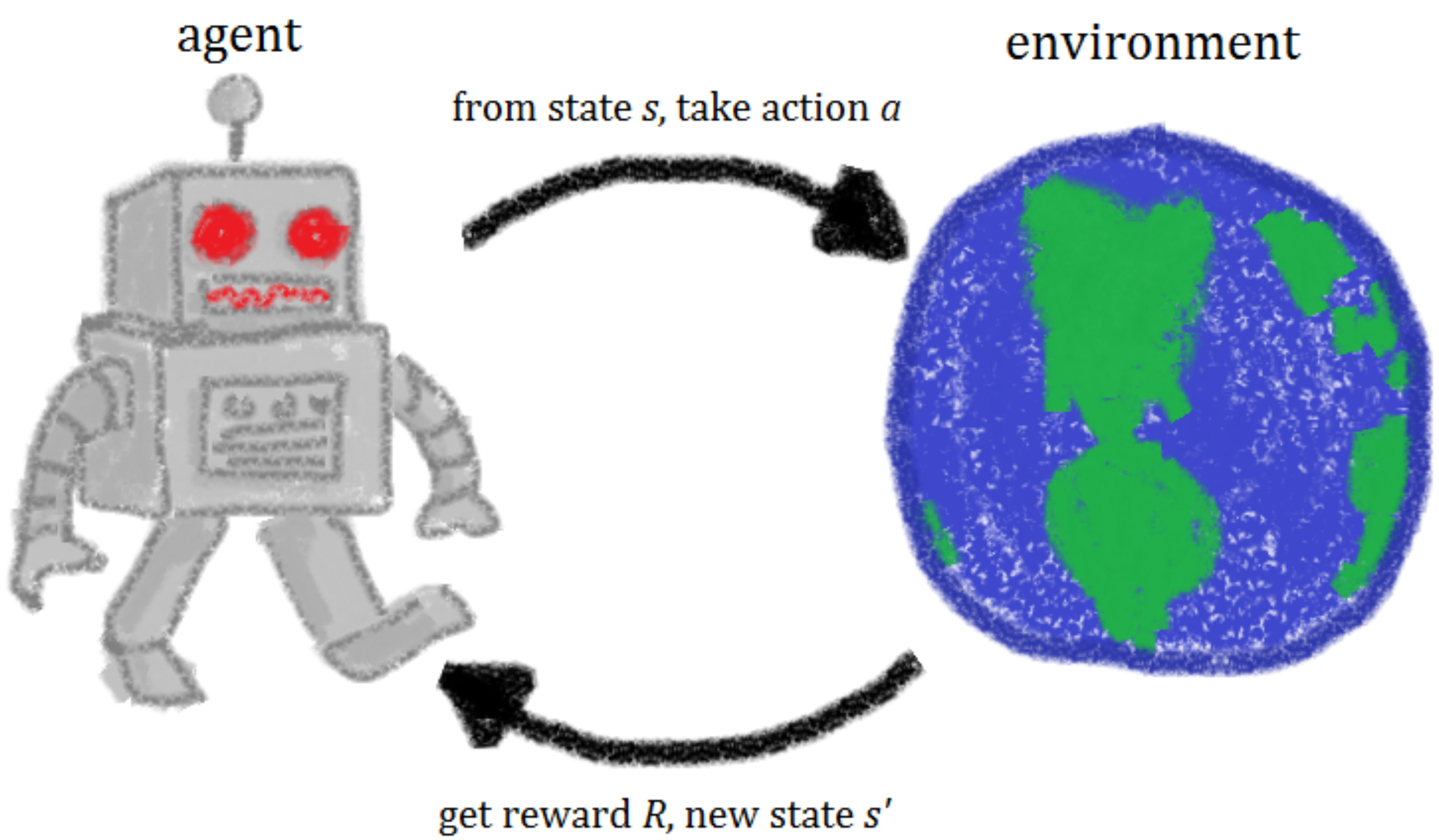
Reinforcement Learning Framework

At each step t the agent:

1. Receives state (observation) s_t (o_t) and reward r_t
2. Executes action a_t

The environment:

1. Receives action a_t
2. Emits state (observation) s_{t+1} (o_{t+1}) and reward r_{t+1}



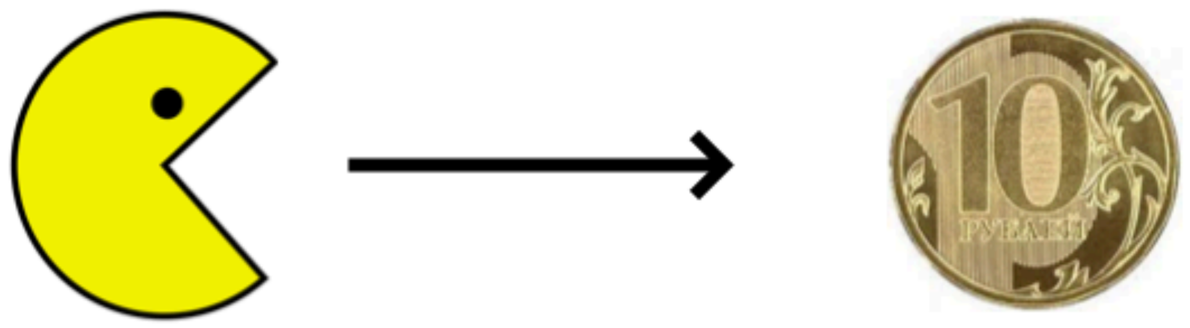
Source

$$\mathbb{E}_{p(\tau|\pi)}\left(\sum_{t=0}^{T} r_t\right) \to \max_\pi$$

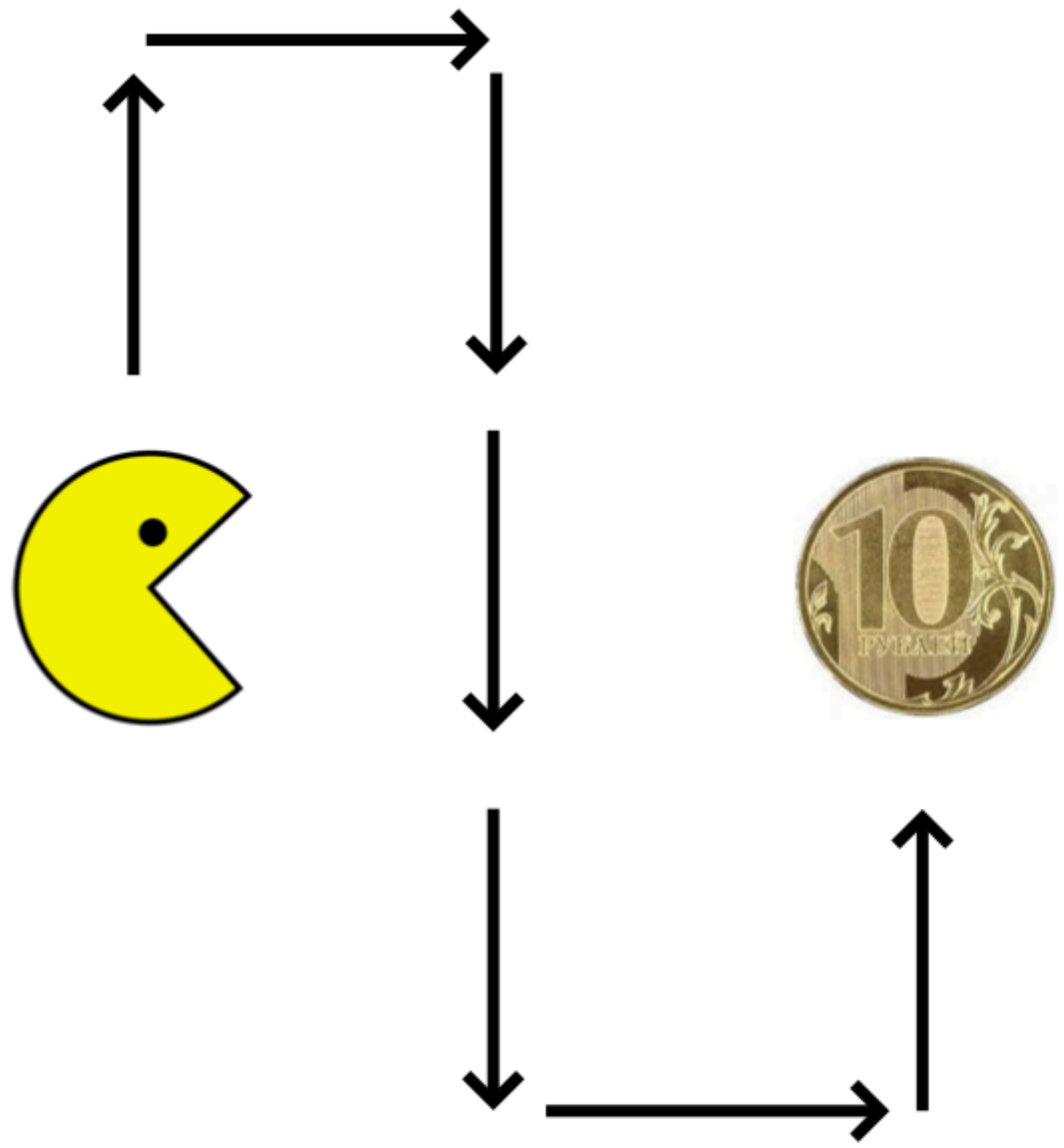
Reward Design

… is a tricky task

Discount Factor

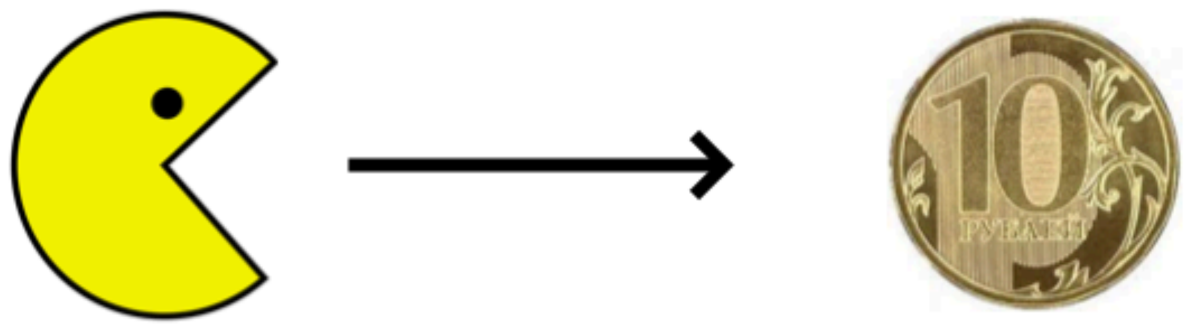


Source

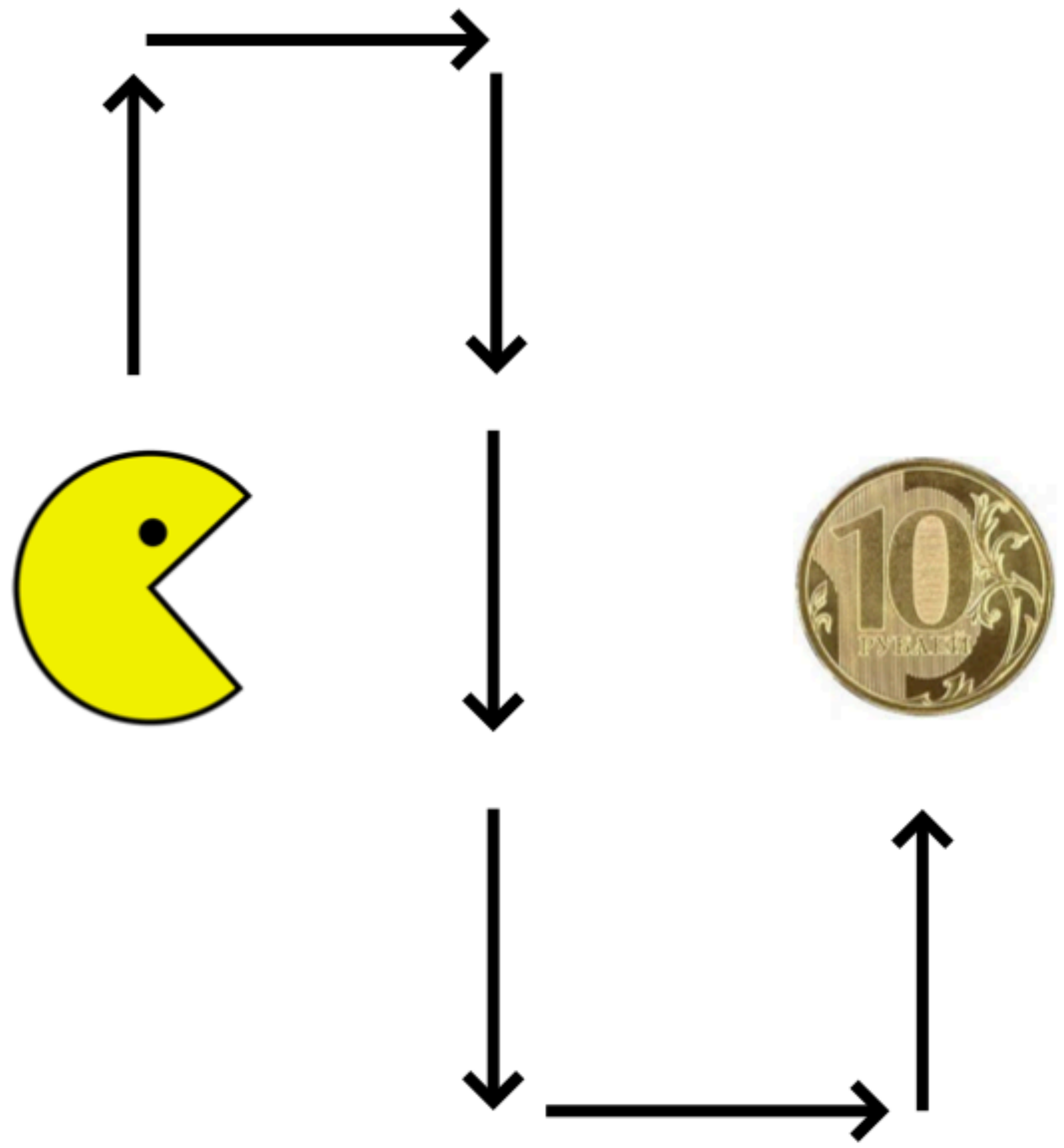


Source

Discount Factor



Source



Source

For $\gamma \in [0,1]$ consider $\mathbb{E}_\pi(\sum_{t=0}^{T} \gamma^t r_t) \to \max_\pi$

Reinforcement Learning Formalism

1. $a \in \mathscr{A}$ - action space

2. $s \in \mathscr{S}$ - state space

2*. $o \in \mathscr{O}$ - observation space for partial observable environment

3. $r(s, a) : \mathscr{A} \times \mathscr{S} \to \mathscr{P}(\mathscr{R})$ - reward function

4. $p(r, s' \,|\, s, a)$ - environment dynamics

5. $p(s_0)$ - initial state distribution

6. $\pi(a \,|\, s) : \mathscr{S} \to \mathscr{P}(\mathscr{A})$ - policy function

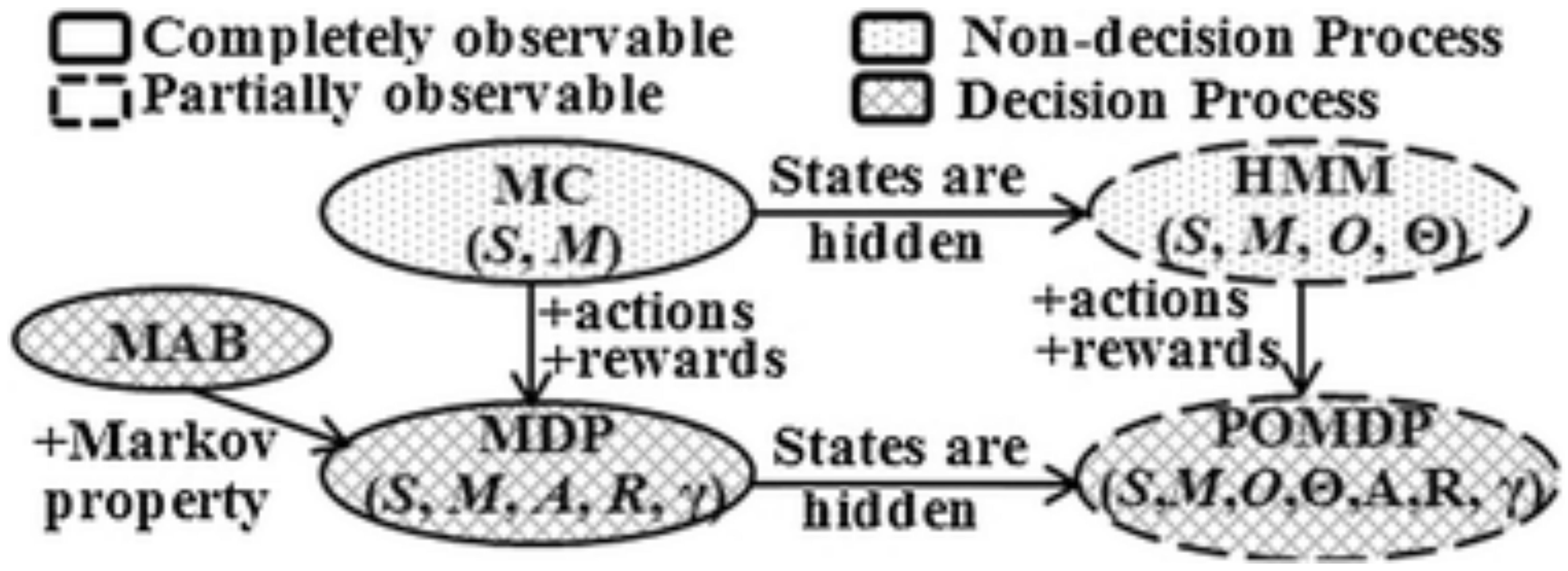
In general, unknown

Would like to find optimal

Markov Decision Process

Markov property:

$$p(R_t, S_{t+1} \mid S_t, A_t, R_{t-1}, S_{t-1}, A_{t-1}, \ldots) = p(R_t, S_{t+1} \mid S_t, A_t)$$

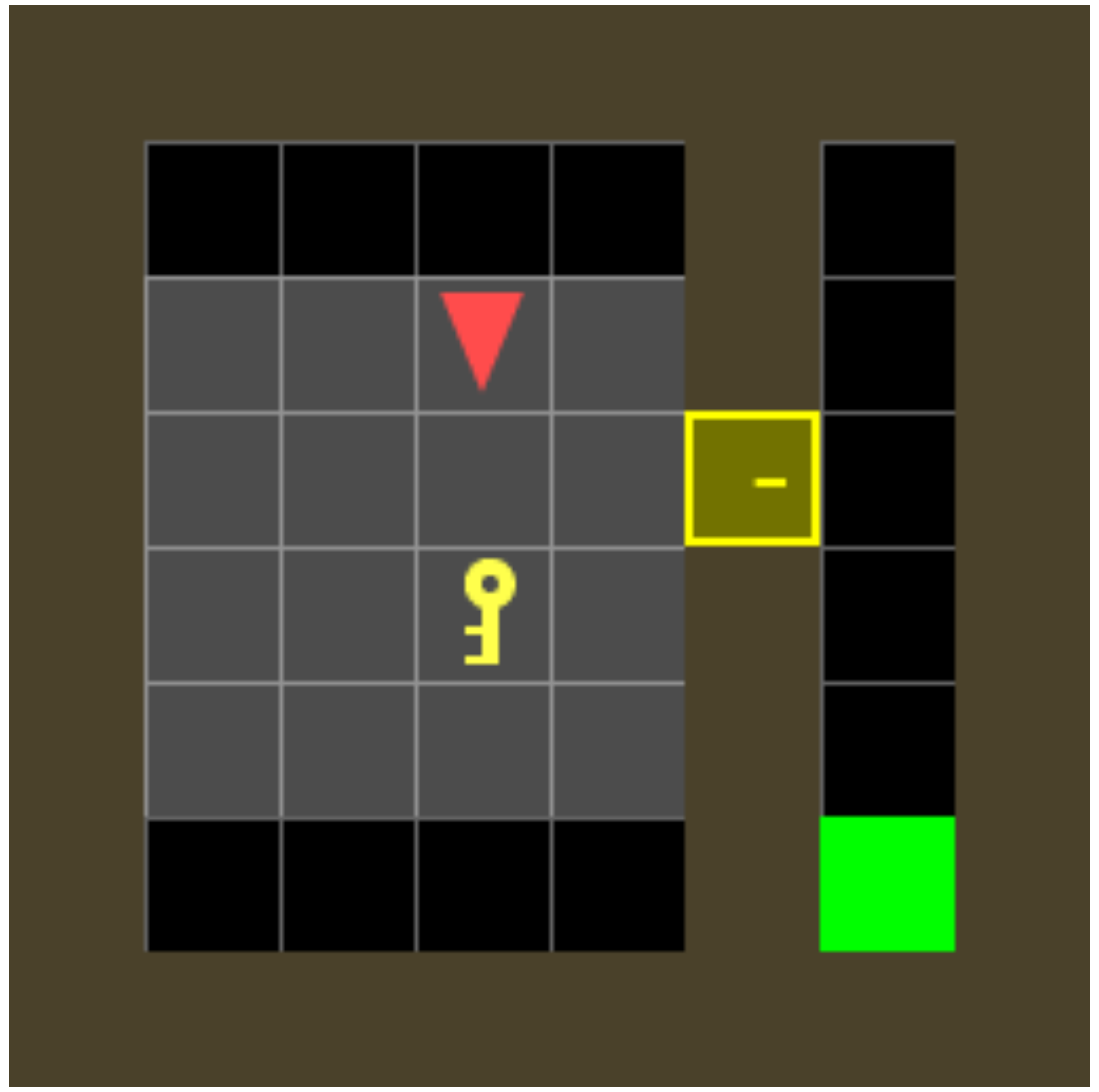


Source

Markov Decision Process

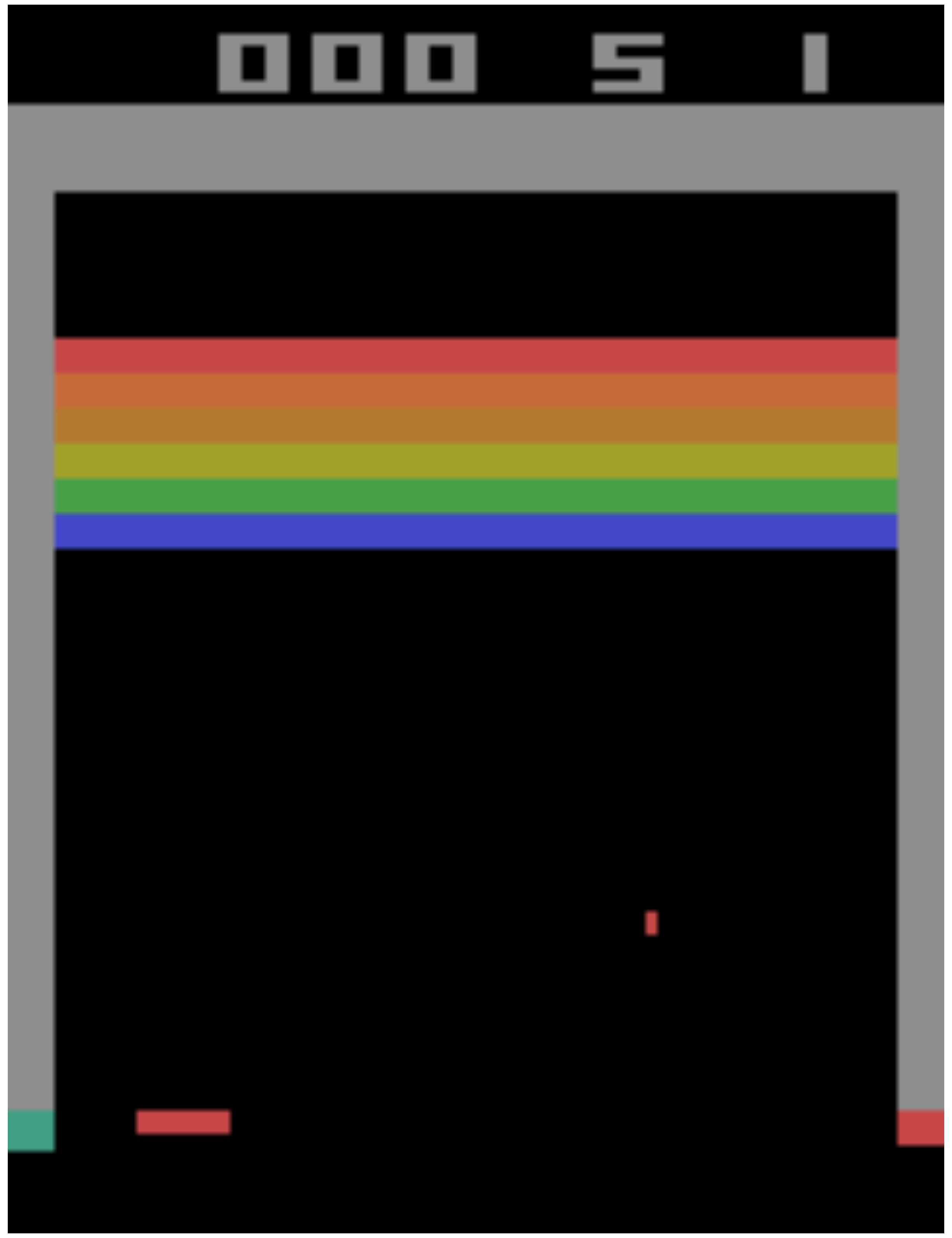
Possible states:

1. Full history
2. Coordinates
3. Coordinates + does the agent have a key
4. Image



Non Markovian Environment

Can you observe ball's direction if only this observation is available?



Reinforcement Learning Problem

1. $a \in \mathscr{A}$ - action space

2. $s \in \mathscr{S}$ - state space

2*. $o \in \mathscr{O}$ - observation space for partial observable environment

3. $r(s, a) : \mathscr{A} \times \mathscr{S} \rightarrow \mathscr{P}(\mathscr{R})$ - reward function

4. $p(r, s' \,|\, s, a)$ - environment dynamics

5. $p(s_0)$ - initial state distribution

6. $\pi(a \,|\, s) : \mathscr{S} \rightarrow \mathscr{P}(\mathscr{A})$ - policy function

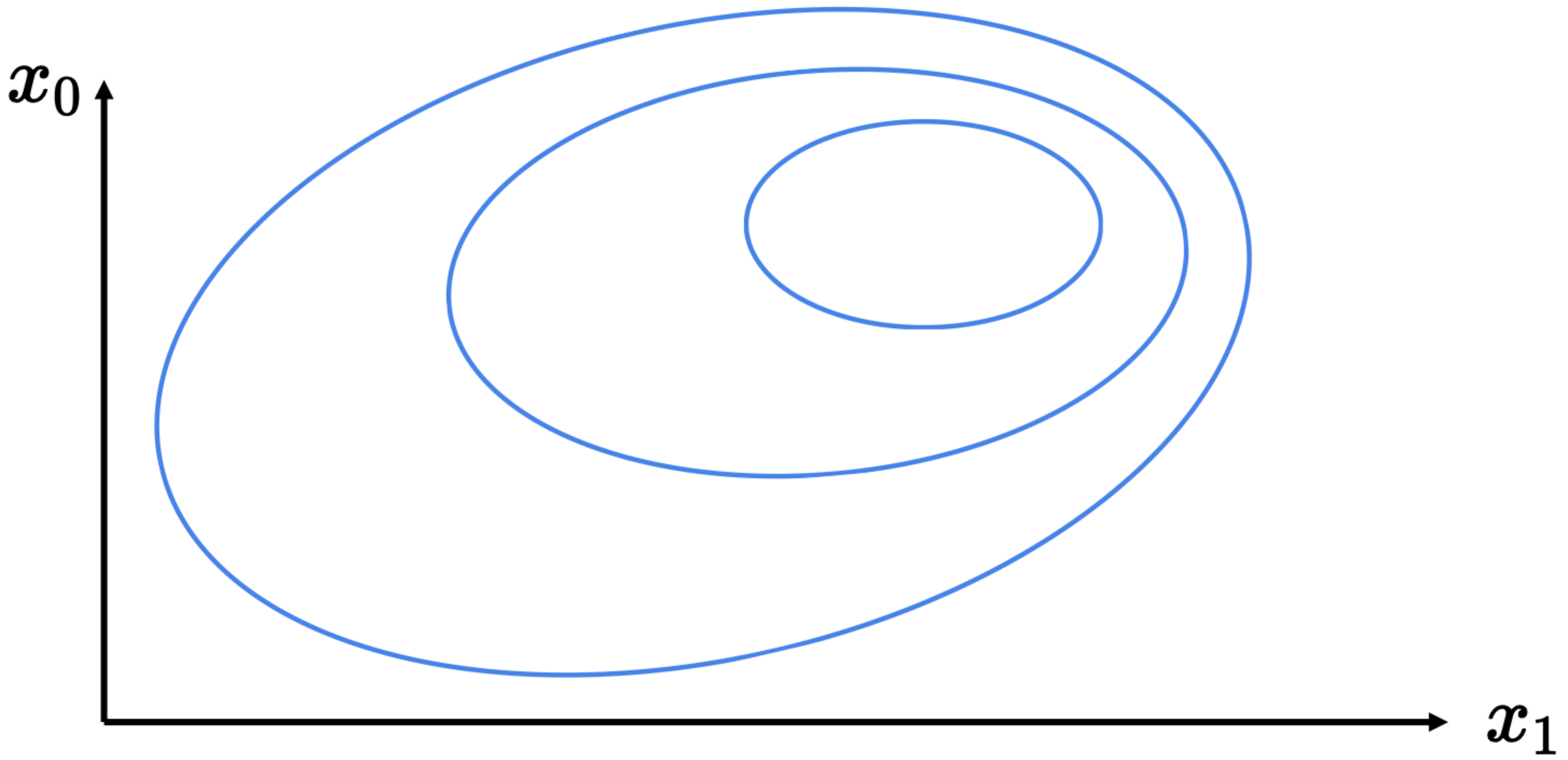
7. $\gamma \in [0,1]$ - discount factor

8. T - time horizon, can be infinite

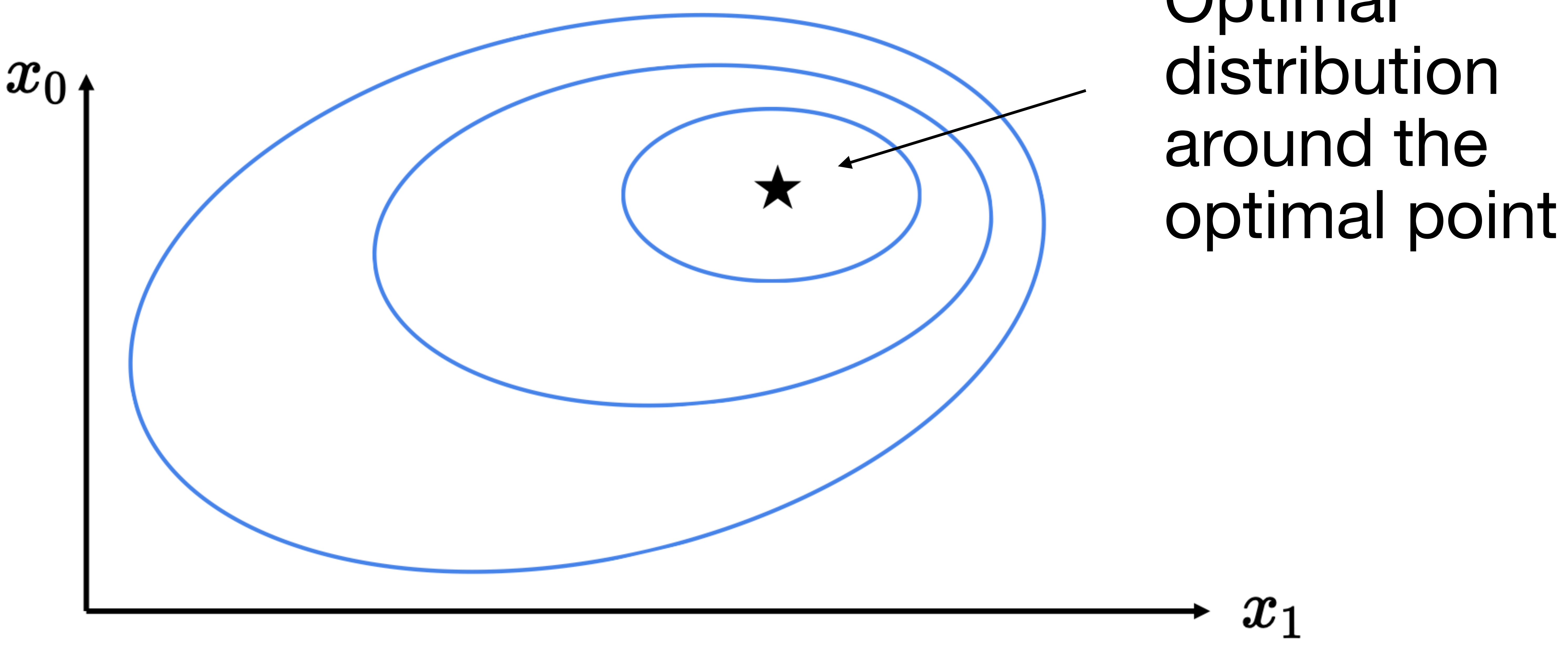
$$J(\pi) = \mathbb{E}_{p(\tau|\pi)}\left(\sum_{t=0}^{T} \gamma^t R_t\right) = \mathbb{E}_{s_0 \sim p(s_0)} \ldots \mathbb{E}_{a_t \sim \pi(a|s_t)} \mathbb{E}_{r_T, s_{T+1} \sim p(r,s'|s_t,a_t)}\left(\sum_{t=0}^{T} \gamma^t R_t\right) \rightarrow \max_{\pi}$$

Cross-entropy Method

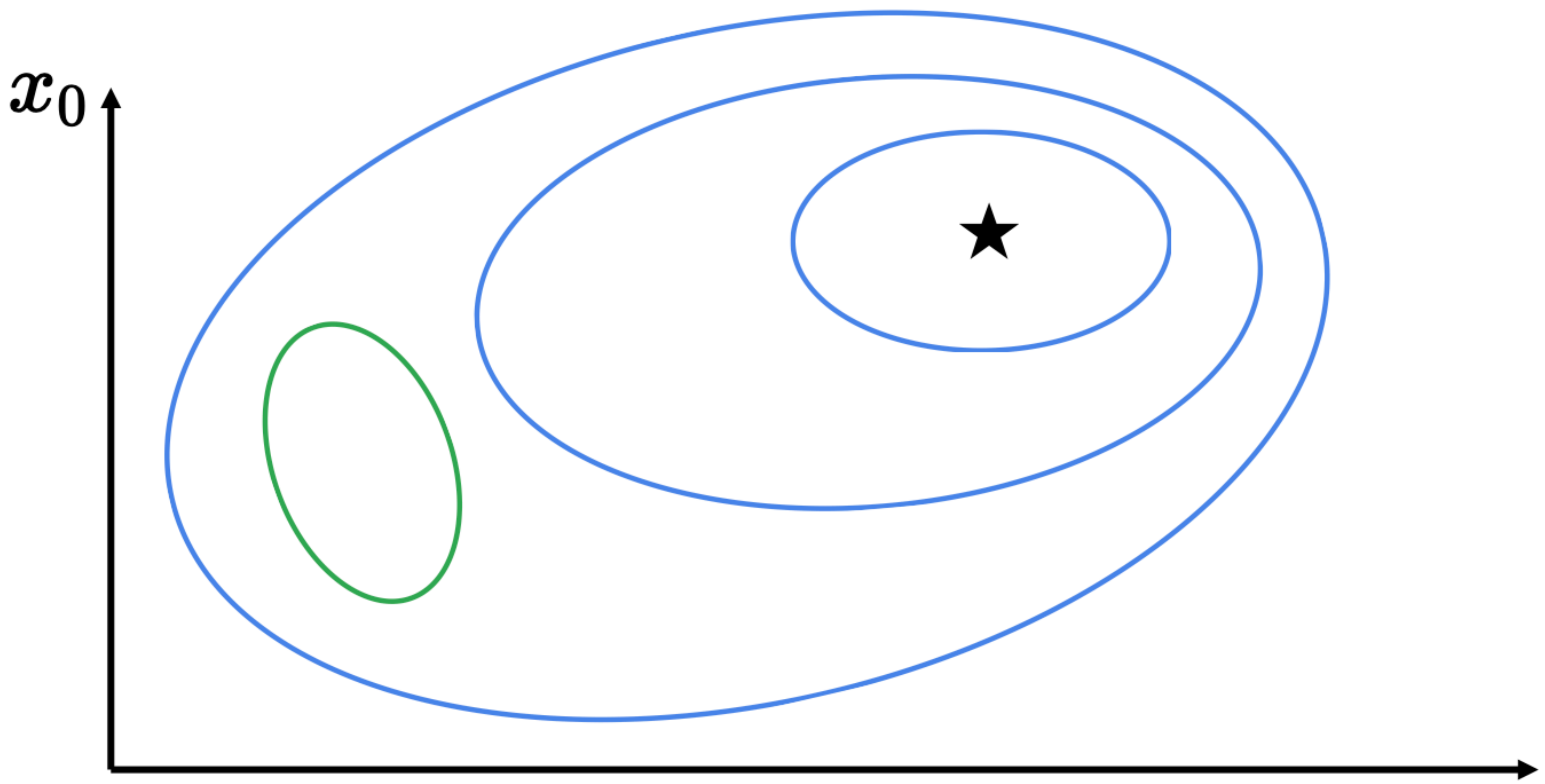
1. Zero-order optimization method
2. Try to find distribution around the optimal point (instead of point itself)



Cross-entropy Method

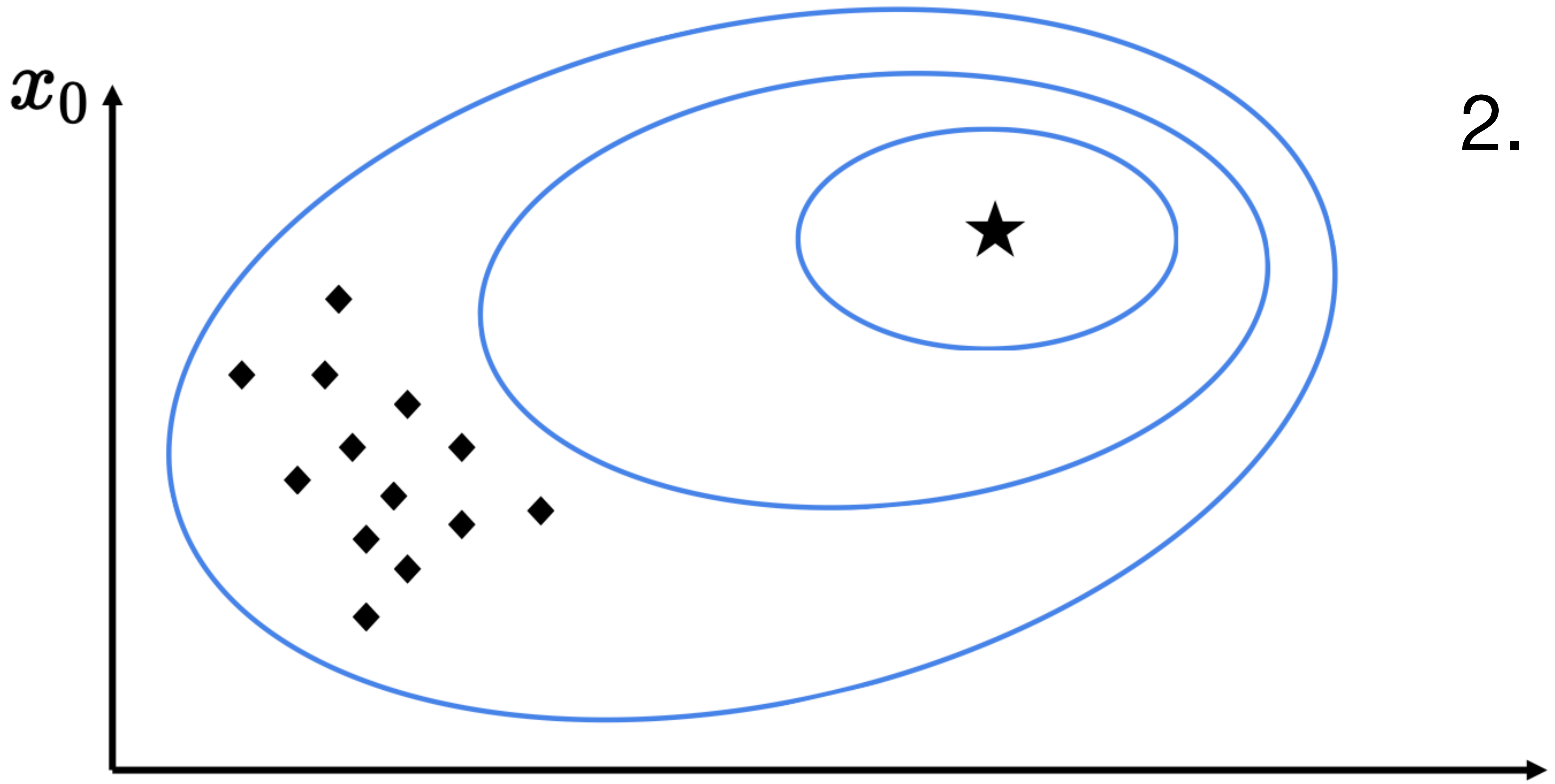


Cross-entropy Method



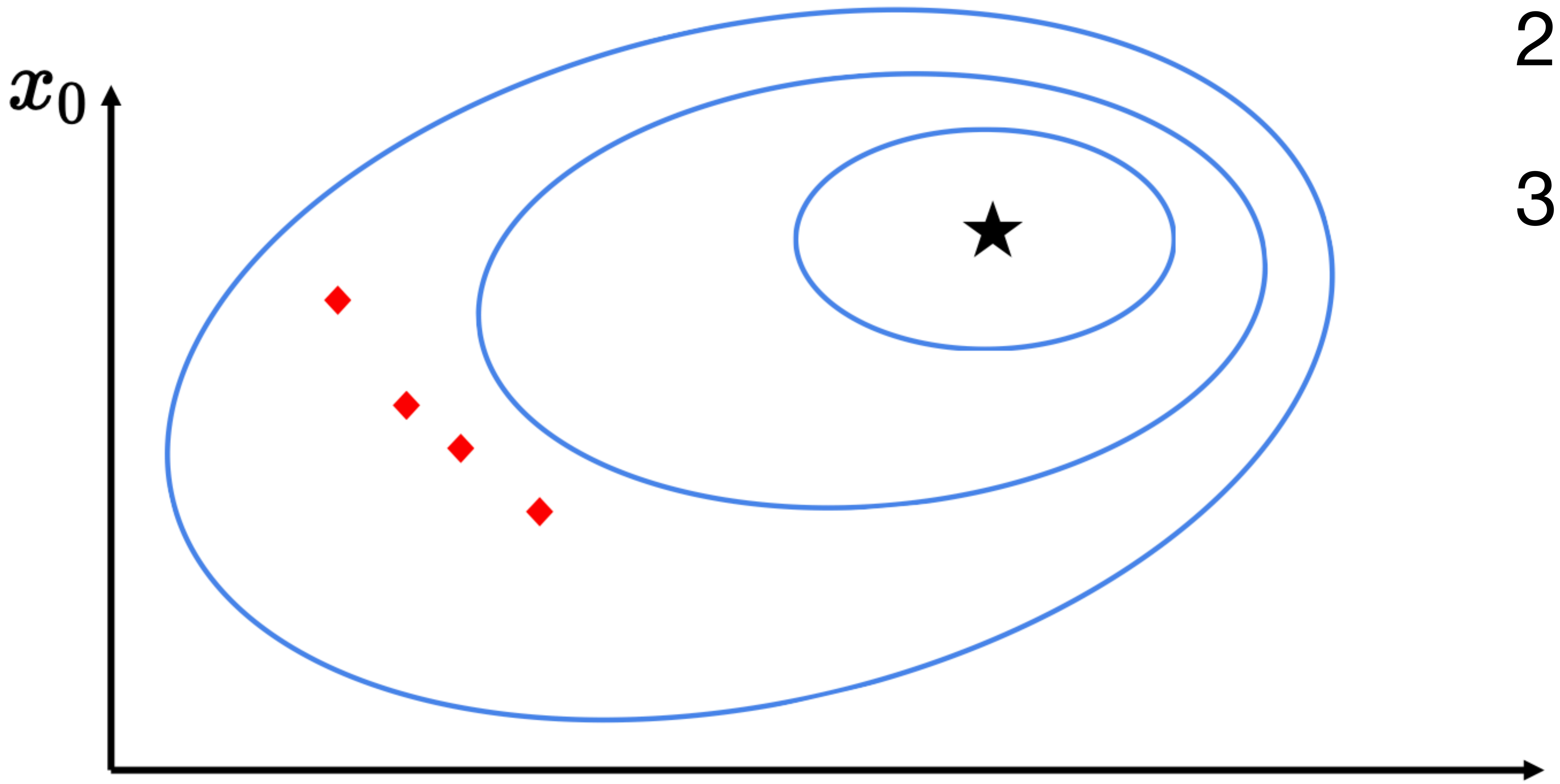
1. Initialise points distribution q_0

Cross-entropy Method



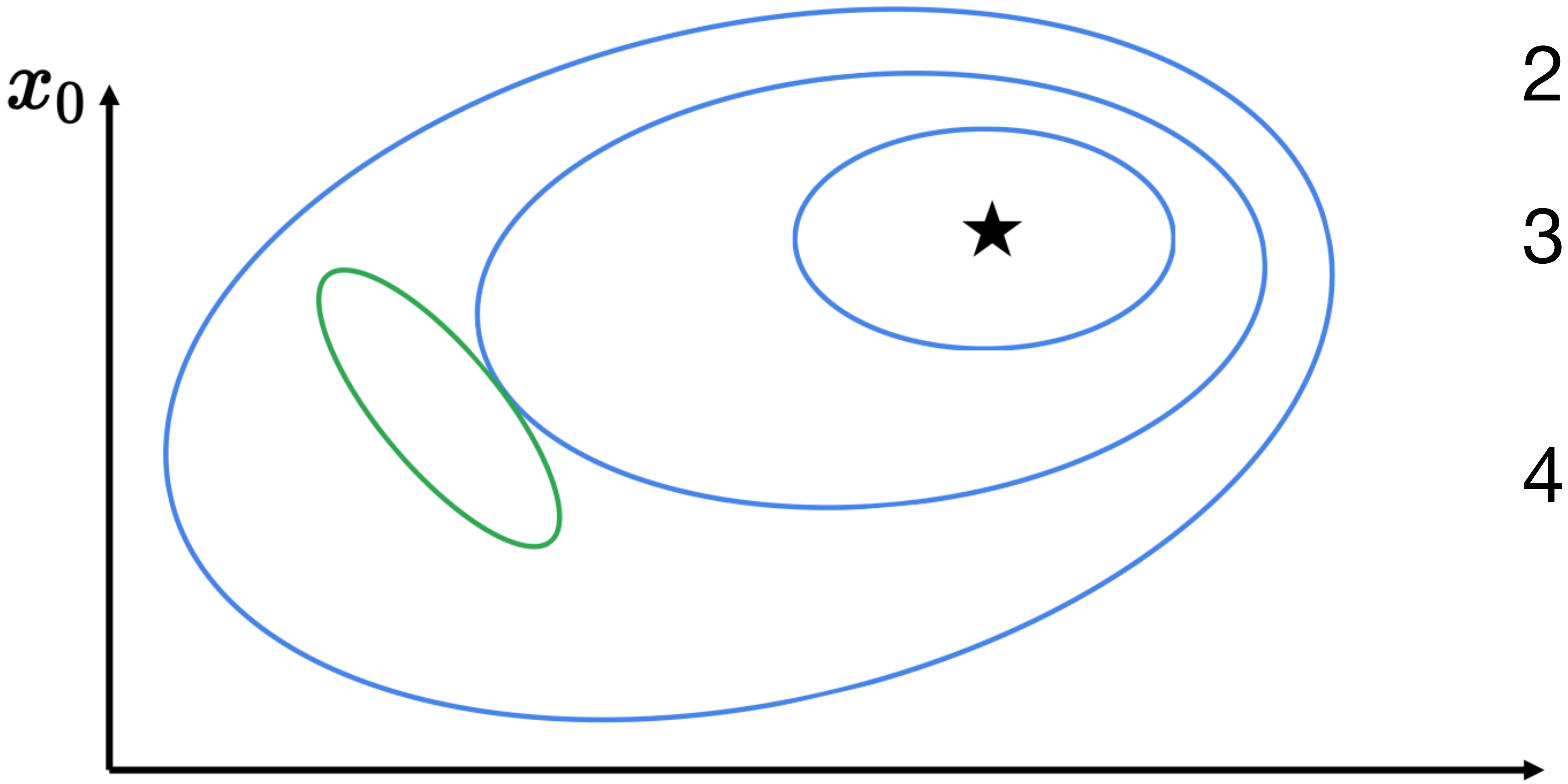
1. Initialise points distribution q_0
2. Sample from $x_i \sim q_0$

Cross-entropy Method



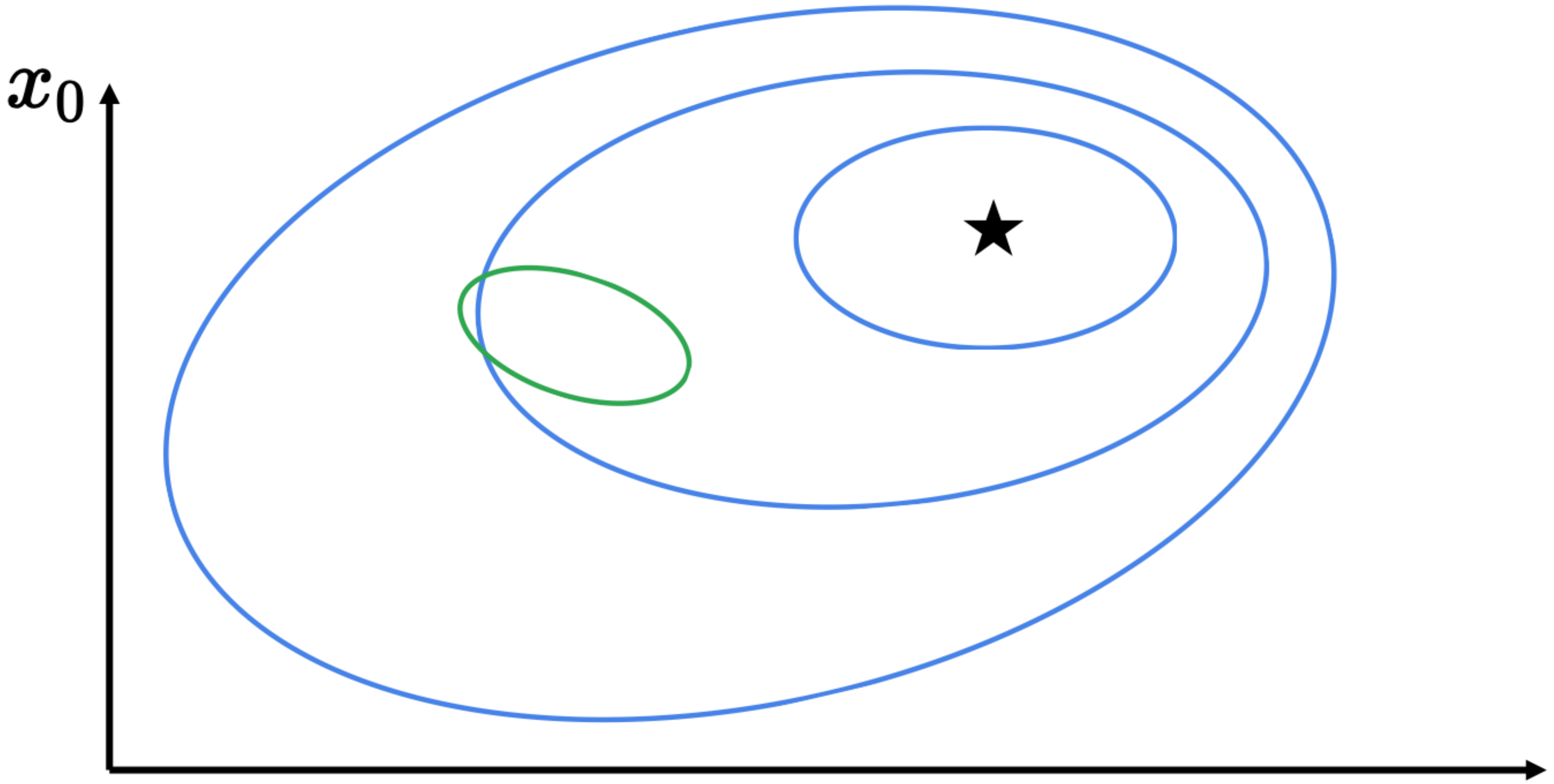
1. Initialise points distribution q_0
2. Sample from $x_i \sim q_0$
3. Select M elite point based on criteria

Cross-entropy Method



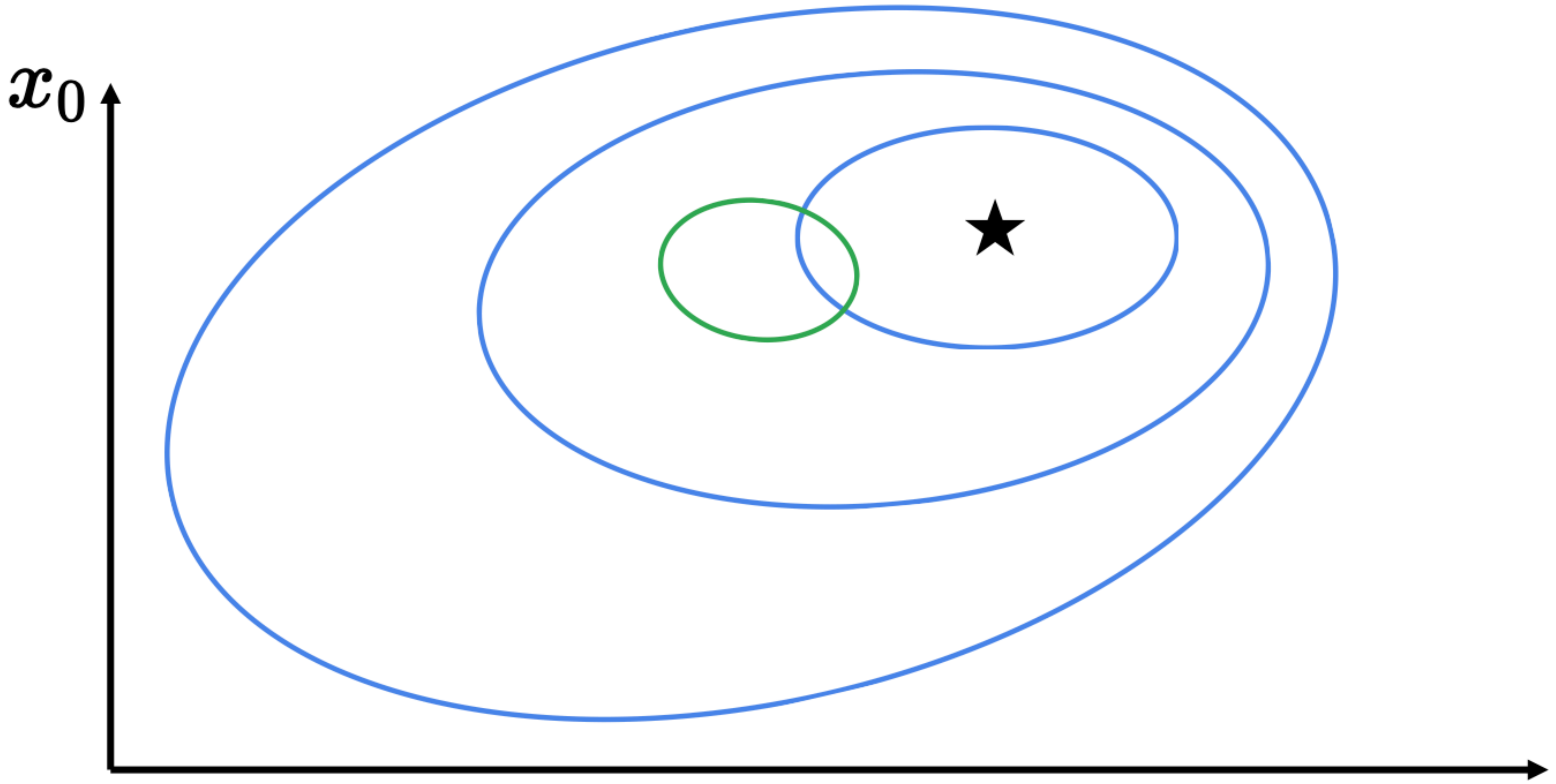
1. Initialise points distribution q_0
2. Sample from $x_i \sim q_0$
3. Select M elite point based on criteria
4. Update to get new distribution q_1

Cross-entropy Method



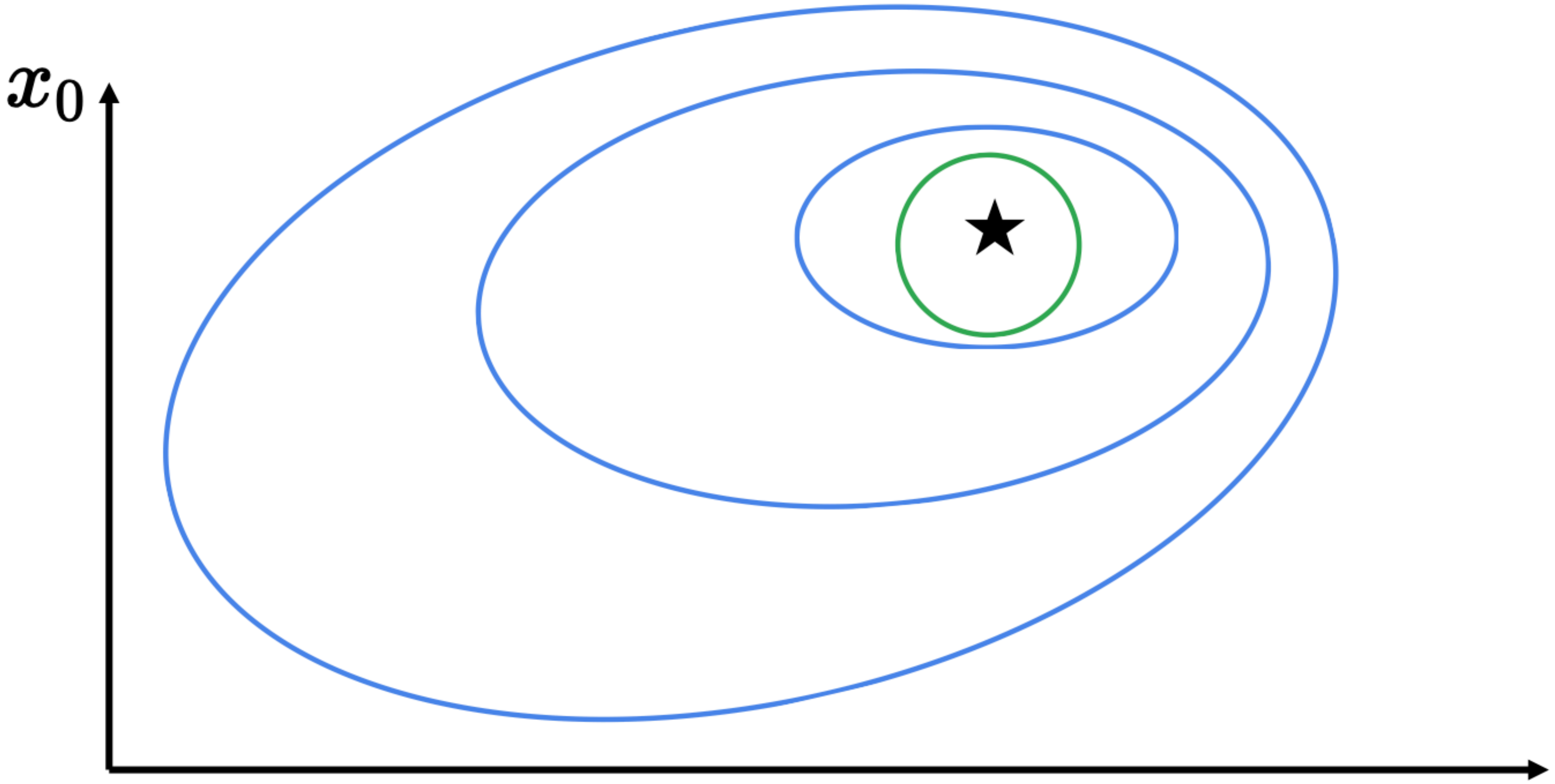
1. Initialise points distribution q_0
2. Sample from $x_i \sim q_0$
3. Select M elite point based on criteria
4. Update to get new distribution q_1
5. Repeat 2-4 until convergence

Cross-entropy Method



1. Initialise points distribution q_0
2. Sample from $x_i \sim q_0$
3. Select M elite points based on some criteria
4. Update to get new distribution q_1
5. Repeat 2-4 until convergence

Cross-entropy Method



1. Initialise points distribution q_0
2. Sample from $x_i \sim q_0$
3. Select M elite point based on criteria
4. Update to get new distribution q_1
5. Repeat 2-4 until convergence

Cross-entropy Method

Distribution update:

$$KL(p_{\text{data}}||q) = \mathbb{E}_{p_{\text{data}}} \log(\frac{p_{\text{data}}}{q}) = -\mathbb{E}_{x \sim p_{\text{data}}} \log q(x) \to \min_q$$

Distribution update: $q^{k+1} = \text{argmin}_q[-\frac{1}{|\mathcal{M}^k|} \sum_{x \in \mathcal{M}^k} \log q(x)],$

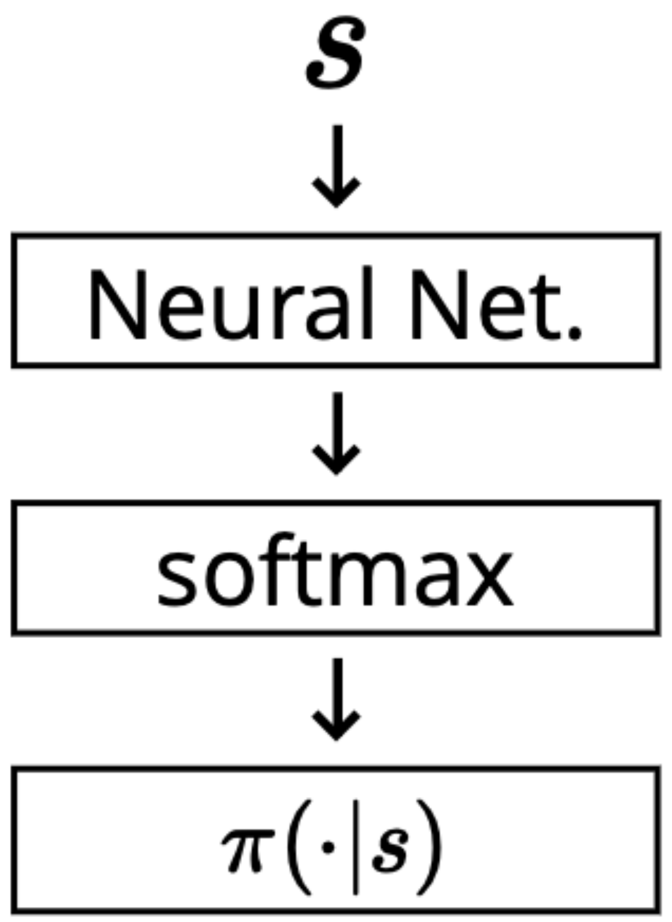
where $\mathcal{M}^k$ is a set of elites points from the k-th iteration

Deep CME for RL

$$J(\pi) = \mathbb{E}_{p(\tau|\pi)}\left(\sum_{t=0}^{T} \gamma^t R_t\right) \to \max_{\pi}$$

Let $\pi = \pi_\theta$ is a neural network

Discrete actions



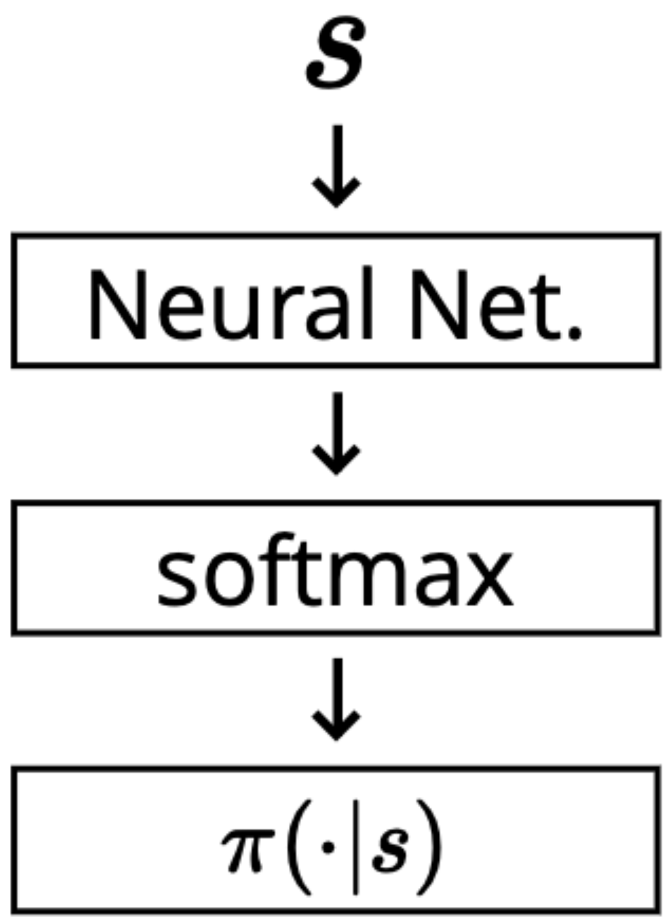
Continuous actions

Deep CME for RL

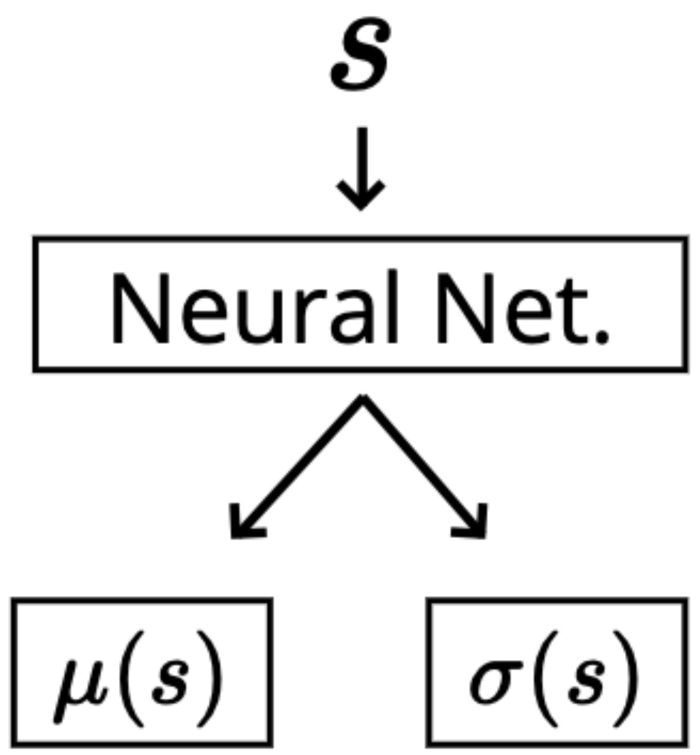
$$J(\pi) = \mathbb{E}_{p(\tau|\pi)}\left(\sum_{t=0}^{T} \gamma^t R_t\right) \to \max_{\pi}$$

Let $\pi = \pi_\theta$ is a neural network

Discrete actions



Continuous actions



Deep CME for RL

1. Input: elite percentile α
2. Initialize $\pi = \pi_\theta$ as a neural network
3. Repeat until convergence:
    1. Run agent π_θ to collect trajectories $\mathcal{T} = \{\tau_i\}$
    2. $\delta =$ percentile$(\mathcal{T}, p = \alpha)$
    3. $\mathcal{M} =$ select_elites$(\mathcal{T}, \delta)$
    4. Partial fit π_θ to predict a_i based on s_i, $(a_i, s_i) \in \mathcal{M}$

Bonus

Evolution Strategies as a Scalable Alternative to Reinforcement Learning

Thank you for your attention!