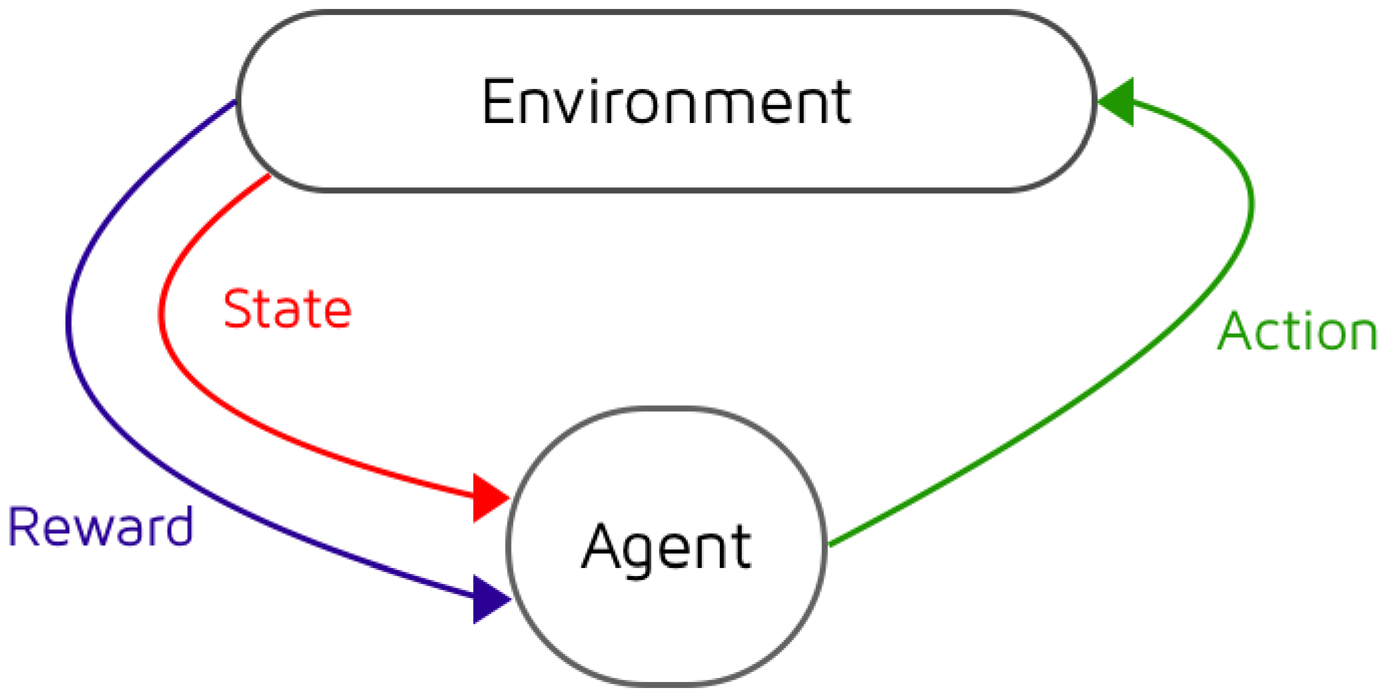
Lecture 1.0

Reinforcement Learning: MDPs

Alexey Gruzdev
alexey.s.gruzdev@gmail.com
HSE, Winter 2020

RL Mechanics



Major Components of an RL Agent

- An RL agent may include one or more of these components:
  - Policy: agent’s behavior function
  - Value function: how good is each state and/or action
  - Model: agent’s representation of the environment

Policy

- A policy is the agent's behavior
- It is a map from state to action, e.g.
- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a \mid s) = \mathbb{P}[A_t = a \mid S_t = s]$

Value Function

- Value function is a prediction of future reward
- Used to evaluate the goodness/badness of states
- And therefore to select between actions, e.g.

$$v_\pi(s) = \mathrm{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s]$$

Model

- A model predicts what the environment will do next

- ***P*** predicts the next state
- ***R*** predicts the next (immediate) reward, e.g.

$$\boldsymbol{P}_{ss'}{}^{a} = \mathrm{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$
$$\boldsymbol{R}_{s}^{a} = \mathrm{E}\,[R_{t+1} \mid S_t = s, A_t = a]$$

Rewards hypothesis revisited

- A reward R_t is a scalar feedback signal
- Indicates how well agent is doing at step t
- The agent's job is to maximize cumulative reward

Reinforcement Learning is based on the reward hypothesis.

The reward hypothesis:

All goals can be described by the maximization of expected cumulative reward .

- Online banners recommender system
- Personal mobile-phone assistants

Markov Processes Family

- Markov Processes (Markov Chain)
- Markov Reward Processes
- Markov Decision Processes
- Extensions to MDPs:
  - Infinite & Continuous MDP
  - POMDP
  - Undiscounted MDP

Introduction to MDPs

- *Markov decision processes* formally describe an environment for reinforcement learning
- Where the environment is *fully observable*
- i.e. The current *state* completely characterizes the process
- Almost all RL problems can be formalized as MDPs, e.g.
  - Optimal control primarily deals with continuous MDPs
  - Partially observable problems can be converted into MDPs
  - Bandits are MDPs with one state

Markov Property

- <u>Definition:</u> a state S_t is Markov if and only if

$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, \ldots, S_t]$$

- "The future is independent of the past given the present"
  - The state captures all relevant information from the history
  - Once the state is known, the history may be thrown away
  - i.e. The state is a sufficient statistic of the future

State Transition Matrix

- For a Markov state *s* and successor state *so*, the *state transition probability* is defined by

$$\boldsymbol{P}_{ss'} = \mathrm{P}\left[S_{t+1} = s' \mid S_t = s\right]$$

- State transition matrix $\boldsymbol{P}_{ss'}$ defines transition probabilities from all states *s* to all successor states *s'*

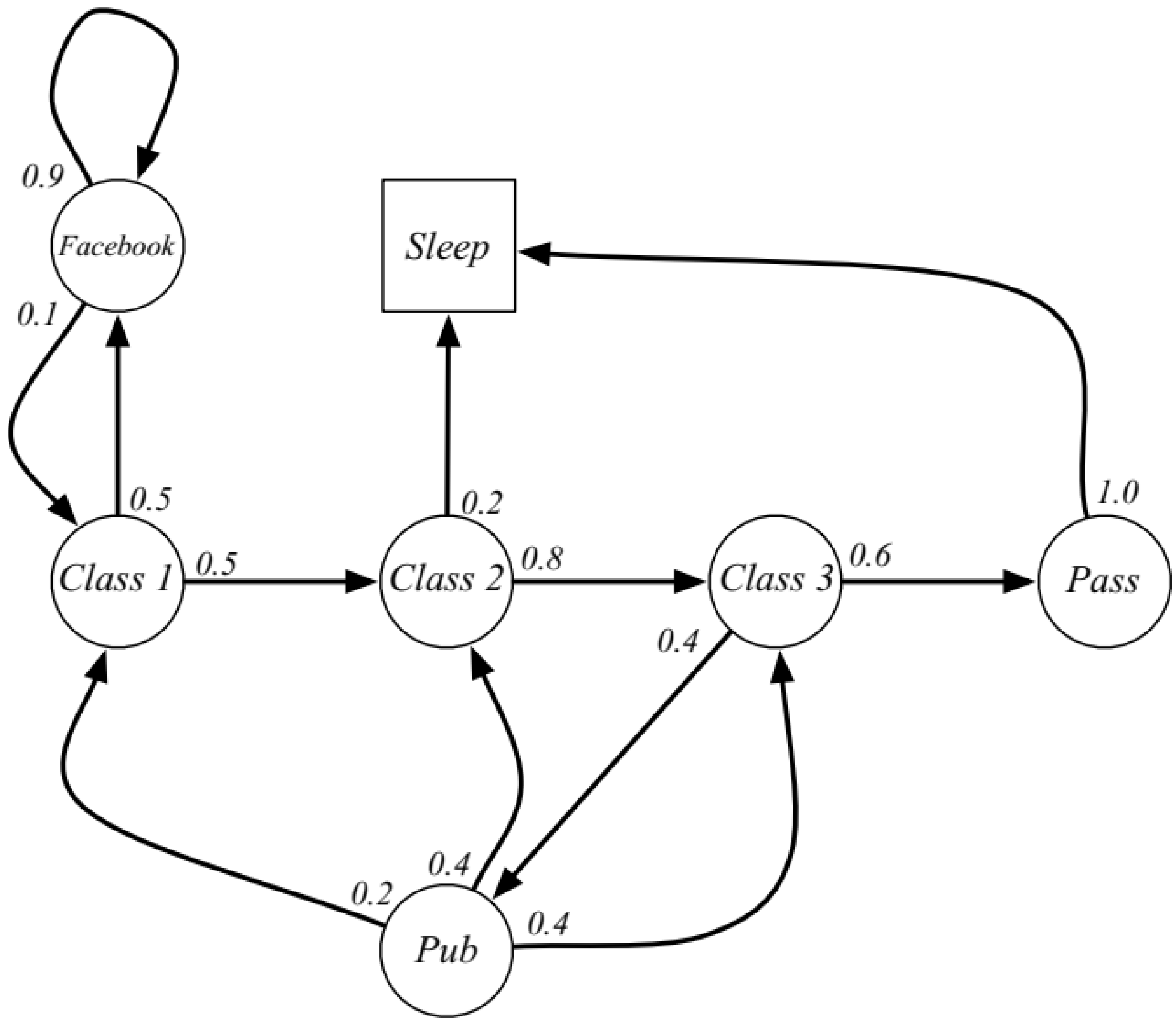
$$\boldsymbol{P}_{ss'} = \begin{pmatrix} \boldsymbol{P}_{11} & \cdots & \boldsymbol{P}_{1n} \\ \vdots & \ddots & \vdots \\ \boldsymbol{P}_{n1} & \cdots & \boldsymbol{P}_{nn} \end{pmatrix}$$

Markov Process

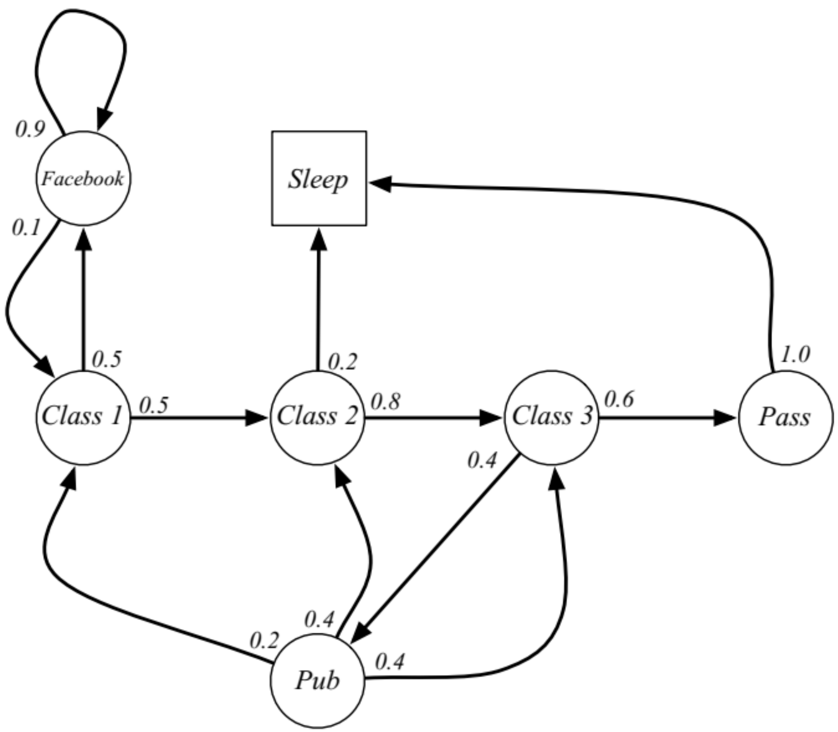
- Markov process is a memoryless random process, i.e. a sequence of random states $\boldsymbol{S_1, \ldots, S_t}$ with the Markov property.

- *A Markov Process* (or *Markov Chain*) is a tuple $(\boldsymbol{S}, \boldsymbol{P})$
  - $\boldsymbol{S}$ is a (finite) set of states
  - $\boldsymbol{P}_{ss'}$ is a state transition probability matrix
  - $\boldsymbol{P}_{ss'} = \mathrm{P}[S_{t+1} = s' \mid S_t = s]$

Markov Process: Student Example



Markov Process: Episodes Sampling

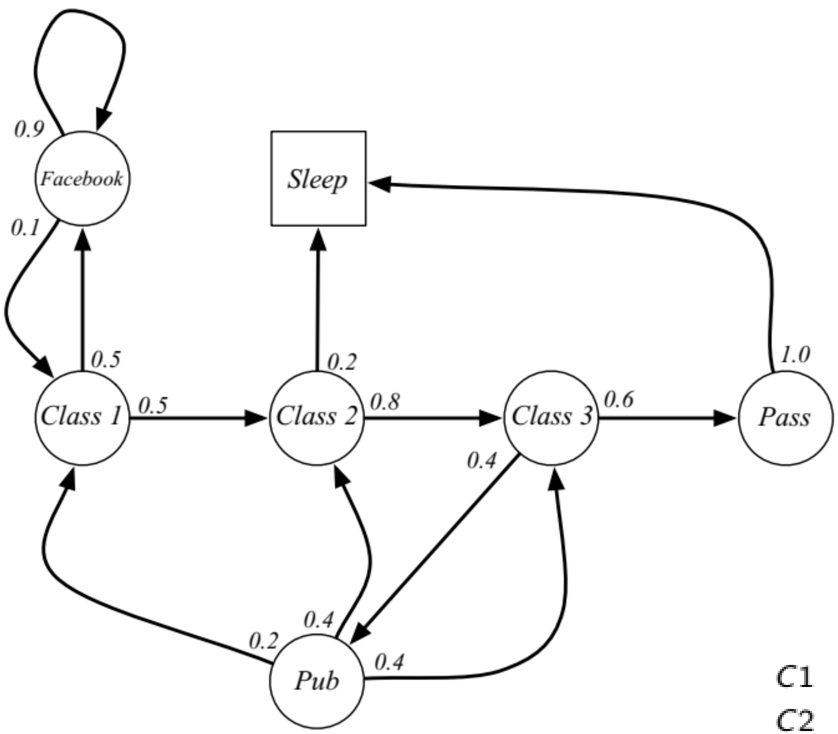


- Sample episodes for Student Markov Chain starting from $S_1 = C1$

$$S_1, \ldots, S_t$$

- C1 C2 C3 Pass Sleep
- C1 FB FB C1 C2 Sleep
- C1 C2 C3 Pub C2 C3 Pass Sleep
- C1 FB FB C1 C2 C3 Pub C1 FB FB FB C1 C2 C3 Pub C2 Sleep

Markov Process: Transition Probabilities



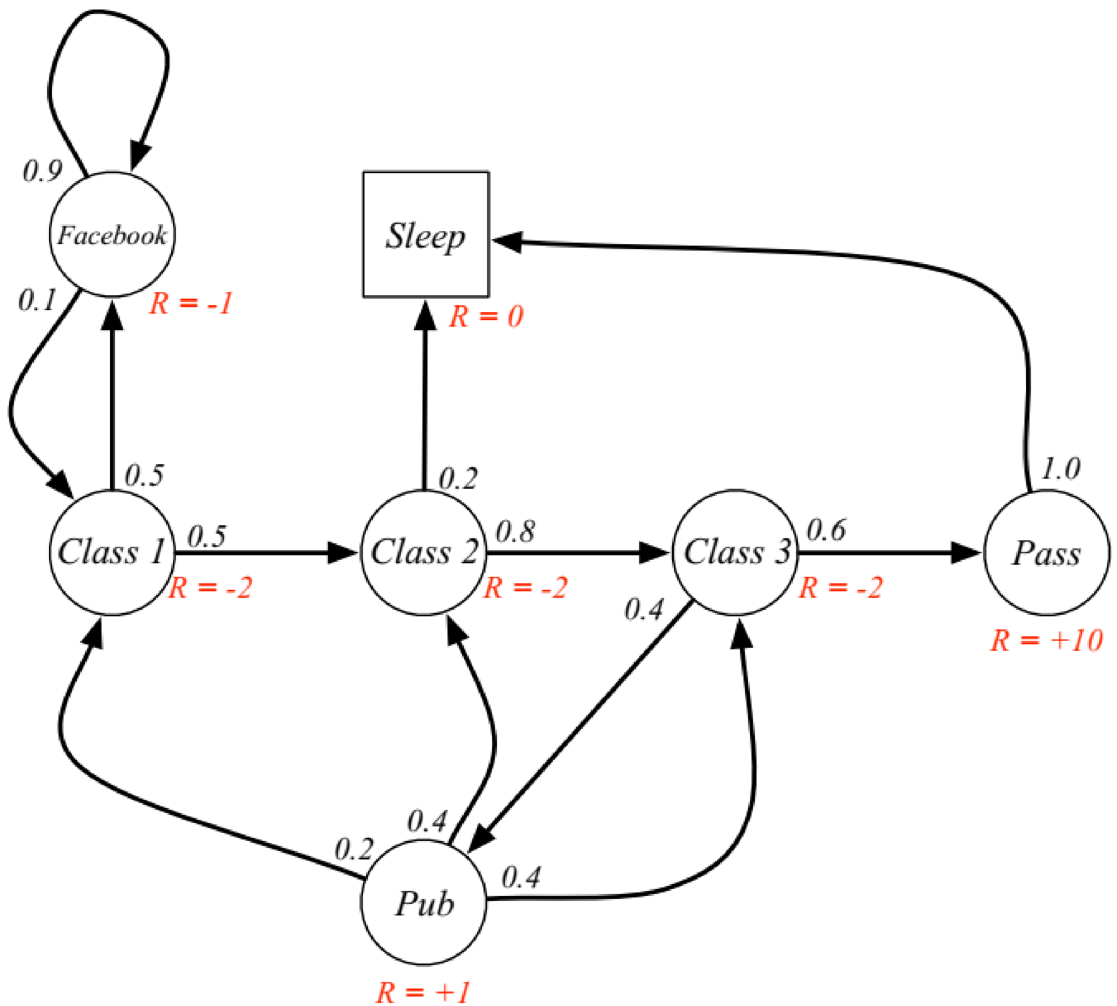
$$\mathcal{P} = \begin{array}{cc} & \begin{array}{ccccccc} C1 & C2 & C3 & Pass & Pub & FB & Sleep \end{array} \\ \begin{array}{c} C1 \\ C2 \\ C3 \\ Pass \\ Pub \\ FB \\ Sleep \end{array} & \begin{bmatrix} & 0.5 & & & & 0.5 & \\ & & 0.8 & & & & 0.2 \\ & & & 0.6 & 0.4 & & \\ & & & & & & 1.0 \\ 0.2 & 0.4 & 0.4 & & & & \\ 0.1 & & & & & 0.9 & \\ & & & & & & 1 \end{bmatrix} \end{array}$$

Markov Reward Process

- A Markov reward process is a Markov chain with values.

- *A Markov Reward Process* is a tuple *(**S**, **P**, **R**, **γ**)*
  - ***S*** is a (finite) set of states
  - $\boldsymbol{P}_{ss'}$ is a state transition probability matrix
  - $\boldsymbol{P}_{ss'} = \mathrm{P}[S_{t+1} = s' \mid S_t = s]$
  - ***R*** is a reward function, $\boldsymbol{R_s} = \mathbf{E}\,[\boldsymbol{R_{t+1}} \mid \boldsymbol{S_t} = \boldsymbol{s}]$
  - ***γ*** is a discount factor, $\gamma \in [0, 1]$

MRP: Student Example



Return

- The *return* G_t is the total discounted reward from time-step t.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{t=0}^{\infty} \gamma^t R_{t+k+1}$$

- The *discount* $\gamma \in [0, 1]$ is the present value of future rewards
- The value of receiving reward R after $k + 1$ time-steps is $\gamma^k R$.
- This values immediate reward above delayed reward.
  - γ close to 0 leads to "myopic" evaluation
  - γ close to 1 leads to "far-sighted" evaluation

Discount intuition

Most Markov reward and decision processes are discounted. Why?

- Mathematically convenient to discount rewards
- Avoids infinite returns in cyclic Markov processes
- Uncertainty about the future may not be fully represented
- If the reward is financial, immediate rewards may earn more interest than delayed rewards
- Animal/human behavior shows preference for immediate reward
- It is sometimes possible to use *undiscounted* Markov reward processes (i.e. $\gamma = 1$), e.g. if all sequences terminate

Value Function

- The value function $v(s)$ gives the long-term value of state s

- The *state value function* $v(s)$ of an MRP is the expected return starting from state s

$$v(s) = \mathrm{E}[G_t \mid S_t = s]$$

Student MRP returns

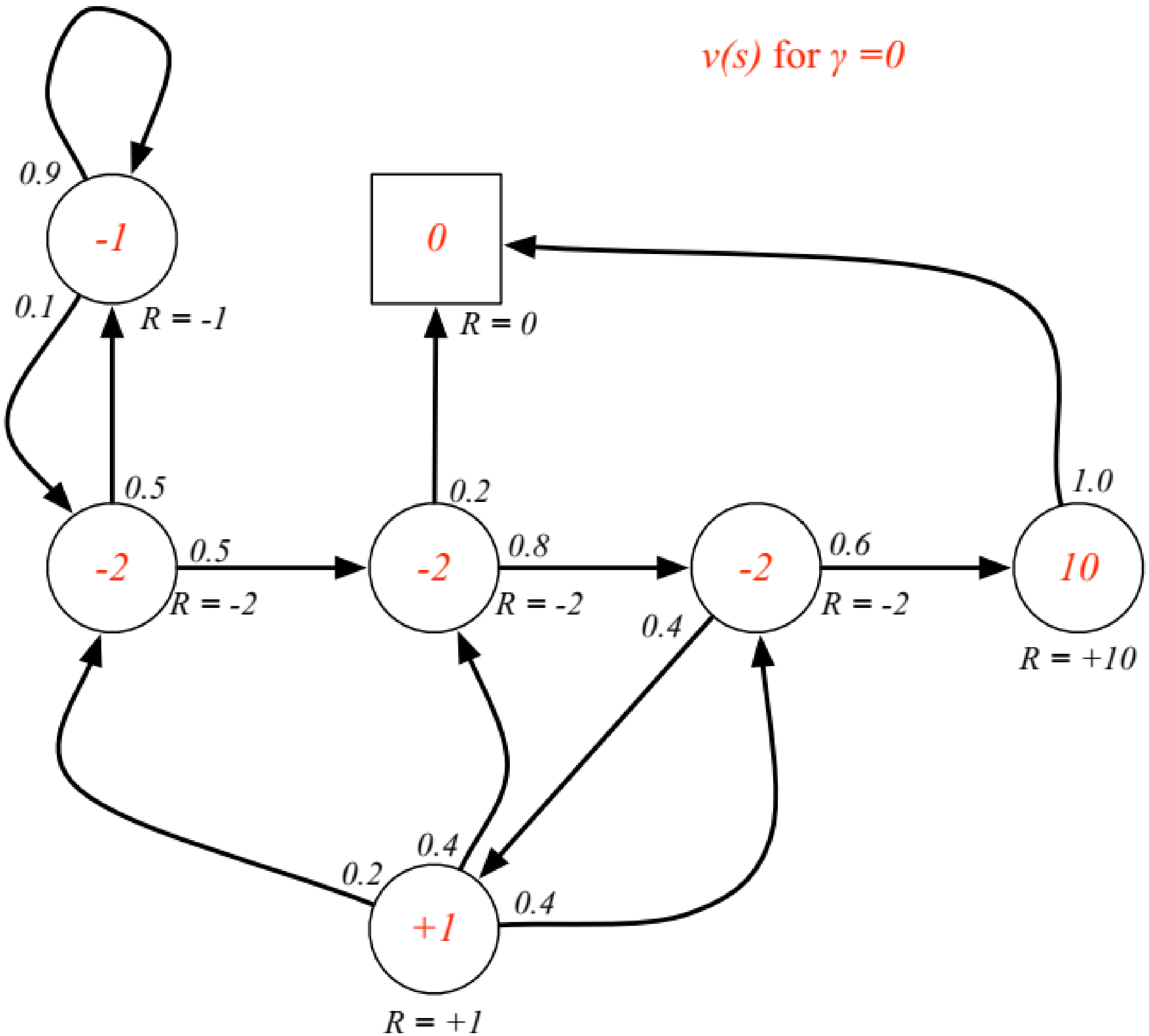
Sample returns for Student MRP with:

- $S_1 = C1$
- $\gamma = 0.5$

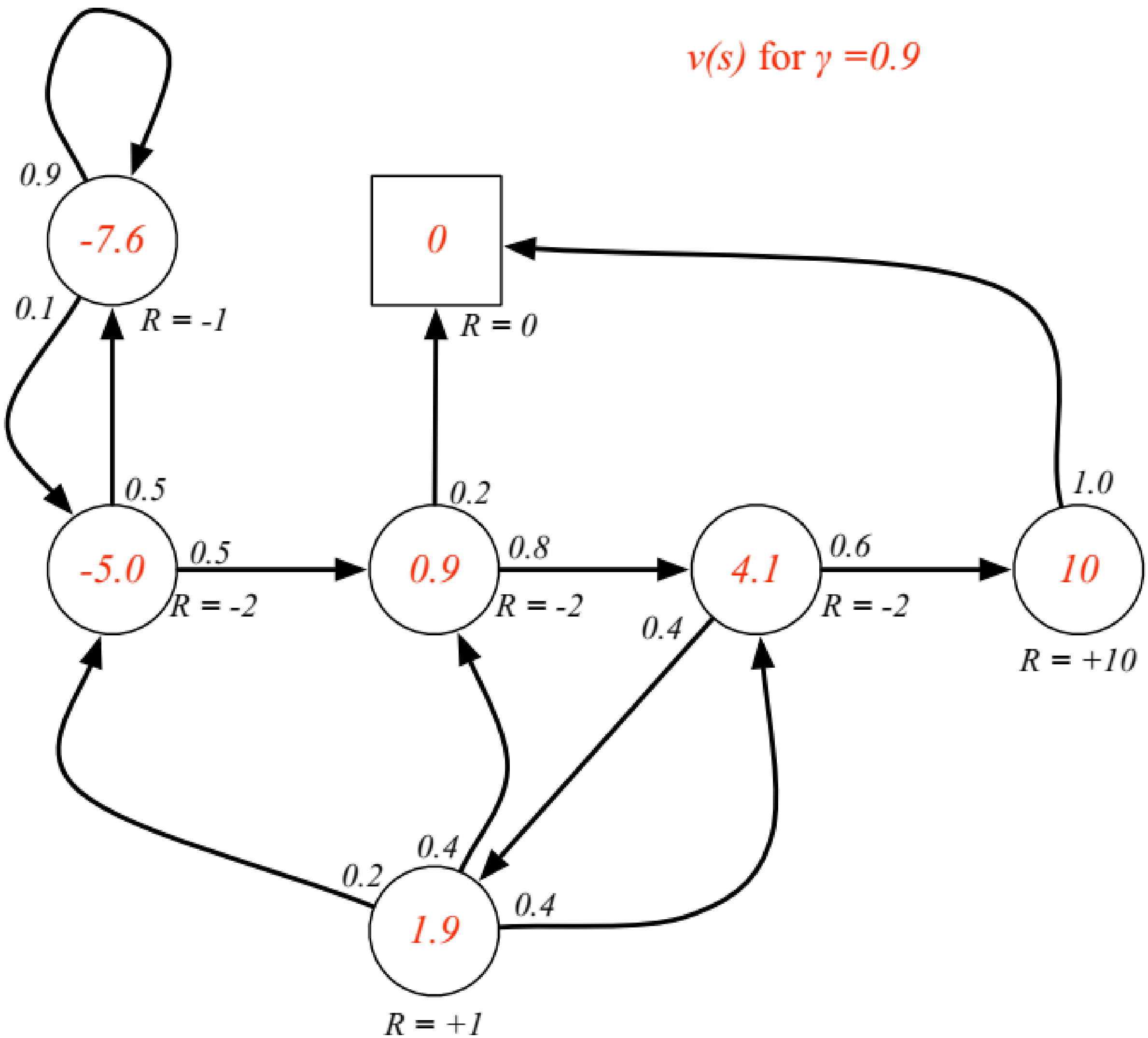
$$G_1 = R_2 + \gamma R_3 + ... + \gamma^{T-2} R_T$$

C1 C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 10 * \frac{1}{8}$	$=$	-2.25
C1 FB FB C1 C2 Sleep	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16}$	$=$	-3.125
C1 C2 C3 Pub C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 1 * \frac{1}{8} - 2 * \frac{1}{16} \ldots$	$=$	-3.41
C1 FB FB C1 C2 C3 Pub C1 ...	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16} \ldots$	$=$	-3.20
FB FB FB C1 C2 C3 Pub C2 Sleep			

State-Value function for student MRP



State-Value function for student MRP



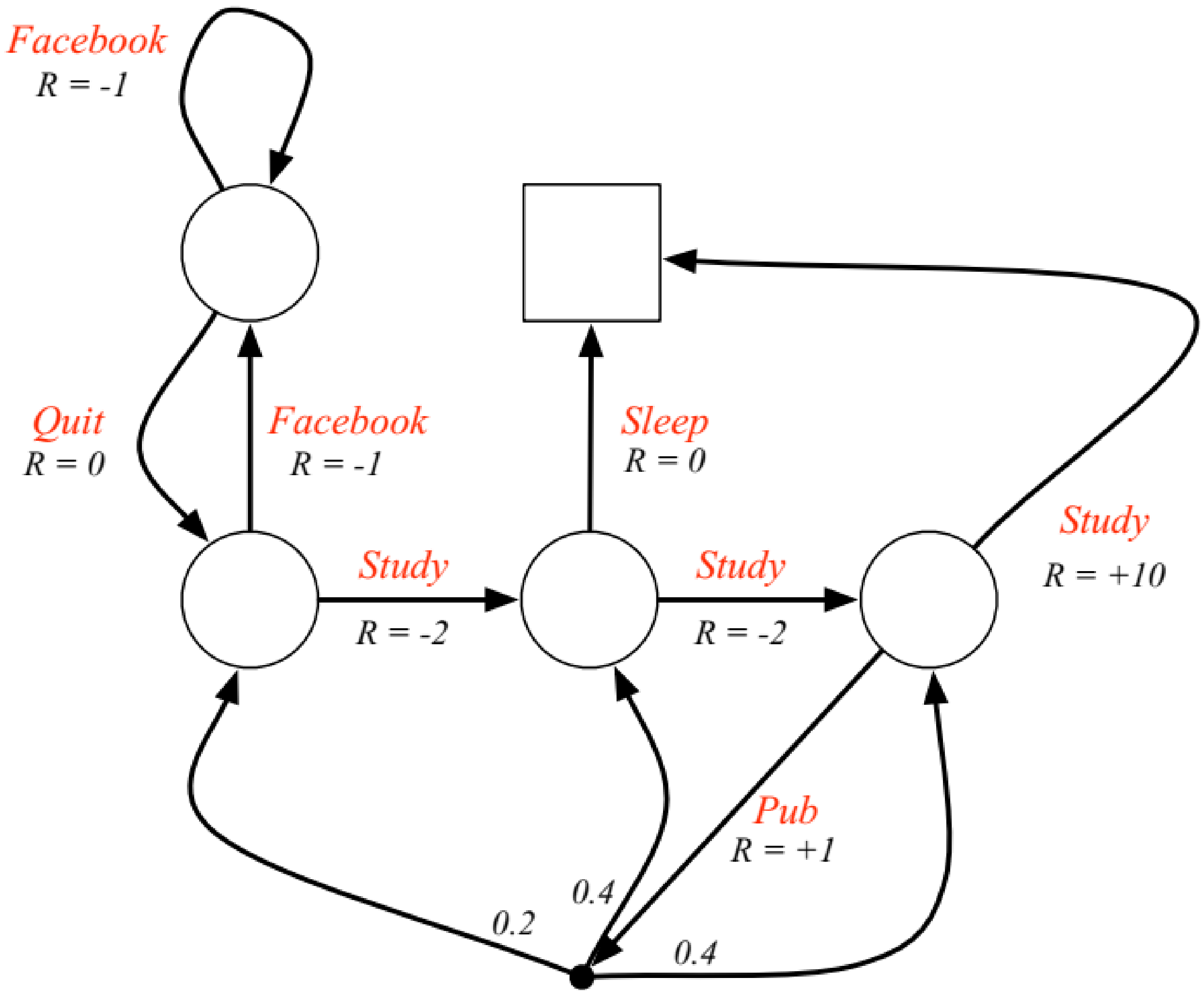
Markov Decision Process

- A Markov decision process (MDP) is a Markov reward process with
  decisions. It is an *environment* in which all states are Markov.

- A Markov reward process is a Markov chain with values.

- *A Markov Decision Process* is a tuple $(\boldsymbol{S}, A, \boldsymbol{P}, \boldsymbol{R}, \boldsymbol{\gamma})$
  - $\boldsymbol{S}$ is a (finite) set of states
  - $\boldsymbol{A}$ is a finite set of actions
  - $\boldsymbol{P}^{a}_{ss'}$ is a state transition probability matrix
  - $\boldsymbol{P}^{a}_{ss'} = \mathrm{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$
  - $\boldsymbol{R}$ is a reward function, $\boldsymbol{R^{a}_{s} = \mathrm{E}\,[R_{t+1} \mid S_t = s, A_t = a]}$
  - $\boldsymbol{\gamma}$ is a discount factor, $\gamma \in [0, 1]$

Markov Decision Process: Example



MDP Policies

A *policy* π is a distribution over actions given states

$$\pi(a \mid s) = \mathbb{P}[A_t = a \mid S_t = s]$$

- A policy fully defines the behavior of an agent
- MDP policies depend on the current state (not the history)
- i.e. Policies are *stationary* (time-independent), $A_t \sim \pi(\cdot \mid S_t), \forall t > 0$

MDP Policies

- Given an MDP $M = (\boldsymbol{S}, \mathbf{A}, \boldsymbol{P}, \boldsymbol{R}, \boldsymbol{\gamma})$ and a policy π
- The state sequence $\boldsymbol{S_1, \ldots, S_t}$ is a Markov process (S, P^{π})
- The state and reward sequence $S_1, R_2, S_2, \ldots,$ is a Markov reward process $(S, P^{\pi}, R^{\pi}, \gamma)$
- where

$$P^{\pi}{}_{s,s'} = \sum_{a \in A} \pi(a \mid s) P^{a}{}_{s,s'}$$

$$R^{\pi}{}_{s} = \sum_{a \in A} \pi(a \mid s) R^{a}{}_{s}$$

Updated Value functions

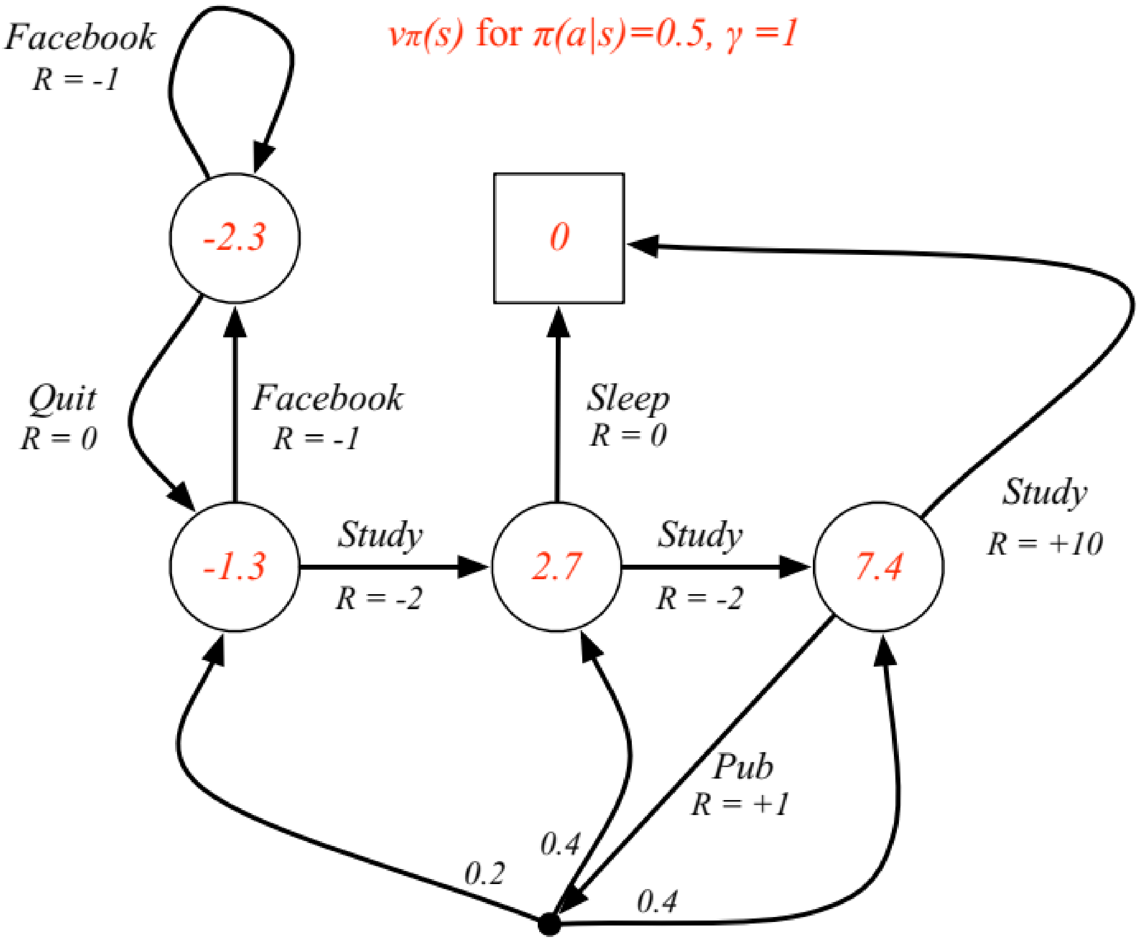
- The *state-value function* $v_\pi(s)$ of an MDP is the expected return starting from state s, and then following policy π

$$\boldsymbol{v_\pi(s) = \mathrm{E}_\pi [G_t \mid S_t = s]}$$

- The *action-value function* $q_\pi(s, a)$ is the expected return starting from state s, taking action a, and then following policy π

$$\boldsymbol{q_\pi(s, a) = \mathrm{E}_\pi [G_t \mid S_t = s, A_t = a]}$$

Student MDP: Value function



Optimal Value Function

- The *optimal state-value function* $v_*(s)$ is the maximum value function over all policies

$$\boldsymbol{v_*(s) = \max_\pi (v_\pi(s))}$$

- The *optimal action-value function* $q*(s; a)$ is the maximum action-value function over all policies

$$\boldsymbol{q_*(s, a) = \max_\pi (q_\pi(s, a))}$$

- The optimal value function specifies the best possible performance in the MDP.
- An MDP is "solved" when we know the optimal value fn.

Optimal Policy

- Define a partial ordering over policies:

$$\pi \geq \pi' \text{ if } v_{\pi}(s) \geq v_{\pi'}(s) \;\; \forall s$$

Theorem: *For any Markov Decision Process*

- *There exists an optimal policy π_* that is better than or equal to all other policies:* $\pi_* \geq \pi \;\; \forall \pi$
- *All optimal policies achieve the optimal value function,*

$$v_{\pi*}(s) = v_*(s)$$

- *All optimal policies achieve the optimal action-value function*

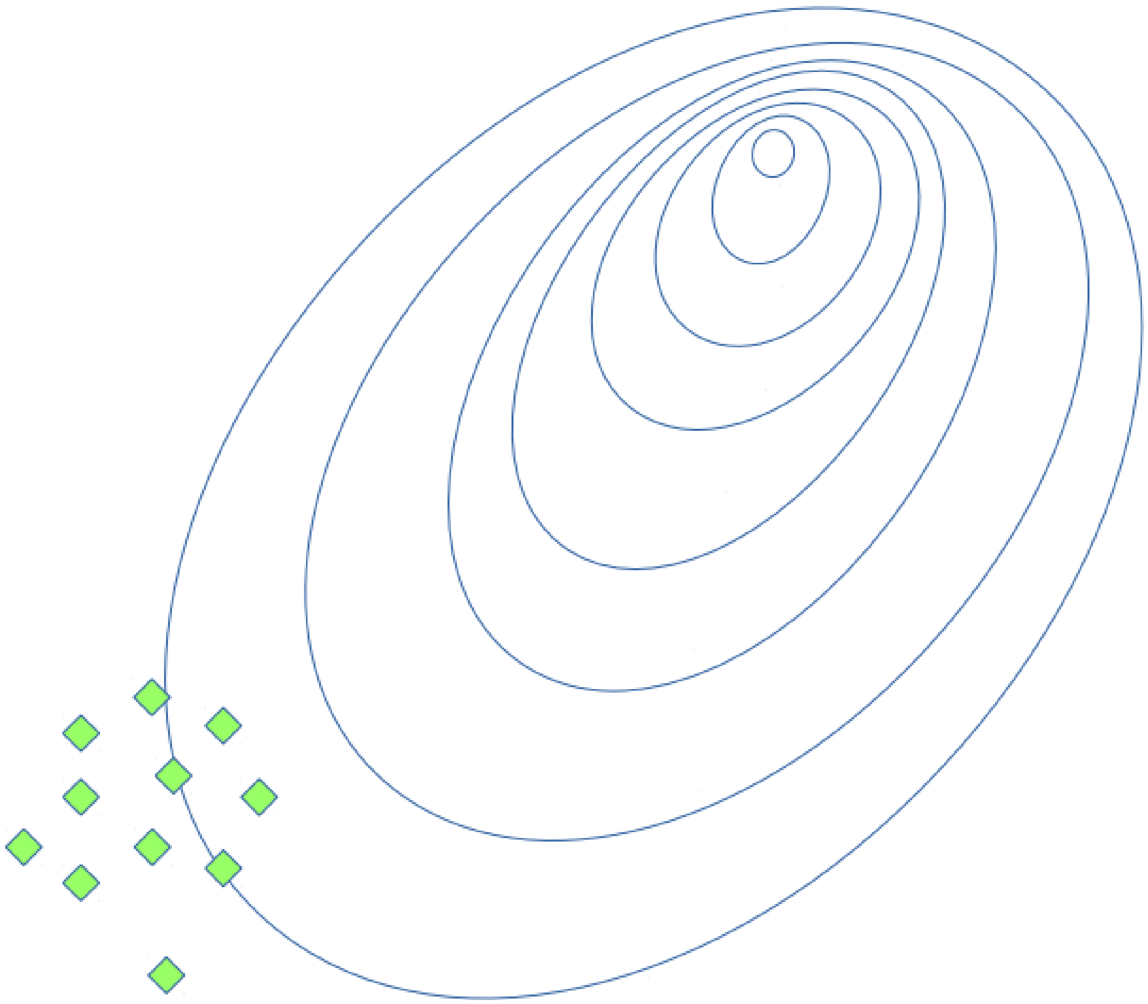
$$q_{\pi*}(s, a) = q_*(s, a)$$

No more theory – let’s do real things!

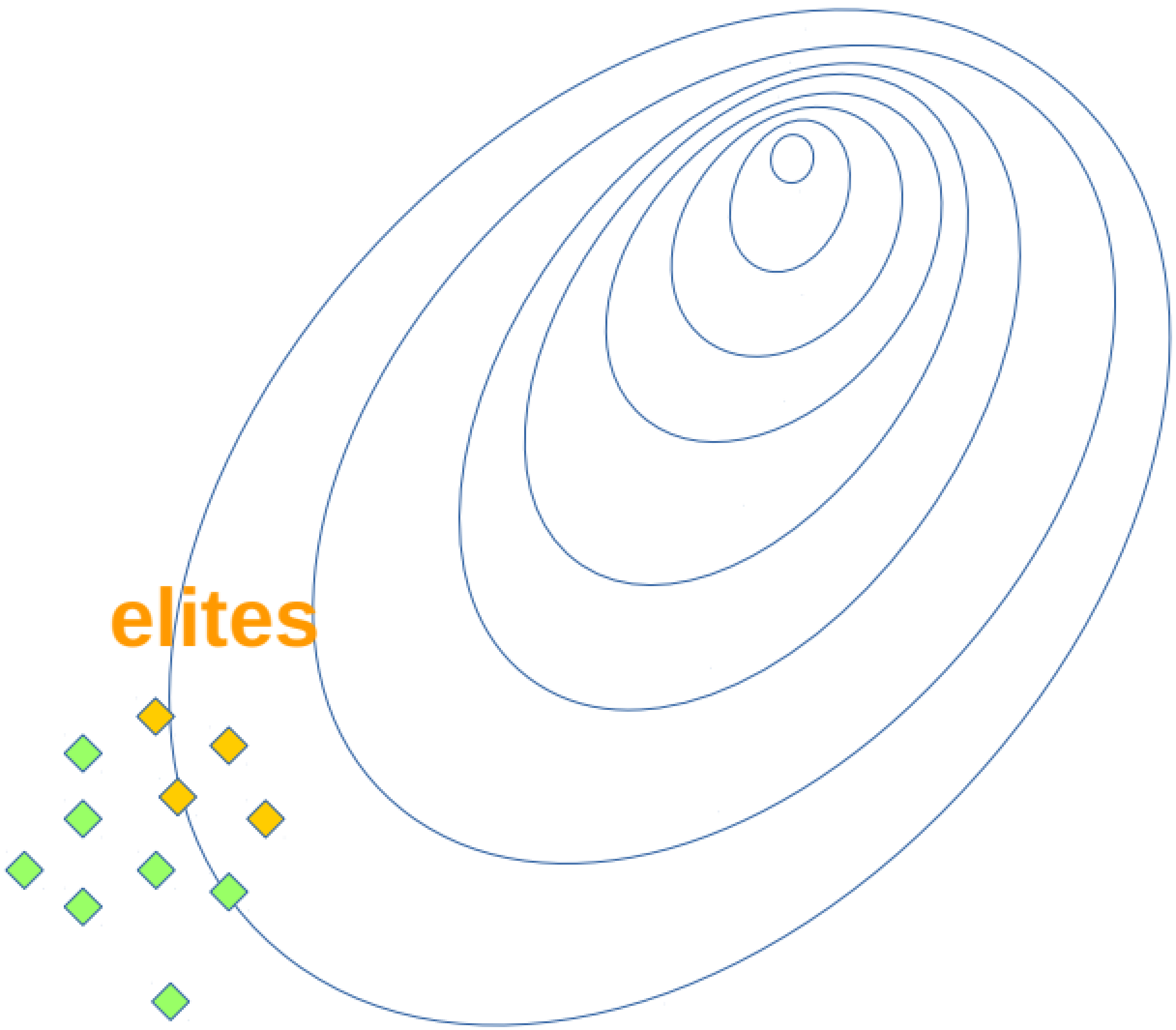
Cross Entropy method

- Let's derive some heuristic:
  - Initialize policy (random!)
  - Repeat:
    - Sample N episodes
    - Pick best 30 % best episodes – a.k.a. "elite" episodes
    - Change policy to choose actions from elite episodes

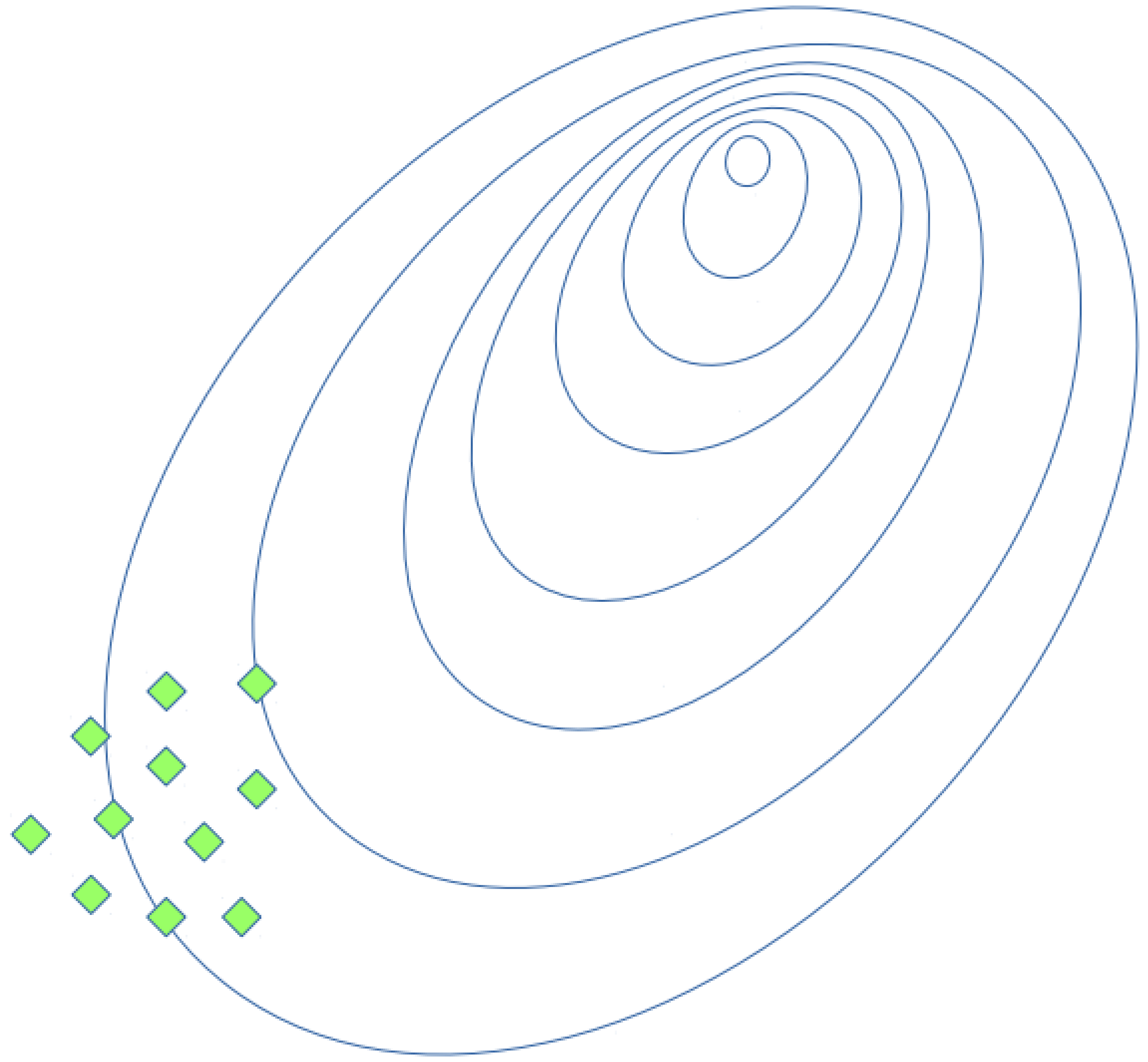
Cross Entropy method: step-by-step



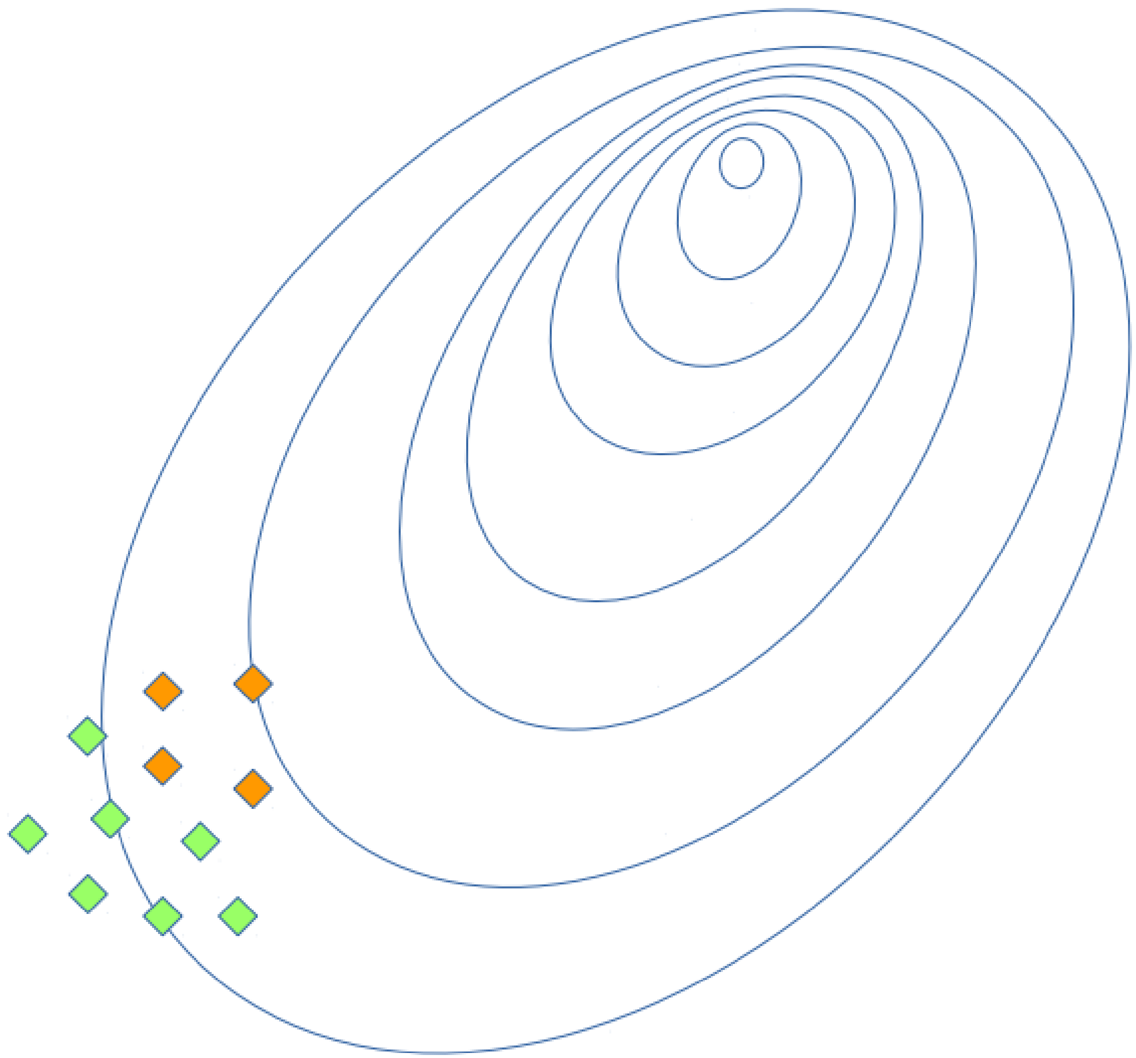
Cross Entropy method: step-by-step



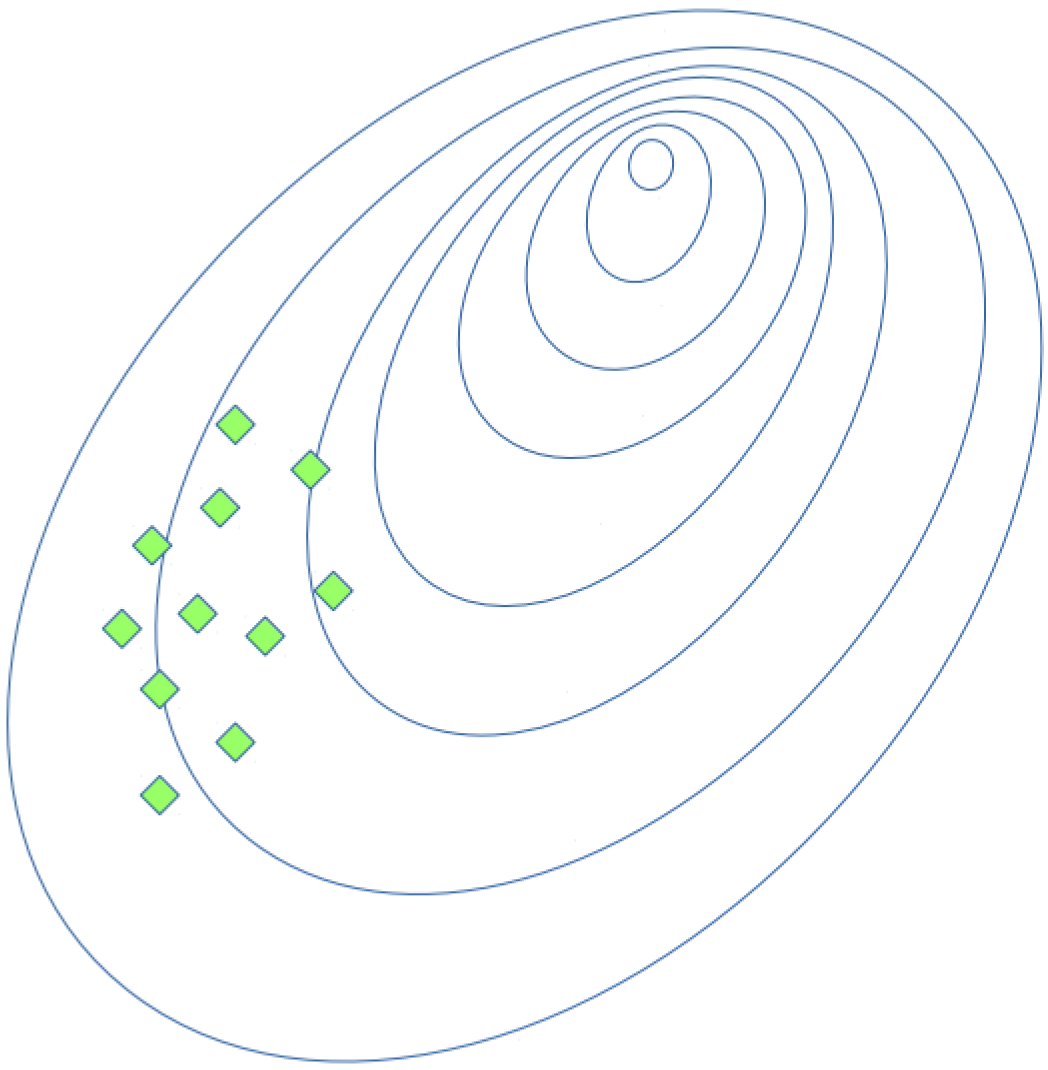
Cross Entropy method: step-by-step



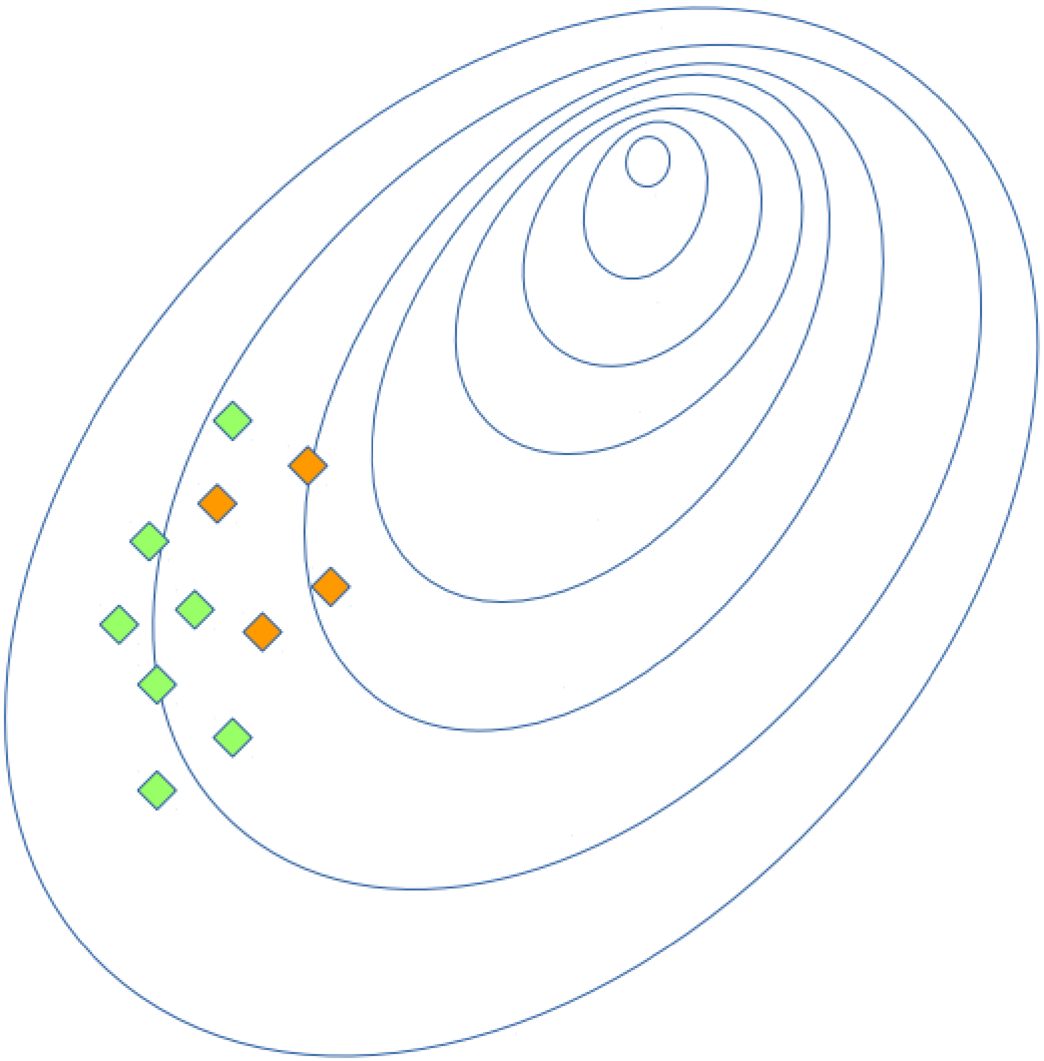
Cross Entropy method: step-by-step



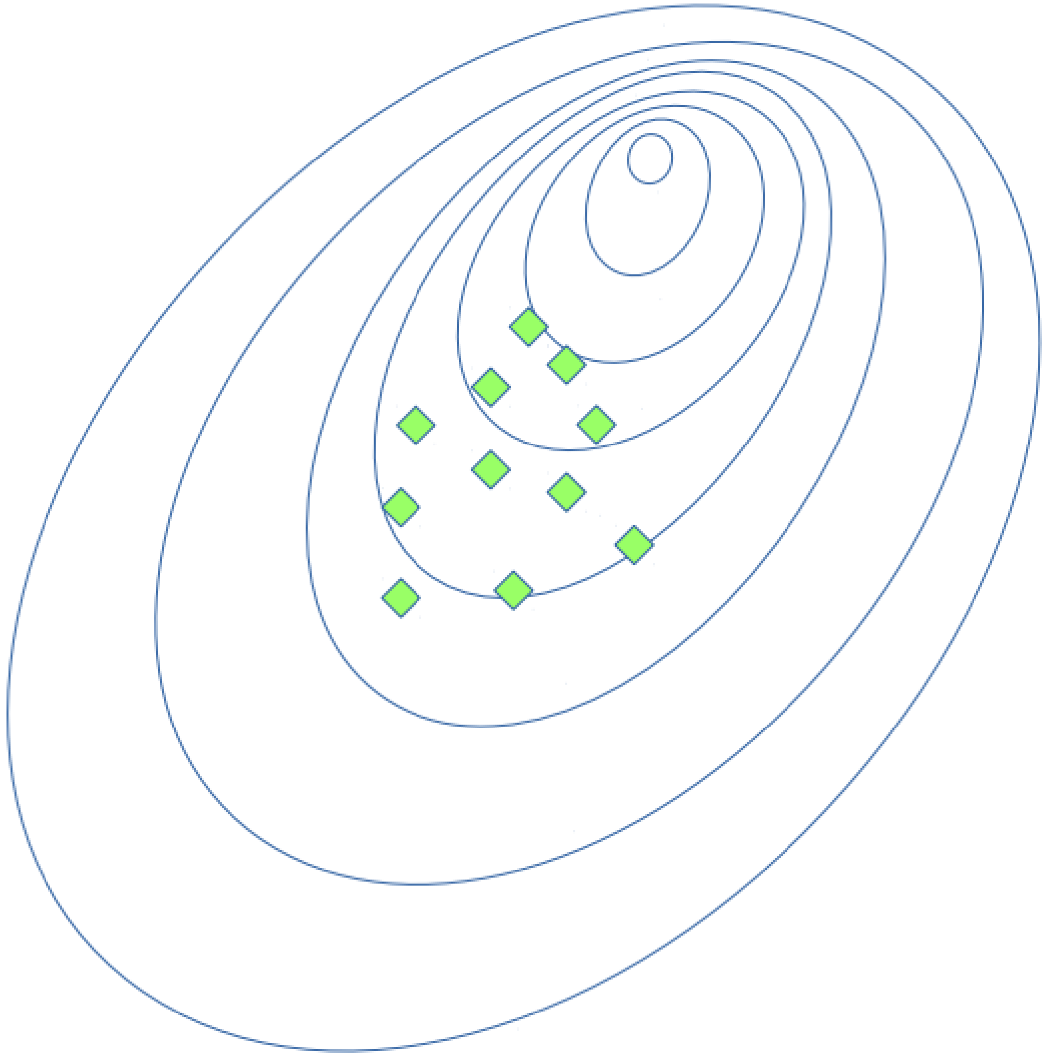
Cross Entropy method: step-by-step



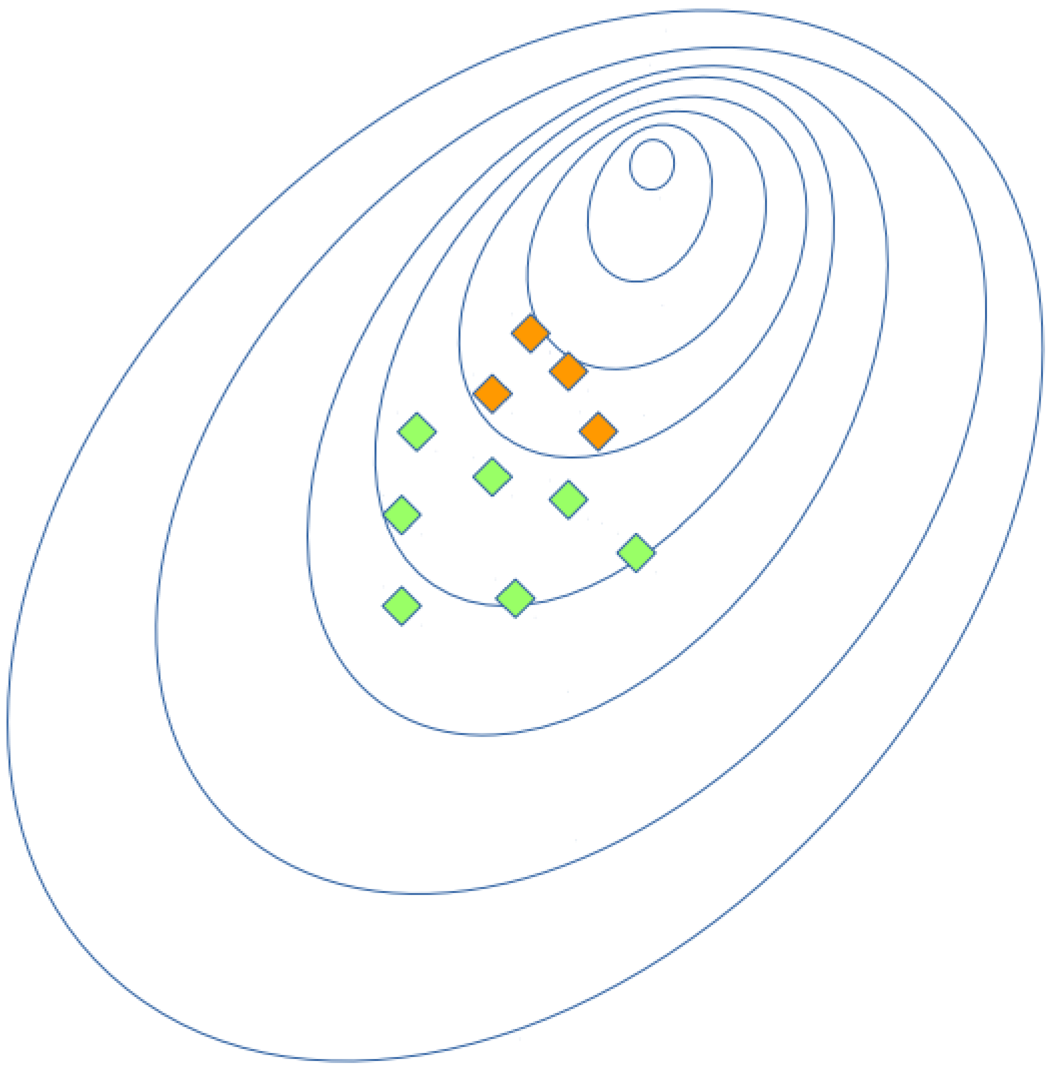
Cross Entropy method: step-by-step



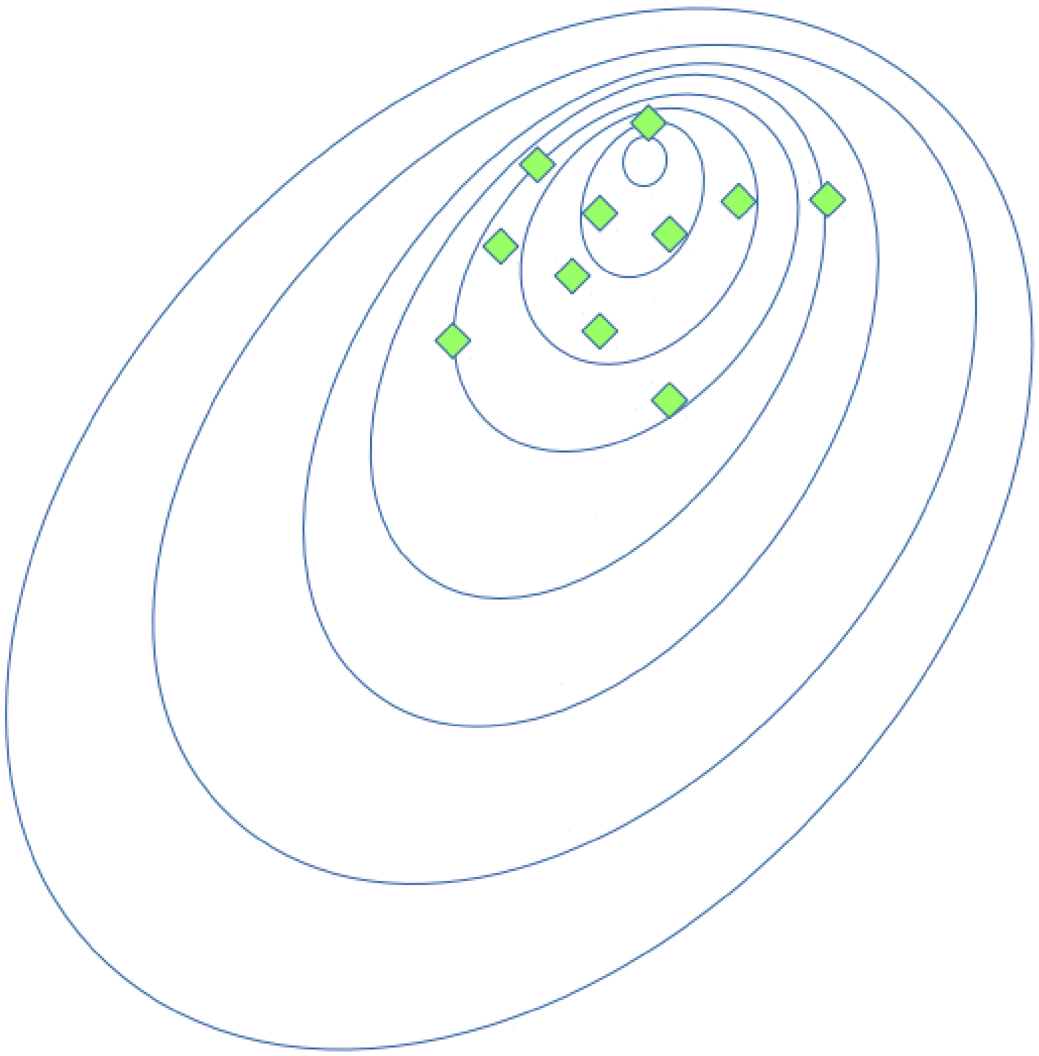
Cross Entropy method: step-by-step



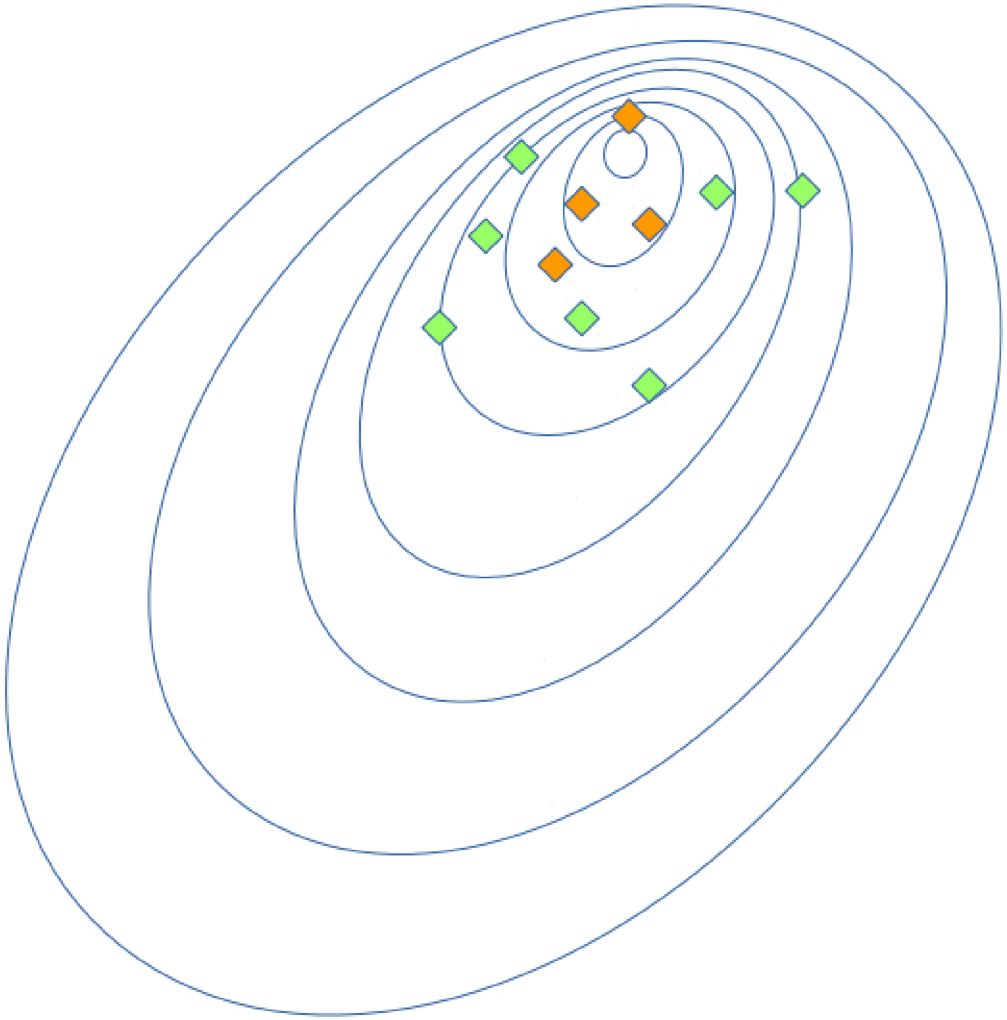
Cross Entropy method: step-by-step



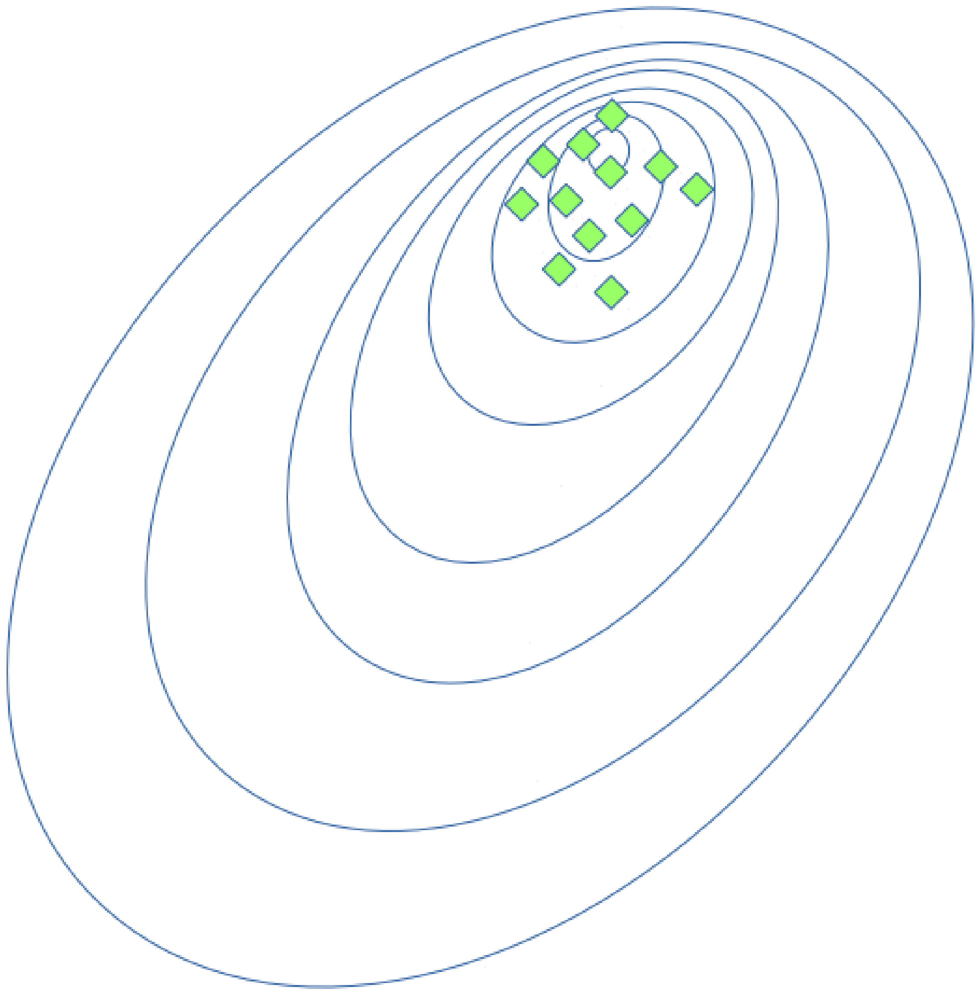
Cross Entropy method: step-by-step



Cross Entropy method: step-by-step



Cross Entropy method: step-by-step



Tabular Cross Entropy method

- Policy is matrix A
  - $\pi(a \mid s) = P[A_t = a \mid S_t = s] = A_{s,a}$

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

- Elite = $[(s_1,a_1), (s_2,a_2), \ldots, (s_k,a_k)]$
- Update policy:

$$\pi(a \mid s) = \frac{took\ a\ at\ s\ state}{was\ at\ s\ state} = \frac{\sum[s_t = s][a_t = a]}{\sum[s_t = s]}$$

Cross Entropy problems

But what if your environment has infinite/large state space ?

Approximated Cross Entropy method

- Policy is approximated

  - $\pi(a \mid s)$ predicted by Neural Network, Random Forest or any other ML algorithm.

- You can't set $\pi(a \mid s)$ explicitly, it's not matrix anymore.

- Training data for our model:

$$\text{Elite} = [(s_1, a_1), (s_2, a_2), \ldots, (s_k, a_k)]$$

Questions?