Principles of Online Decision-Making

CS-303

Prof. Matthias Grossglauser

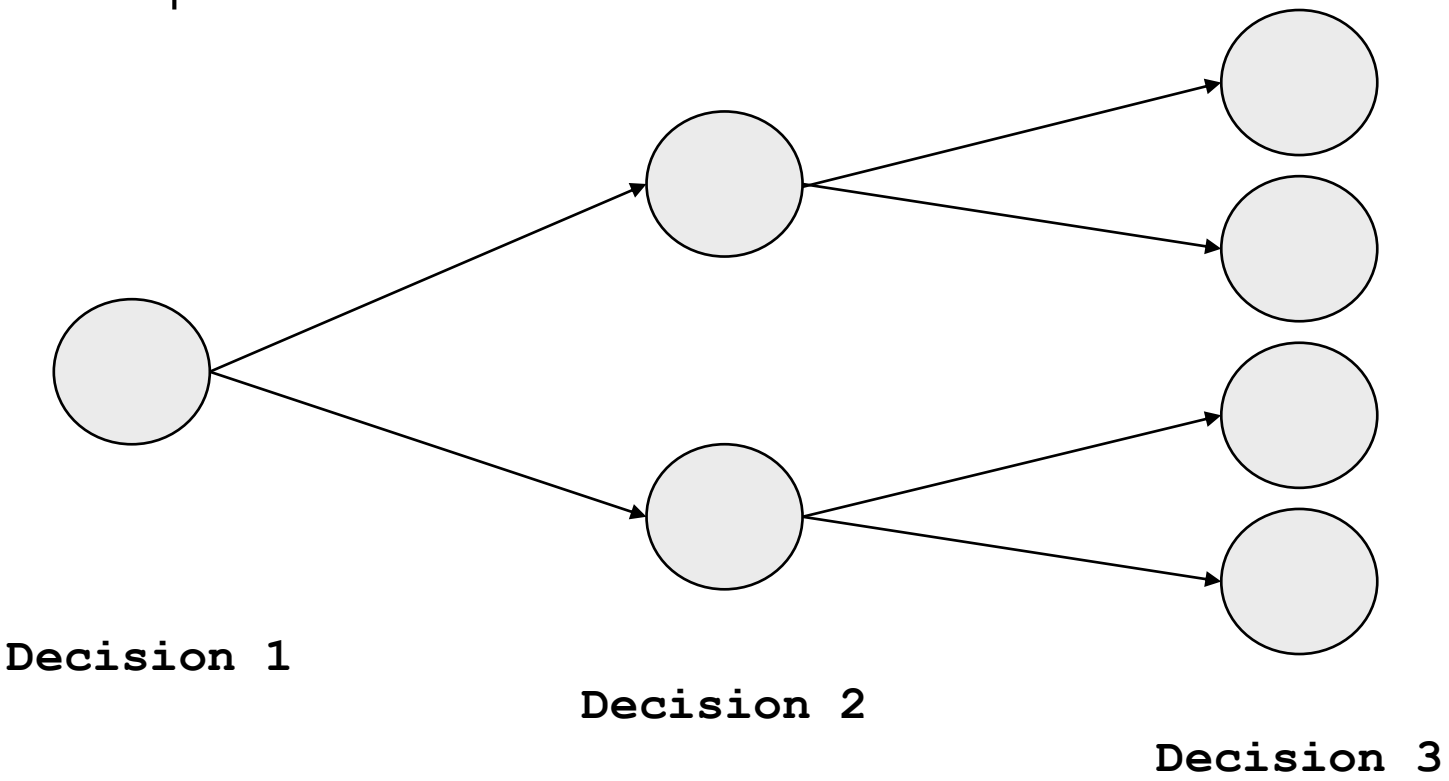
Information and Network Dynamics (INDY) lab
School of Computer and Communication Sciences (I&C)
EPFL

Organization

- Class notes
  - Slides published on moodle before class
  - Textbooks: starting next week
- Instructors:
  - Matthias Grossglauser (INF 015)
- Teaching Assistants:
  - Daichi Kuroda
  - Oscar Villemaud
- Grading:
  - Midterm exam: 30%
  - Final exam: 70%
- Homeworks:
  - Mostly pencil-and-paper, with some experiments/simulations, esp. later in class
  - Not graded – solutions given along with next problem set
  - Strongly encouraged to try to solve problems before consulting solutions

What is online (sequential) decision-making?

- Key ingredients:
  - Sequential: agent decides on an action, gets feedback, decides on another action,…
  - Usually: sequence = time
  - Decision: meaning “irrevocable”
  - Uncertainty: environment not known → probe; future not known → statistical assumptions



Example: teaching a new class

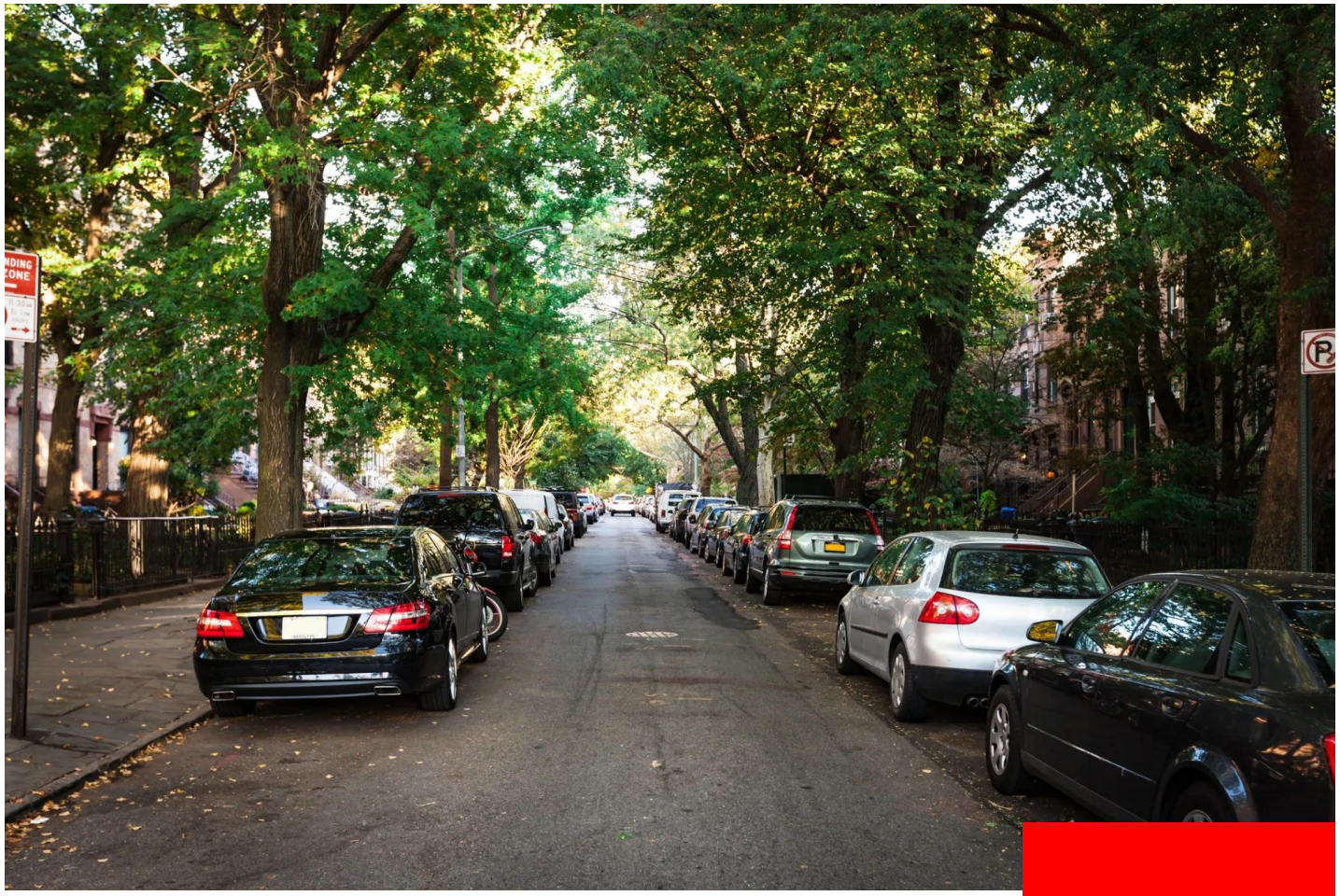
- A class is taught the first time
  - Some prior information on student background, curriculum, goals
- Finite horizon: 14 weeks of lectures
- Uncertainties:
  - Student background and expectations
  - How does content get acquired? Presentation, homeworks,…
  - Extraneous effects (other lectures, weather,…)
- Each lecture will provide information on these:
  - Questions during lecture and homework sessions
  - Midterm exam
- Agents take actions:
  - Lectures get tweaked
  - Students drop class, others sign up

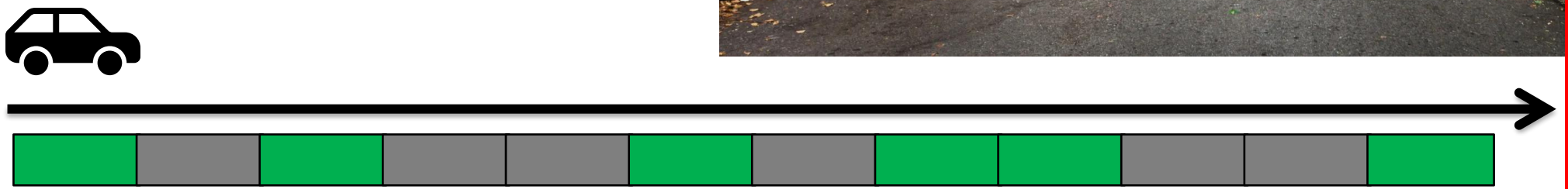
Three classes of models

- Optimal stopping: the secretary problem (this week)
  - Finite horizon and need to commit
- Bandits: learning the value of options (arms) over time (about 6 weeks)
  - Repeated decision on which arm (option) to play
  - Outcomes are random
  - There is an optimal arm, we just do not know it a priori
- Reinforcement learning: learning to interact with an environment over time (about 6 weeks)
  - Environment has “state”, i.e., memory
  - Outcomes of actions are random
  - Reward has delay

Example: optimal stopping

- Finding a parking spot close to a destination
  - Limited lookahead
  - If you reach the destination without having found a spot → FAIL
  - If you find a spot, the reward is = - distance to the destination
- Tradeoff:
  - Park too early: a lot of walking
  - Park too late: miss the show
  - How to do this optimally?



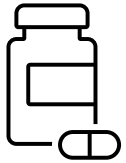


Multi-armed bandits

- Set of actions = arms
  - Sequence of decisions on actions
  - Receive a reward after each action
- Reward function
  - Stochastic bandit: reward is random
  - Independent of everything except choice of action
- History: Thompson [1933]
  - Example: running a clinical trial without harming people unnecessarily

Multi-armed bandits

- Gold standard for clinical trials: randomized
  - Randomly give each patient the drug or a placebo



- There is always a nonzero probability that medicine does not work or is more harmful than placebo
- When do we decide that this is so unlikely that we should stop giving placebos?

Example: clinical trial of a new medication

- Classical statistical approach: randomized trial, hypothesis test
  - Fix some confidence criterion beforehand (p-value) for a statistical test → size of trial n, i.e., how many patients?
  - More patients → higher confidence
- But:
  - What if the medicine works very well? (Or seems worse than placebo?)
    - In this case, many people stay sick needlessly
  - We should be able to abort early once the outcome becomes clear; if outcome is positive, start treating everybody
    - But what if it was a fluke? We'd be harming people forever…
- Question:
  - How to do this optimally?
  - Instead of random for every patient: patients come in sequentially, decide for every new patient: drug or placebo?
    - Use all available preceding information

Multi-armed bandits

- Every arm a generates a reward distributed like X_a, independent of everything
- The importance of the horizon
  - How would you play this for $n = 1$?
  - How would you play this for $n \to \infty$?
- Analogy:
  - Stocks have higher return but also higher volatility
  - Bonds have lower return but also lower volatility
  - Investment advice for a 20 year old vs 50 year old
- The horizon over which we need to do well will be a key parameter

Probability tools: concentration results

- Concentration results: looking at limiting probability distributions at different "zoom levels"
  - Suppose we have a sequence of i.i.d. random variables X_i
  - What can we say about sum of such RVs: $S_n = \sum_{i=1}^{n} X_i$?

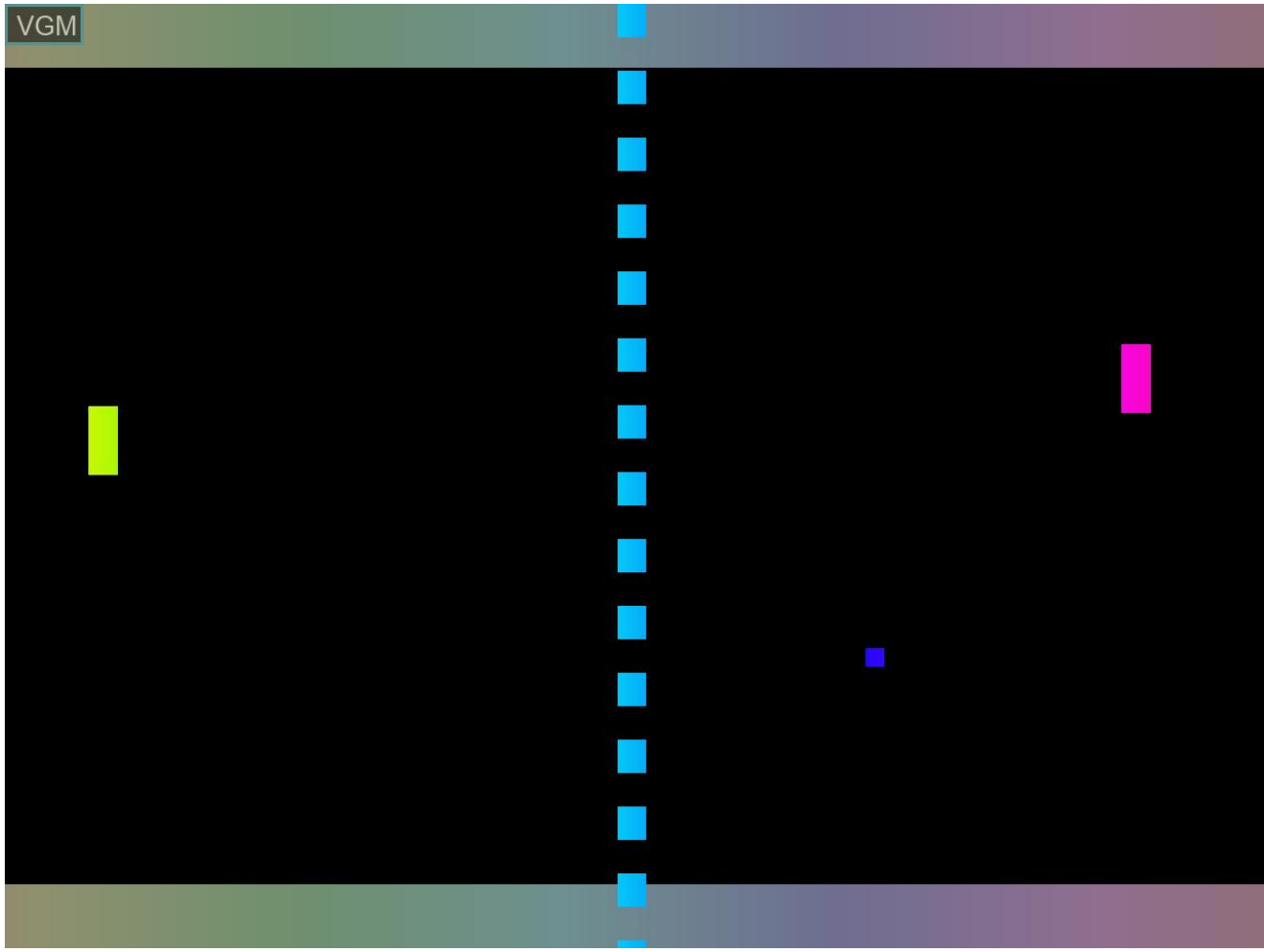
- Law of large numbers: $\frac{S_n}{n} \to \mu$
  - Average converges to mean

- Central Limit Theorem: $\frac{S_n - n\mu}{\sqrt{n}} \to N(0, \sigma^2)$
  - Characterizes "typical" (standard) deviation from mean

- Chernoff bound: $\mathbb{P}\left(\frac{S_n}{n} \geq \mu + \epsilon\right) = \exp(-f(\epsilon)n)$
  - More precise tail behavior, i.e., probability of very large departures from mean

Reinforcement learning

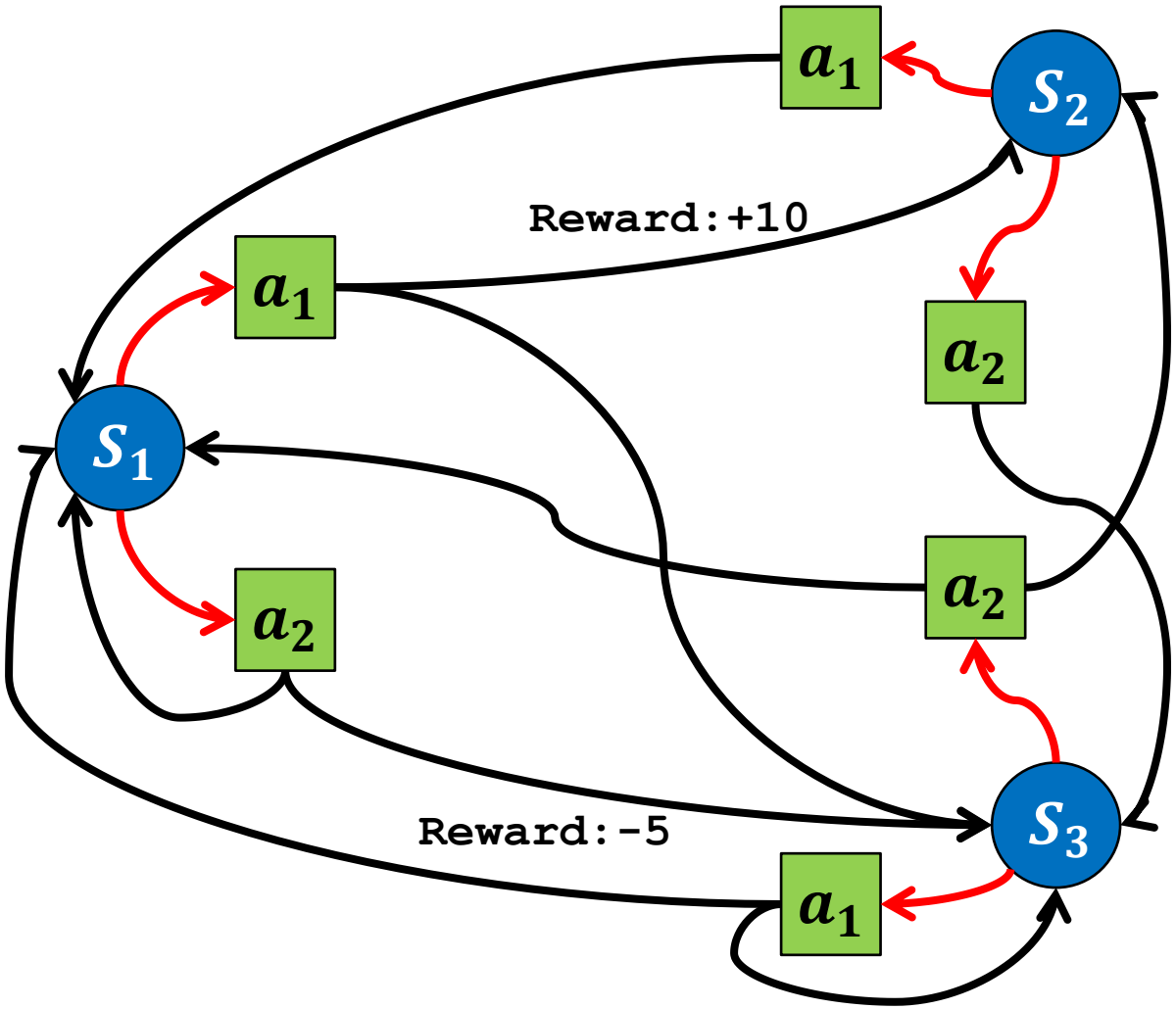
- Example: learn to play a computer game (ATARI Pong)
  - We do not (or cannot) program the rules of the game and then find the optimal move (UP or DOWN)
  - Instead, an algorithm should observe and master the game as quickly as possible by playing and observing
- Feedback: reward signals
  - +1: I score
  - -1: adversary scores
- State: the environment has a state
  - Here: ball (pos, velocity vector), own pos, (adversary pos?)
  - Right action depends on the state
- Key difficulty: signals arrive with delay!



Probability tools: Markov chains

- You know sequences of i.i.d. random variables well: $X_1, \ldots, X_n$
  - Law of large numbers, central limit theorem,…
- How to add "a bit of dependence"?
  - (X_1, X_2, X_3), and assume that $X_1 \perp X_3 | X_2$ → Markov chain
- Now assume number of variables $\to \infty$
  - Assume homogeneity: probabilistic relationship between neighboring variables is same
  - Assume discrete RVs
  - Transition matrix P: $P_{ij} = \mathbb{P}(X_n = j | X_{n-1} = i)$
- Classification of states:
  - Depending on P, sets of states can have different long-term behavior (absorbing, transient, periodic)
- Stationary distribution: existence, how to compute
  - Under certain conditions on P, the long-term behavior of the Markov chain converges (ergodicity)
  - Ergodic regime, often of interest; computing the stationary distribution π, with π the (normalized) solution of eigenvector problem $\pi = \pi P$

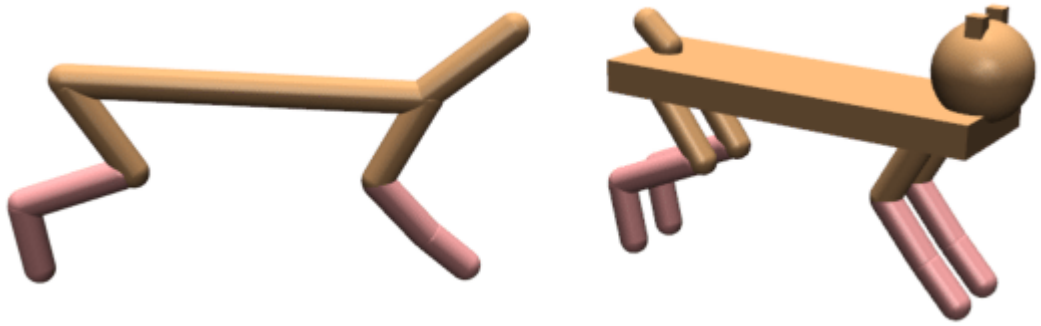
Probability tools: Markov Decision Processes (MDP)



- Policy π:
  - Maps a **state** to an action
- Goal: max reward over a long time horizon

Example: learning to walk

- Model: physics of limbs, gravitation, friction with ground
- Actions: move joints
- Rewards: getting to some place



[medium.com]

- Key challenges: large state space, continuous state space and actions
  - We will not touch on this much, topic for future classes

Reinforcement learning vs supervised learning

- Classical supervised learning:
  - Collect some labeled data
  - Training: find a model giving good predictions
  - Test time: deploy the model and get the benefits
- Real world:
  - Data accruing during test time → refine the model
  - The system might not be stationary → adapt the model
  - Retrain the model periodically (for example)
- Updating the model may influence the data distribution
  - Example: recommender system → people consume items recommended by system → future data is biased toward recommended items → positive feedback loop?
- Reinforcement learning integrates “training” (explore) with “testing”, i.e., using the model to accomplish something (exploit)

Key concept: explore-exploit tradeoffs

- Explore: collect samples to learn the model
  - A bit like the training set in supervised learning
  - We do not attempt to take good decisions, but to understand the problem
- Exploit: try to take good actions/decisions
  - A bit like the test set in supervised learning
  - We do try to take good decisions, even though the model is still noisy
- Parking: explore (look for better spot), exploit (park)
- Clinical trial: explore (give a random med), exploit (give the best treatment)
- Pong: explore (do a random move), exploit (guess the best move)
- This class:
  - Probabilistic models for such situations
  - Algorithms
  - Analysis of the optimality of these algorithms

Tools and prerequisites

- This is essentially a probability class
  - Introductory probability is enough
  - …but: a certain flair is a plus
- Not a bad idea to take Modeles stochastiques in parallel
  - Small overlap: mostly Markov chains
  - But PODM is self-contained
  - Solid prerequisites in basic probability required

Secretary problem, aka “apartment hunting problem”

- You are looking for an apartment to rent; the market is hot → take it or leave it forever
- Known number of options: n apartments can be viewed, need shelter by then
- Every apartment has a quality level: $x_i \sim$ iid from some **unknown** law
  - Alternatively, simply assume that you view apartments in a random order
- No information about the probability law
  - The only information we have is the x_i seen so far
- Irrevocable decision: sign the lease or gone forever
- Goal: maximize probability of selecting the overall best apartment

What does this model capture?

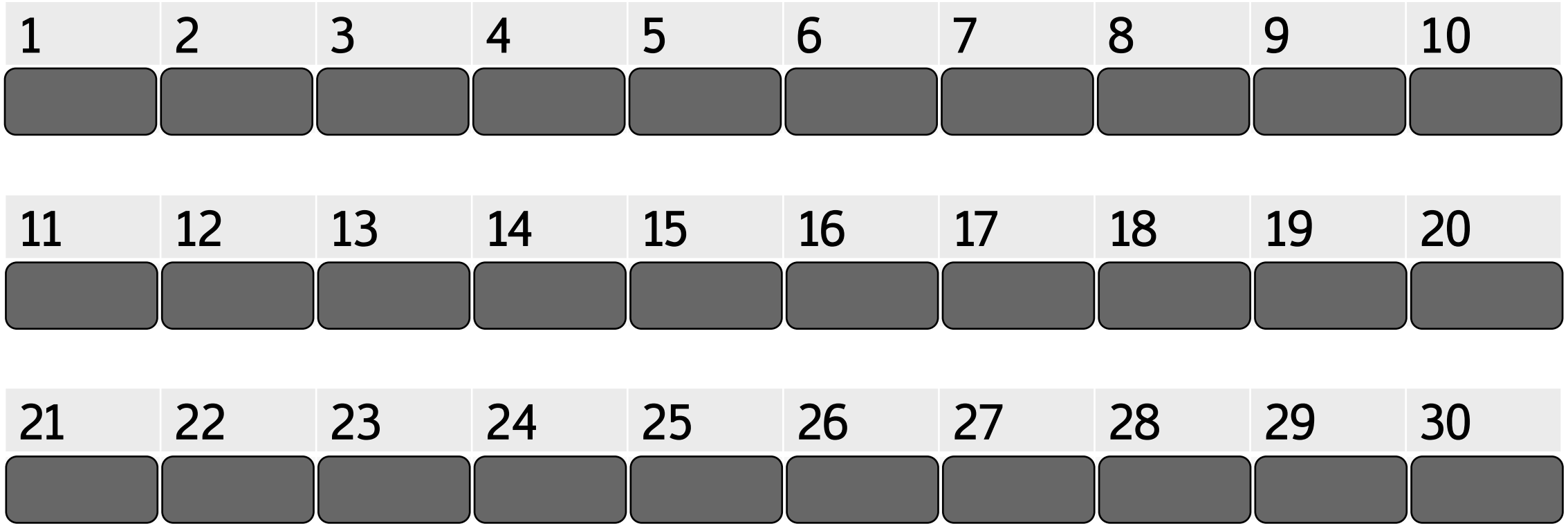
- Many practical applications have a flavor of this problem:
  - Resource allocation: how much effort to search and evaluate before hiring a candidate/giving a contract to a vendor,…)
  - Online auction: seeing other bids to update own bids
  - Acquiring friendships and social contacts over a lifetime?
  - Can you imagine others?
- Other names:
  - Marriage problem
  - Sultan's dowry
  - Googol game

Let's try it...

- I will read random 30 numbers to you
  - No information about the range/distribution
- You try to guess the winner – you need to commit before I read the next number
- Think of how you want to play the game? What is your policy?
- ...
- When you decide that you want to rent the apartment, write down the score and raise your hand

Let's go apartment hunting...

- Scores and best-so-far



Secretary problem: optimal policy

- Claim: the optimal policy is necessarily of the following form:
  - (1) If x_i is not a new maximum, continue
  - (2) If it is, then STOP (commit) if $i > k$ (where k is a constant), otherwise continue
- Why (1)?
  - Because otherwise, guaranteed failure → never good
- Why (2)?
  - We claim that the optimal policy continues before k, and stops after k if new maximum
  - In other words, the policy is monotonic: if we decide to stop (for a new max-so-far) at i, we must also stop (for a new max) at any $j > i$
  - Note: conditional on seeing the max over $x_1, \dots, x_i$ so far (i.e., $x_i > max(x_1, \dots, x_{i-1})$), what is the probability of success?
    This is simply i/n: The overall max is distributed uniformly over $[n]$ → the conditional probability of success is increasing
  - But the probability of seeing a new max-so-far is decreasing with i
  - Hence: if the tradeoff is favorable at i, it is even more favorable at $j > i$

Secretary problem for $n = 1, 2, 3$

- $n = 1$:
  - Nothing to do, commit to only option
- $n = 2$:
  - Best is first with 50%, and second with 50% → either commit to first; if not, implicitly commit to second=last; $\mathbb{P}(success) = \frac{1}{2}$
- $n = 3$:
  - Smallest n where we actually "make a decision"
  - $k = 2$: we use the comparison of first and second position to decide whether to take 2nd or 3rd

$k = 1$	$k = 2$	$k = 3$
123	123	123
132	132	132
213	213	213
231	231	231
312	312	312
321	321	321

$\mathbb{P}(E) = \ldots$ $\mathbf{\frac{1}{3}}$ $\mathbf{\frac{1}{2}}$ $\mathbf{\frac{1}{3}}$

Secretary problem: optimal policy (first new max after k)

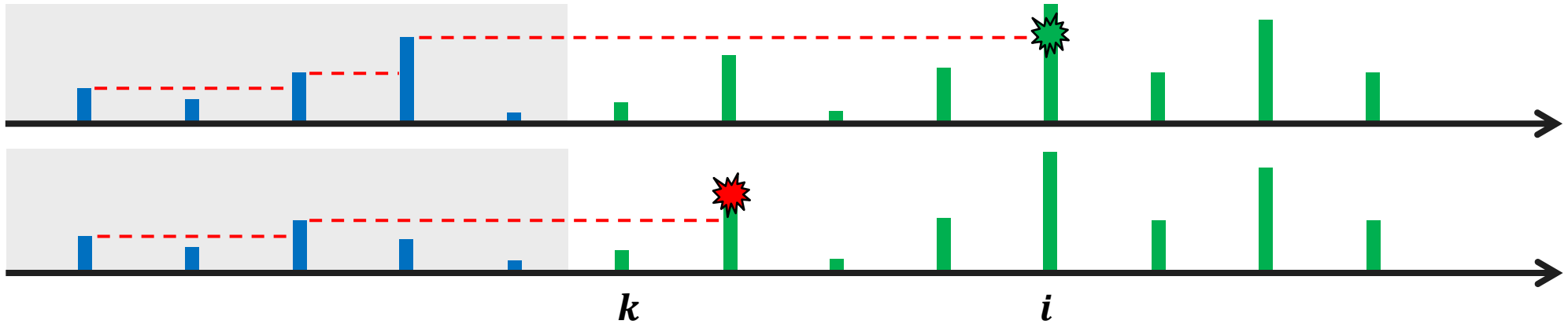
- What should k be?
- Call U index of the true max; note: $U \sim unif(1, n)$
- Call Z the output of the policy (its guess of U)
- Call the event E=success, i.e., policy chooses i and i is best

$$\mathbb{P}(E) = \sum_{i=k}^{n} \mathbb{P}(Z = i \cap U = i) = \sum_{i=k}^{n} \mathbb{P}(Z = i | U = i)\mathbb{P}(U = i)$$

- $\mathbb{P}(U = i) = 1/n$:
  - Unconditionally, U is clearly uniform

Optimal policy

- $\mathbb{P}(Z = i|U = i)$:
  - When is $\{(Z = i|U = i)\}$ true?
    - i is the overall best → triggering (2) is certain at step i → success
    - But we need to make sure to make it here, i.e., policy must not stop earlier → how to guarantee this?
    - Observe that if and only if all previous best-so-far are before k, we make it to i



    - Conditional on $\{U = i\}$, where is the **best before i**?
      Note: not necessarily second-best overall (which could be after i), just best so far!
      All we know it's not at i → conditional position is $\sim \text{unif}(i-1)$
    - Hence $\mathbb{P}(Z = i|U = i) = \mathbb{P}(\text{unif}(i-1) \in [k-1]) = (k-1)/(i-1)$

Optimal policy

- Finite n:

$$\mathbb{P}(E) = \sum_{i=k}^{n} \mathbb{P}(Z = i | U = i)\mathbb{P}(U = i)$$

$$= \frac{1}{n}\sum_{i=k}^{n} \frac{k-1}{i-1} = \frac{k-1}{n}\sum_{i=k}^{n} \frac{1}{i-1} = \frac{k-1}{n}(H_{n-1} - H_{k-2})$$

- Optimal k as a function of n:

n	1	2	3	4	5	6	7	8	9	10	20	30	40	50	100	1000
k	1	1	2	2	3	3	3	4	4	4	7	11	15	18	37	369
$\mathbb{P}(E)$	1	0.5	0.5	0.458	0.433	0.428	0.414	0.410	0.406	0.399	0.384	0.379	0.376	0.374	0.371	0.368

- Note: success probability seems to converge to something positive!
- Note: k and $\mathbb{P}(E)$ do not depend on the distribution of x_i, only on n!

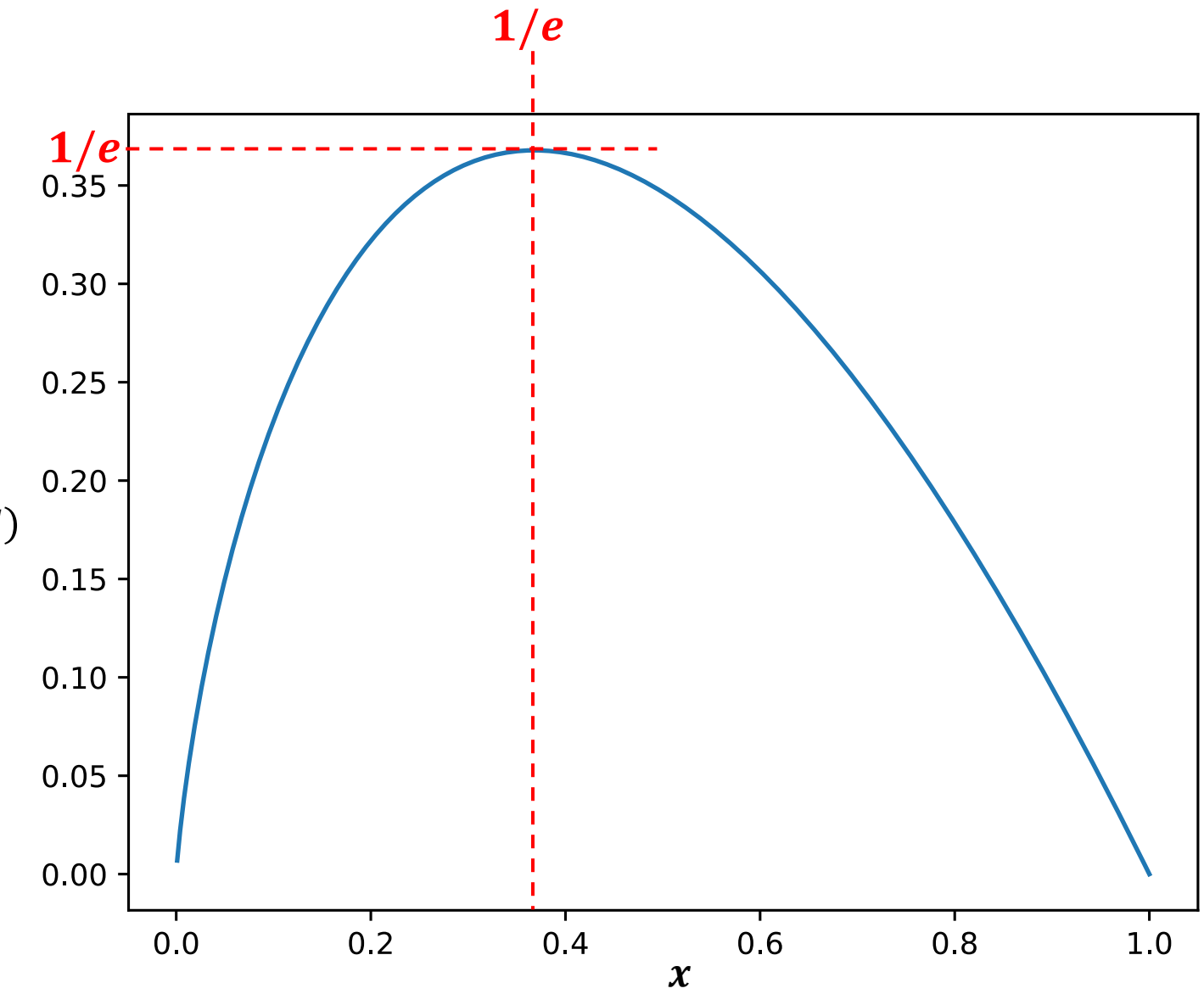
Optimal policy

- Large n:
  - Set $x = \frac{k}{n}$

$$\mathbb{P}(E) \xrightarrow{n\to\infty} x \int_{y=x}^{1} \frac{1}{y}\,dy = -x \ln x$$

- Optimal x:
  - Derivative $= \frac{d\mathbb{P}(E)}{dx} = 0$

  → $x = e^{-1} \approx 37\%$
- "Look at 37%, then leap!"



Let's go apartment hunting...

- Scores and best-so-far

1	2	3	4	5	6	7	8	9	10
73	89	35	14	2	4	32	31	149	106

11	12	13	14	15	16	17	18	19	20
42	155	20	10	35	32	70	115	41	38

21	22	23	24	25	26	27	28	29	30
7	2	82	8	36	47	57	38	8	143

Random order vs i.i.d. random variables

- Important assumption: no information about future values
  - I.i.d. random variables: assume player does not know anything about the distribution
    - Counterexample: suppose we know the distribution → we can compute the probability that x_i is overall max directly, if that probability is large no need to wait to k
  - Rank order: only partial order over items seen so far (up to i)
    - Counterexample: suppose x_i is the total rank order over $[n]$→ just wait for top item at any time and STOP
- The need to "pay" with some samples to discover the distribution and to judge future samples is baked into this assumption

The goal matters: max $\mathbb{P}(\mathbf{Z} = \mathbf{U})$ vs max $\mathbb{E}(\mathbf{x}_{\mathbf{Z}})$

- Optimal policy specific for the particular criterion we adopted
  - Ensures that we should never accept anything below max-so-far
- Suppose we wanted instead to maximize the expected value of the choice (i.e., rent a good apartment, no disaster if not the best) → does optimal policy change?
  - Yes it does! Counterexample: suppose we are close to n, and get second-best so far → very unlikely that this can still be beaten, so take it
  - In fact, we should accept anything above the median so far!
  - This guarantees failure under the previous policy, but does pretty well in terms of expected score

Explore and exploit tradeoff

- The key underlying concept of this class
- **Explore**: make potentially bad decisions (in terms of some specified goal) to estimate parameters of the system, and to help make better decisions in the future
- **Exploit**: based on these estimates, make good decisions with respect to the specified goal
- Secretary problem: sharp separation, two phases, a-priori threshold when to switch
- In bandit problems and in reinforcement learning, these will meld more gradually
  - Bandits: gradually improve estimates for rewards of different arms – stop playing some earlier than others, gradually optimize rewards
  - Reinforcement learning: gradually exclude states that are unlikely/costly to lead to rewards, and drive system towards good states
- We start on bandits next week

Reading assignment

- None this week
- Homework: starting next week
  - Familiarize with bandit simulation framework