Lecture 1: What is Reinforcement Learning

Learning objectives

1. Who should take the course? How will this course be run?
2. What is the RL problem? When applications might be amenable to RL?

Introduce course staff. Will do rest of logistics at end of lecture.

Dolphin trainer. Q: what makes this hard? What is a good strategy? Q: How would you right down the optimal strategy for this game?

1 What is reinforcement learning?

Definition 1: Reinforcement learning (RL) is a mathematical definition for optimal decision making.

Definition 2: Reinforcement learning (algorithms) are a set of techniques for solving sequencial decision making problems.

Examples of RL problems.

Defining the Markov Decision Process (informal) We have states/observations s_t, actions a_t, rewards $r(s_t, a_t)$. Draw the MDP loop. Our goal is to maximize the sum of these rewards:

$$\sum_t r(s_t, a_t). \tag{1}$$

Note that we're maximizign rewards both now and in the future. Would want to take actions now that put us in a place to get high rewards in the future. We'll make this definition more precise on Thursday, but this will be enough to get started.

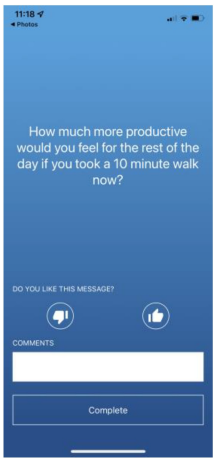

Figure 1: Mobile apps as an RL application.

- **Chess**: actions are moves, observations are board position, reward is win/loose.
- **Roomba**: actions are the velocity of each wheel, observation are bump sensors, reward is whether the floor is clean.
- **Dolphin**: actions are movements, observations are visual inputs, rewards is whistle.
- **Squirrel**: actions are muscle contractions, observations are touch/sight/smell/etc, reward is warmth/satiated/safe/etc.
- **Nuclear fusion**: actions are magnet strength, observations are shape/characteristics of plasma, reward is energy production (and penalty for breaking the reactor).
- **Medical app**: actions are push notifications (text), observations are movement (tracked with phone), rewards are steps (also tracked with phone).
- **Social media**: actions are posts to display, observations are videos watched/liked/commented, rewards are minutes spent on platform.

1.1 Is reinforcement learning just supervised learning?

Supervised learning. Example: mapping images to cat/dog label.

- Inputs: data $\mathcal{D} = \{(x, y)\}$. Note that the labels are provided.
- Output: function $f(\cdot)$.
- Objective: learn a function $f : x \mapsto y$ such that $f(x) \approx y$.

Reinforcement learning.

- Inputs: state/observation s_t. Note that you'll get a sequence of these.
- Outputs: actions a_t. Again, note that you have a sequence of these.
- Objective: learn a function $\pi : a \mapsto s$ such that $\sum_t r(s_t, a_t)$ is large.

Key differences between RL and Supervised learning.

- In RL, we don't have labels for the correct actions.
- In RL the feedback can be delayed – we might only learn whether the first action was good after taking many subsequent actions. E.g., studying today may improve your grade on next month's midterm.

What makes RL hard?

- long horizon reasoning. Curse of dimensionality.
- learning new behaviors that haven't been done before.
- exploration, specifying rewards/tasks/costs/constraints/preferences

1.2 Is reinforcement learning just search?

One way to think about the reinforcement learning is as searching through a graph: each node represents a state, and each outgoing edge represents an actions. Each edge has a certain reward associated with it, and you want to figure out which actions to take so that you visit the high-reward edges.

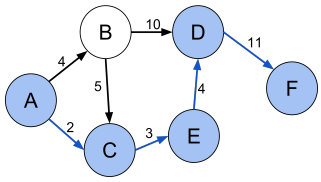

Figure 2: Reinforcement learning seen as a graph search problem.

Q: How would you define the shortest path problem?

This connection with search highlights some the salient attributes of the RL problem. For one, we aren't given the solution (like in supervised learning), but rather have to discover it from scratch. Second, if you think about this problem for a bit, you might realize that you can *cache* or *memoize* the solutions to some of the subproblems. In an algorithms course, you might have learned about dynamic programming methods that will do just this. It turns out that this will be a key component in developing tractable RL methods as well.

However, the connection with search will only take us so far. The key challenge with treating RL problems as search problems is that it assumes that the number of states and actions is small. However, for practical problems, the number of states grows quite quickly. For example, while there are 6 states in this graph, there are 10^{120} states in the game of chess (The so-called Shannon number). For these sorts of practical problems, we're going to need a way of finding patterns. In the same way that an image classifier finds that most cats have whiskers, we'll need to learn an RL agent that finds patterns that help it determine the right action, so that it can *generalize* and play good actions in states it hasn't seen exactly before.

1.3 Is reinforcement learning just optimization?

The objectives we've written for supervised learning and RL are different: while in supervised learning it seems like we're trying to model a certain function, in RL it seems like we're trying to maximize something. It is for this reason that people sometimes think about RL as just solving a certain optimization problem. And, in a sense, they're right: we are trying to optimize rewards.

However, the tools of optimization will only get you so far. While it might be pretty easy to solve the optimization problem $\max_x -(x-1)^2$, the optimization problem we want to solve doesn't have a nice analytical solution. Worse, it's not even an optimization over a single variable, but rather over a whole sequence of actions. To make things even worse, those actions might depend on the current state, so we really need to be optimizing over an entire table that tells us which actions to take in which states. And this gets even worse when we look at practical problems, where the states and actions can be high-dimension: this table is now infinitely big.

The way that we're going to make this problem tractable is to leverage the same tools that are leveraged to make supervised learning problems tractable: machine learning. In the same way that supervised learning methods automatically find *patterns* in the data (e.g., detecting ears and whiskers), reinforcement learning methods will automatically find matter in the observations that indicate what actions to take at those states.

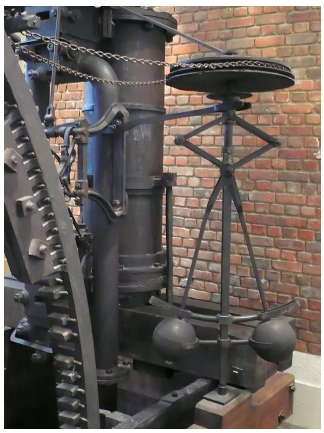

Figure 3: Maxwell's centrifugal governor is an early example of a control strategy, one directly implemented in hardware.

2 Where does RL come from?

Psychology.

- Edward Thorndike, 1898, "law of effect": Build puzzles for his cats to escape, and found that they learned to escape more quickly over time: *responses that produce a satisfying effect in a particular situation become more likely to occur again in that situation, and responses that produce a discomforting effect become less likely to occur again in that situation*.
- B. F. Skinner, 1930s, Skinner box with buttons (actions), lights/speakers (observations), and food/shocks (rewards).[1]

Neuroscience.

- RL is a common model of the brain. Dopamine is a neurotransmitter that serves as a "reward." Actions are neural signals sent to rest of body, observations are neural inputs from rest of body.[2]
- Lots of work here at Princeton on this subject (e.g., Yael Niv, Nathaniel Daw).

Control theory.

- James Clark Maxwell. In addition to laying some of the foundation of electromagnetic radiation, also build one of the first control mechanisms, the *centrifugal*

[1] Skinner, best known as a psychology, also was a science fiction writer. His "Walden Two" book describes a world where reinforcement learning (operand conditioning) is used to incentivize collective behavior and build a utopian society.

[2] Thought experiment: what if you could artificially stimulate this dopamine signal? Is this what addictive games already do?

governor. Used to regulate the control of an engine or turbine, had two balls that would spin out and apply a breaking force.

- Control theory was also central to aerospace, from the Wright Brothers to Apollo 11.
- Basically the same as the RL problem, but typically written in continuous time, so it ends up using integrals in place of summations. Often the dynamics are known; often they are assumed to be deterministic.

AI and Games

- Arthur Samuel: Checkers, 1959. Not super-human, but super-Samuel
- Gerald Tesauro: Backgammon, 1992. Slightly worse that best human player.
- Chess: Deep Blue, 1990s, eventually beat world champion Garry Kasparov.
- Atari. 2013. Beats human performance on most games.
- Go. 2016. AlphaGo beats Lee Sedol. Move 37 was unexpected – RL methods can find new solutions.

RL and Generative AI Today. When we think about GenAI today, usually think about LLMs and generative image models. These models are both trained primarily with supervised learning, but they are fine-tuned with RL. For example, LLMs are trained to mimick the internet, but that's not necessarily useful for humans. So, we give the feedback that they seek to optimize through a sequence of actions (utterances). One of the neat aspects of LLMs is that, in some scenarios, they can provide their own feedback. Later in this class we'll return to this point when discussing where rewards come from. Generative images models are often also fine tuned with RL. A human user gives a prompt, the model generates an image through a *sequence* of denoising steps (actions), and the rewards are whether the human thinks that the generated image depicts the prompt.

RL also has the potential to lift several of the key challenges with GenAI today. Finding "good" data is a key challenge in GenAI, but RL methods can automatically synthesize good data, both via exploration and by recombining pieces of old data (dynamic programming).

3 Pedagogical Strategy

RL algorithms are a sledgehammer. There's a huge variety of problems that are amenable to them. But, because they are so versatile, they can be overly complex and sample inefficient for some problems. In the language of computational complexity, while a wide variety of problems can be *reduced* to the RL problem, that doesn't mean that it'll be the best method for those problems. In other words, "easier" problems have easier solutions. We'll spend the first couple lectures talking about those easier strategies, methods which are not RL. However, the techniques that we'll discus will be useful for a wide variety of (non-RL) problems. And, this will serve to reinforce (pun intended) the definition of the RL problem.

We might think of this like swiss cheese. When given a practical problem, there are a couple of checks we should do before immediately applying an RL method. Only if those checks all pass will we resort to using RL methods.

Here's the rough roadmap:

1. Does time matter? If not, use a bandit method.
2. Do you have a model of your system? If so, consider using MPC.
3. Do you have demonstrations? If so, consider imitation learning.

Outline for the rest of the semester

1. **What is RL?** (1 week) We will introduce the RL problem both intuitively and formally and discuss practical examples. We will also discuss high-level challenges associated with RL:

- Long-horizon reasoning (Chpt. 4)
- Continuous actions (Chpt. 5)
- Task specification (Chpt. 6)

2. **Do you really need RL?** (2 weeks) Before introducing our first RL algorithms, we will talk about some easier approaches that can be used when extra information about your problem is available. These easier approaches will begin to introduce the technical foundation for the RL algorithms that we will subsequently study.
3. **3 simple RL methods** (1 week) We will introduce our first practical RL methods, which all focus on directly learning a policy. These methods will be very simple, but will only work in limited settings. To lift these limitations, we will need to introduce the value function (next chapter).
4. **Value functions** (4 weeks) We will introduce value functions, discuss how they are learned and why they can lift limitations of the simple RL methods introduced above. DQN will be our canonical example. The main limitation of these methods is that they require a small number of discrete actions, a limitation that the next chapter will address. This chapter will conclude with a midterm held right before Spring Break. Chapter 5 will start after Spring Break
5. **Actor critic methods** (3 weeks) To address practical problems where actions are continuous, we will study methods that combine a learned value function with a learned policy. PPO, TD3, and SAC will be the canonical algorithms studied in this section.
6. **Where do rewards come from?** (3 weeks) We will discuss techniques for specifying tasks and learning from limited feedback.

4 Course logistics

1. Waitlist. We have taken the first batch of students off the waitlist. Priority given to seniors. So, currently have 119 students enrolled in the course, above the course cap of 100 students. If you have questions about the waitlist, email Cathy (TA). We will pull more students from the waitlist of enrollment drops below 100. If you must take the course to graduate, talk to me. We will offer the course in S26.
2. Syllabus quiz. Make sure to ask for names
   (a) When are lectures?
   (b) When are precepts?
   (c) When is the midterm?
   (d) Is there a final exam?
   (e) How is final grade computed.
   (f) Should you come to class?
   (g) Who do you ask if you have questions about the waitlist?
   (h) How do you find office hours?
   (i) How should you ask questions about lecture, assignments?
   (j) Can you work in groups for the homeworks?
   (k) Can I use my laptop in class?
   (l) What is the policy about using LLMs and GenAI?
   (m) Will there be readings in the course?
3. Who should take this course?

Go to precepts on Friday. HWo due early next week.

5 Probability

What is a random variable?

- A random variable is a variable that can take on different values randomly.
- Takes on values in an alphabet $\mathcal{X}$. E.g., $\{0,1,2,3\}$, $\mathbb{R}$, $[0,1]^d$
- Can be continuous or discrete.

Probability distributions $p(x)$ must satisfy:

$$\text{DISCRETE:} \quad \sum_{x\in\mathcal{X}} p(x) = 1 \quad \text{and} \quad 0 \le p(x) \le 1 \text{ for all values of } x.$$

$$\text{CONTINUOUS:} \quad \int_{\mathcal{X}} p(x) = 1 \quad \text{and} \quad p(x) \ge 0 \text{ for all values of } x.$$

The **expected value** of a random variable $x \sim p(x)$ is:[3]

$$\mathbb{E}[x] \triangleq \int_{\mathcal{X}} p(x)x dx.$$

Similarly, we can write the expected value of a conditional random variable $x \sim p(x \mid y)$ as

$$\mathbb{E}[x \mid y] = \int_{\mathcal{X}} p(x \mid y) dx.$$

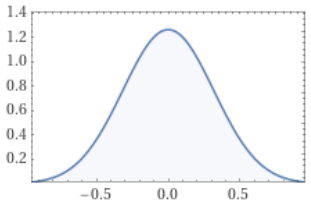

Figure 4: Probability densities can be greater than 1, as shown by a $\mathcal{N}(\mu = 0, \sigma = 0.1)$ distribution.

Brief note about wording. Typically, we will use "probability mass function" to refer to probability distributions over discrete x and "probability density function" for continuous x. For continuous random variables, we can have $p(x) > 0$ (Fig. 4). Strictly speaking, the *probability* of a continuous random variable taking a certain value is 0: $\mathbb{P}(x = 0.3) = 0$. Here's one way to think about this. Imagine that you have a needle sticking up in a bin, and drop a grain on salt randomly inside that bin. The probability that the grain of salt will end up balanced on the point of the needle is zero – it will never happen. However, the *probability density* is still non-zero – if the grain of salt is dropped uniformly at random inside the bin, then the probability density of it landing at any point in the bin is $p(x) = \frac{1}{\text{BinVolume}}$.

5.1 Joint distributions.

A joint distribution $p(x, y)$ can be viewed as an single-variate distibution over the alphabet $\mathcal{X} \times \mathcal{Y}$, satisfying the usual properties. For example, if x and y are both continuous, then the joint distribution must satisfy $\int_{\mathcal{X}\times\mathcal{Y}} p(x, y)dxdy = 1$. Note that we can also have joint distributions where x and y have different *types*. For example, when modeling the state of a car, x might be an image and y might be a LIDAR scan.

For a joint distribution $p(x, y)$, we can define the *marginal distributions* as $p(x) = \int_{\mathcal{Y}} p(x, y)dy$ and $p(y) = \int_{\mathcal{X}} p(x, y)dx$.[4] The marginal distributions tells you about the state of the world in the absence of any information about one of the variables (e.g., if a car's LIDAR breaks).

The conditional distribution $p(x \mid y)$[5] is an updated belief about the state of the world, given some observation. Let's say that we observe a concrete value $y = 4$. Then the

[3] I will use the symbol "$\triangleq$" to denote "is defined to be equal to," in the same way that computer scientists sometimes use ":=". Please let me know if/when notation is unclear. The site `https://detexify.kirelabs.org/` is also sometimes useful for looking up unfamiliar notation.

[4] Moving forward I'm going to assume everything is continuous, unless explicitly stated. The math still works for discrete objects.

[5] Use `\mid` to typeset conditional distributions.

conditional distribution satisfies

$$p(x \mid y=4) = \frac{p(x, y=4)}{p(y=4)} = \frac{p(x, y=4)}{\int p(x, y=4) dx}.$$

We can think about conditional distributions as a table, with $\mathcal{X}$ on one axis and $\mathcal{Y}$ on the other axis (see Table 1). The cells of the table sum to 1. Conditioning corresponds to selecting one row (or column) of the table, and then re-normalizing that row so that it sums to one.

	$\mathcal{X}$				
	0.1	0.03	0.05	0.0	0.06
	0.0	0.0	0.08	0.01	0.1
	0.01	0.02	0.04	0.04	0.1
$y=4$	0.01	0.01	0.0	0.03	0.2
	0.08	0.0	0.0	0.02	0.05

Table 1: Visualizing a conditional distribution for discrete random variables.

This same "table" analogy works in higher dimensions, as shown in Fig. 5.

There can be some notational confusion about conditional distributions: is $p(x \mid y)$ a function of one input or two inputs? This really depends on the problem setting, on whether you're just analyzing this function mathematically, or really have observed a certain value of y. We will use notation like $p(x \mid y=4)$ in instances when we are only considering varying x.

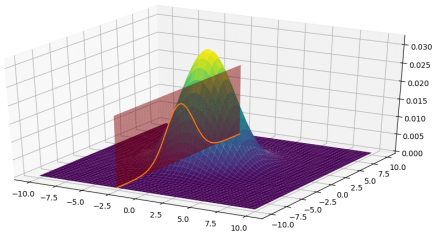

Figure 5: Visualizing a conditional distibution of continuous random variables. Animation. Note the similarity with Table 1.

We say that random variables x, y are **conditionally independent** if observing one of them doesn't tell you anything about the other. Formally, we might write this as

$$p(x \mid y) = p(x) \qquad \text{for all values of } x, y. \tag{2}$$

5.2 Chain Rule

The chain rule lets you write a joint distribution as a product of certain conditional distributions:

$$p(x, y, z) = p(x \mid y, z) p(y \mid z) p(z). \tag{3}$$

The same rule applies if everything is conditioned on some additional variable (say) a:

$$p(x, y, z \mid a) = p(x \mid y, z, a) p(y \mid z, a) p(z). \tag{4}$$

The process of writing a joint distribution in terms of conditional distributions is known as **factoring**. Factoring becomes interesting when certain random variables are independent of one another – that allows you to exclude certain terms on the RHS of the "|". For example, suppose that z is conditionally independent of x given y. That is, $p(z \mid x, y) = p(z \mid y)$ for all values of x, y, z. Then we can factor the joint distribution as $p(x, y, z) = p(x) p(y \mid x) p(z \mid y)$.

5.3 Bayes' Rule

Bayes' Rule allows us to use observations of one random variable to make guesses about the values of another random variable. Often we have some sense of how observations are generated. For example, consider the GPS on your phone, which gets noisy measurements y of distances to satellites orbiting above, which depend on your true (unknown) GPS

location, x. If we have an estimate for what the distances y will look like given your current location x, then Bayes' Rule tells us how to use that information to estimate your current location:

$$p(x \mid y) = \frac{p(y \mid x)p(x)}{p(y)}. \tag{5}$$

If you ever forget Bayes' Rule, it's easy to rederive from the following identity:

$$p(x, y) = p(y \mid x)p(x) = p(x \mid y)p(y). \tag{6}$$

Bayes' Rule will be incredibly useful in this course because it will allow us to infer the solutions to decision making problems. Precisely, we will use it to answer the counterfactual question: if you start at A and want to get to B, what states might you visit or what actions might you take? Often we have data or some model that can tell us the likelihood of getting to B if we take some action (or visit some state); plugging this into Bayes' Rule allows us to identify the best actions for arriving at B:

$$p(\text{action} \mid \text{start} = A, \text{end} = B) = \frac{p(\text{end} = B \mid \text{action}, \text{start} = A)p(\text{action} \mid \text{start} = A)}{p(\text{end} = B \mid \text{start} = A)}.$$

Note that in this application of Bayes' rule, each of the four terms is conditioned on the same event "start $= A$".

5.4 Important distributions

In this course we will frequently make use of the geometric distribution $\textsc{Geom}(1-\gamma)$ and the $\textsc{Bernoulli}(1-\gamma)$ distribution. The Bernoulli distribution with parameter p is a coin toss:

$$p(x) = \begin{cases} 1 & \text{with probability } 1-\gamma \\ 0 & \text{with probability } \gamma. \end{cases}$$

The expected value of a Bernoulli random variable is $\mathbb{E}[x] = 1-\gamma$.

The geometric distribution corresponds to the number of times you have to toss that coin until it lands on heads:

$$p(x) = (1-\gamma)\gamma^x \qquad \text{for } x = 0, 1, \cdots.$$

The expected value of a geometric distribution is $\mathbb{E}[x] = \frac{1}{1-\gamma}$. This tells you, on average, how many times you have to toss the coin until it lands on heads. I have defined these distributions with a parameter $1-\gamma$, rather than γ, because it will highlight connections with reinforcement learning in later lectures.

References