STA 4273: Minimizing Expectations

Lecture 1 - A common problem

Chris J. Maddison

University of Toronto

This course

- An exploration of the frontiers in (variational) Bayesian machine learning and reinforcement learning.
  - Gradient estimation, variational inference, offline policy evaluation, policy optimization, and the interplay between control and inference.
- These two subfields have very different motivations and are studied by different communities. But, they share a common mathematical structure and (increasingly) influence each other.
- This is a seminar course, so the majority of the course will consist of readings, student presentations, discussions.
  - For the first two weeks I'll be going through motivation, basic tools, and jargon.

This course

- What I hope you get out of this:
  - A unified view of two subfields that are rarely taught together.
  - A chance to practice research skills.
  - A chance to discover new subfields and open questions.
  - The assessment worth the most marks is a final project.
    - You will have a lot of freedom and input / feedback from.
    - I want to give you the room to really engage.
    - A wildly successful project might be submitted to a conference, but this is not necessary for top marks.
- My role:
  - Curate the readings, guide the discussion, and help explain how our readings connect to a broader context.
  - I want to put most of my energy into helping you give high-quality presentations and produce high-quality projects.

Outline

- Admin
- A tale of two problems
  - Unsupervised learning & variational inference
  - Reinforcement learning
- A common problem: minizing expectations

Course Information

Course Website:
`https://www.cs.toronto.edu/~cmaddis/courses/sta4273_w21/`

- We will use Quercus for announcements and assignments.
  - You should all have been automatically signed up. Did anyone not receive the announcements this week?
- We will use Piazza for discussions and forming groups.
- We will try GatherTown for office hours.
  - Chris Maddison, Thursdays 4PM-5PM.
- URLs for all of this are on Quercus.

The only things you need to pay attention to are the course website for content updates and Quercus (i.e., email) for announcements.

Course Information

- Lectures will be delivered synchronously via Zoom, and recorded for asynchronous viewing by enrolled students and auditors.
- You may download recorded lectures for your own academic use, but you should not copy, share, or use them for any other purpose.
- During lecture, please keep yourself on mute unless called upon.
- In case of illness, you should fill out the absence declaration form on ACORN and notify the instructors to request special consideration.
- For accessibility services: If you require additional academic accommodations, please contact UofT Accessibility Services as soon as possible, `studentlife.utoronto.ca/as`.

Course Information

Recommended readings will be given for each lecture. But the following will be useful throughout the course:

- Asmussen and Glynn: "Stochastic Simulation: Algorithms and Analysis"
- Wainwright and Jordan: "Graphical Models, Exponential Families, and Variational Inference"
- Sutton and Barto: "Reinforcement Learning: An Introduction"
- Bertsekas: "Dynamic Programming and Optimal Control"

Even more on the course website.

Assignments and Grading

- 25% – Paper presentation and code notebook
  - Handout on course website
- 15% – Project proposal
  - Due Feb. 17, handout coming soon!
- 60% – Project report
  - Due April 7, handout coming soon!

Presentation

- You will present a paper and your goal is to communicate the key ideas of the paper. See handout on the website for more details.
- To make things run smoothly, you will be assigned a week to present (weeks 3 - 12).
  - We are collecting your preferences via a Google form (see Quercus announcements).
  - Please fill this out ASAP.
- If you decide to drop the course (don't feel bad), but please email me so that I can make sure we have enough presenters!

Presentation Timeline

- Two weeks before your presentation
  - Meet me after class and we will decide on which papers to present and how to organize teams.
- A few days before your presentation
  - Meet with one of the teaching staff to practice your presentation and get feedback.
- Day of your presentation
  - Present!

Presentation Tips

- Cite the paper and the paper's authors on the first slide.
- You will need to build a bit of a background in order to understand the key contribution of your paper.
    - Read / skim some of the other papers in the same stack for your presentation week.
    - Follow the citation graph, use Google Scholar.
    - Your presentation should be able to comment on where your paper sits in the literature.
- Literally script your first sentence.
- Ask for help!

Code notebook

- You will be submitting a code notebook (Jupyter or Google colab) that demonstrates a key idea of the paper that you are presenting.
  - Can be a toy experiment or a visualization.
  - It can be the same toy experiment as in the paper, or one you design. Extra marks for originality.
- You can write this notebook in Python or R.
  - We *strongly recommend* Python, because we cannot provide support for R.
  - Must be runnable on Google Colab or UToronto's Jupyter Hub.
- Due the same day as your presentation.
- See handout for details.

More on Assignments

Collaboration on the assignments is allowed. You can use Piazza to find teammates. We will restrict the size of teams for various reasons. We are capping the size of presentation teams at 2 and project teams at 4. You can work alone, but we will not modify the marking scheme.

The schedule of assignments will be posted on the course web page.

Assignments should be handed in by 11:59pm; a late penalty of 10% per day will be assessed thereafter (up to 3 days, then submission is blocked).

Extensions will be granted only in special situations, and you will need a Student Medical Certificate or a written request approved by the instructor at least one week before the due date.

Prerequisites

- No formal prerequisites, but I assume you know the basic language of machine learning or statistics. The course is designed to be flexible otherwise.
  - Your project can be more applied to more theoretical, so you can play to your strengths.
  - For your presentation, we can try to get you a paper that you feel comfortable with.
- The marking schemes are there to help guide your efforts.
- *This being said*, I would recommend a certain level of mathematical maturity to get the most out of the course. This will involve linear algebra, calculus, basic probability, and programming skills.

Questions?

?

Unsupervised learning & variational inference

Unsupervised learning

- Most statistical problems do better if you have more data, but we very often have limited data.
- What about generating our own data?
- Suppose we have a dataset of observation $\{x_i\}_{i=1}^n$ (e.g., each x_i is a vector containing the pixel values of a color image).
    - Can we learn to generate *new* data points x that look like the ones in our dataset?
    - This is an example of the problem of unsupervised learning.

Unsupervised learning

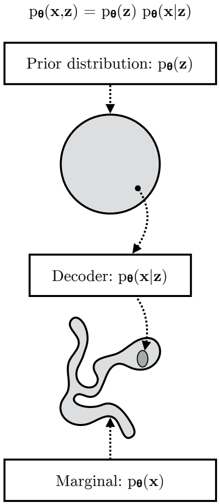
- One of the major paradigms of unsupervised learning is based on maximum likelihood estimation.
  - Assume $\mathbf{x} = (x_i)_{i=1}^n$ is generated from a probability distribution with density $p \in \mathcal{P}$ in some family of densities. Try to find

$$p^* = \arg\max_{p \in \mathcal{P}} \log p(\mathbf{x})$$

- This paradigm is one of the most successful in machine learning. Let's see some recent examples.

Variational autoencoders

1. Variational autoencoders model the data by mapping a simple distribution (e.g., a multivariate Gaussian) through a decoder that warps and stretches out the probability mass.
   - NVAE (Vahdat and Kautz, 2020) is a latest advance.
2. They can be trained to model very high dimensional x_i.



$p_\theta(\mathbf{x},\mathbf{z}) = p_\theta(\mathbf{z})\, p_\theta(\mathbf{x}|\mathbf{z})$
Prior distribution: $p_\theta(\mathbf{z})$
Decoder: $p_\theta(\mathbf{x}|\mathbf{z})$
Marginal: $p_\theta(\mathbf{x})$

(Kingma and Welling, 2019)

NVAE

Each x_i is an arrays of pixel intensities for a handwritten digit.

(Vahdat and Kautz, 2020)

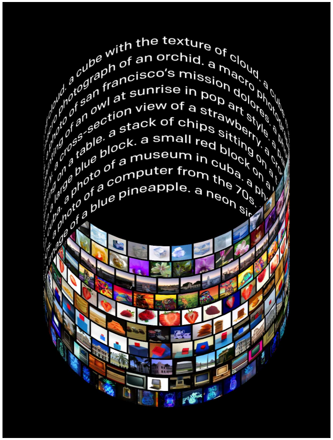
NVAE

Each x_i is an arrays of RGB pixel values for a face.

(Vahdat and Kautz, 2020)

DALL-E

1. DALL-E (built by OpenAI in San Francisco) is the most recent excitement in generative models, released just a few weeks ago.
2. It models text-image $(\mathbf{s}, \mathbf{x})$ pairs using neural networks, and some ideas related to variational autoencoders.
3. The model architecture makes it easy to generate images given a text prompt $p(\mathbf{x}|\mathbf{s})$.
   - Let's play with the model...


a cube with the texture of cloud. a macro photograph of an orchid. a photo of san francisco's mission dolores. a photo of an owl at sunrise in pop art style. a cross-section view of a strawberry. a stack of chips sitting on a table. a large blue block. a small red block on a museum in cuba. a photo of a computer from the 70s. a photo of a blue pineapple. a neon sign

DALL-E, OpenAI

Latent variable models

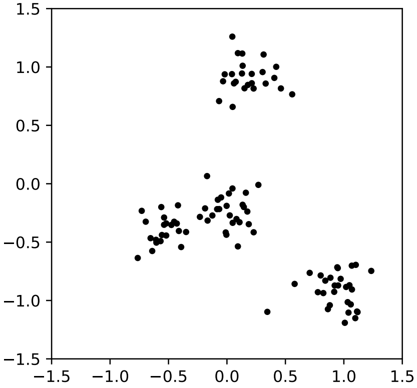
- Both of these examples use a latent variable model. Assume that each $p \in \mathcal{P}$ is the marginal distribution of a probability distribution over an extended space with a latent variable $\mathbf{z}$:

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

- Learning in this setting is challenging, and most methods rely on Bayesian inference, i.e., computing $p(\mathbf{z}|\mathbf{x})$.
- This is where variational inference shines, as we will illustrate with an example.

Example

We observe a dataset of points $\{x_i\}_{i=1}^n \subseteq \mathbb{R}^D$.

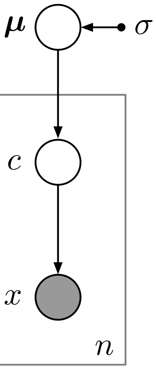


This data seems to clump, let's design a model that can capture "clumps".

Bayesian mixture of Gaussians

Consider the following latent variable model for our data.

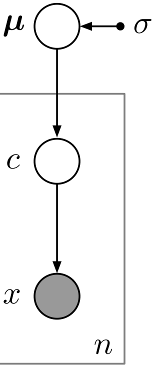
1. Latent: $\boldsymbol{\mu} \in \mathbb{R}^{K \times d}$ is a matrix of independent, latent, normal, random variables with mean 0 and variance σ^2.
2. Latent:$c_i \in \{0,1\}^K$ for $i = 1, \dots, n$ are independent one-hot, uniform, latent, categorical random variables.
3. Observed: $x_i \in \mathbb{R}^D$ for $i = 1, \dots, n$, which are independent, normal random variables with mean $c_i^T \boldsymbol{\mu}$ and variance 1.



Bayesian mixture of Gaussians

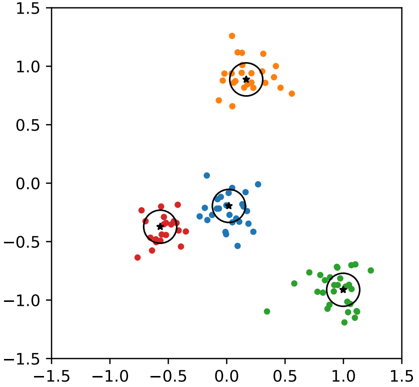
Consider the following latent variable model for our data.

$$\mu_k \sim \mathcal{N}(0, \sigma^2 I) \quad k = 1, \ldots, K$$
$$c_i \sim \mathrm{cat}(1/K, \ldots, 1/K) \quad i = 1, \ldots, n$$
$$x_i | c_i, \boldsymbol{\mu} \sim \mathcal{N}(c_i^T \boldsymbol{\mu}, 1) \quad i = 1, \ldots, n$$



Toy Example

In fact, our data was a sample from this model!



Here the points are coloured according to c_i and black stars are μ_k.

Our goal: Bayesian inference

- Recall: we want the conditional distribution of $\boldsymbol{\mu}, \{c_i\}_{i=1}^n$ given $\{x_i\}_{i=1}^n$, i.e.,
$$p(\boldsymbol{\mu}, \{c_i\}_{i=1}^n | \{x_i\}_{i=1}^n)$$
- Could use MCMC, but that might be slow and hard to scale.
- The main idea behind variational inference is to turn our Bayesian inference problem into an optimization problem.

Variational inference

Let $\mathbf{z} = \{\boldsymbol{\mu}\} \cup \{c_i\}_{i=1}^n, \mathbf{x} = \{x_i\}_{i=1}^n$ and consider the following optimization problem.

$$q^*(\mathbf{z}) = \arg\max_{q \in \mathcal{Q}} \mathbb{E}_{\mathbf{z}\sim q}\left[\log \frac{p(\mathbf{z},\mathbf{x})}{q(\mathbf{z})}\right]$$

Where

- $q \in \mathcal{Q}$ is a probability density in a family of probability densities.
- The objective is called the evidence lower bound (ELBO):

$$\mathbb{E}_{\mathbf{z}\sim q}\left[\log \frac{p(\mathbf{z},\mathbf{x})}{q(\mathbf{z})}\right] = \int q(\mathbf{z}) \log \frac{p(\mathbf{z},\mathbf{x})}{q(\mathbf{z})} d\mathbf{z} := \text{ELBO}(p, q, \mathbf{x})$$

KL Divergence

- Let us re-write this problem to understand its optimum:

$$\begin{aligned}
\arg\max_{q\in\mathcal{Q}} \mathbb{E}_{\mathbf{z}\sim q}\left[\log \frac{p(\mathbf{z},\mathbf{x})}{q(\mathbf{z})}\right] &= \arg\max_{q\in\mathcal{Q}} \mathbb{E}_{\mathbf{z}\sim q}\left[\log \frac{p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z})} + \log p(\mathbf{x})\right] \\
&= \arg\max_{q\in\mathcal{Q}} \mathbb{E}_{\mathbf{z}\sim q}\left[\log \frac{p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z})}\right] \\
&= \arg\min_{q\in\mathcal{Q}} KL(q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x}))
\end{aligned}$$

- $KL(q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x})) = 0$ iff $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x})$ almost everywhere.

Variational inference

Thus, for this variational objective,

$$q^*(\mathbf{z}) = \arg\max_{q \in \mathcal{Q}} \mathbb{E}_{\mathbf{z}\sim q}\left[\log \frac{p(\mathbf{z},\mathbf{x})}{q(\mathbf{z})}\right]$$

- its solution is is equal to $p(\mathbf{z}|\mathbf{x})$ a.e. if $\mathcal{Q}$ is the set of all possible probability densities.
- But this problem is as hard as our original problem!

Variational inference

This is a variational objective,

$$q^*(\mathbf{z}) = \arg\max_{q \in \mathcal{Q}} \mathbb{E}_{\mathbf{z}\sim q}\left[\log \frac{p(\mathbf{z},\mathbf{x})}{q(\mathbf{z})}\right]$$

The main idea of variational inference is

- to restrict the family $\mathcal{Q}$ to make this optimization (more) tractable.
- use q^* in lieu of $p(\mathbf{z}|\mathbf{x})$.

Mean-field VI

- The mean-field assumption is to assume very strong independence structure in $\mathcal{Q}$, i.e.,

$$q(\mathbf{z}) = \prod_{j=1}^{m} q_j(z_j)$$

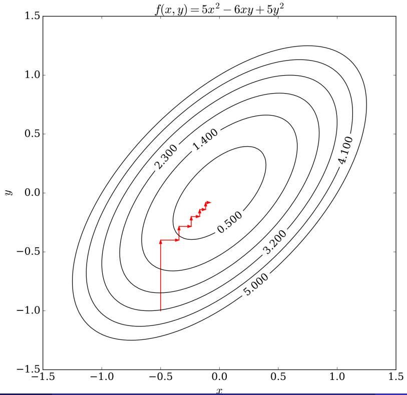
  - Typically q_j is taken to be in some restricted family, e.g., it is assumed to be $\mathcal{N}(\nu_j, s_j^2)$.
  - See Parisi, G. (1988) *Statistical Field Theory* for the physics origins of these ideas.

- In our example, we would have a separate Gaussian q_j for each element of the matrix $\boldsymbol{\mu}$ and a separate categorical for each c_i, so $m = K \times d + n$.

CAVI

Coordinate ascent mean-field variational inference (CAVI) is a general optimization routine for the variational problem under the mean-field assumption that performs coordinate descent by optimizing on q_j at a time, holding the others fixed.

Coordinate descent



CAVI

This is CAVI is a bit more detail.

1. Initialize the parameters of q_j for all $j = 1, \ldots, m$.
2. Until convergence, for $j = 1, \ldots, m$ find

$$q_j^* = \arg\max_{q_j} \mathrm{ELBO}(p, \textstyle\prod_j q_j, \mathbf{x})$$

   and set $q_j = q_j^*$.

Note: ELBO is not (generally) convex in q, so this procedure converges to a local optimum.

Let's see how to compute one of the inner optimizations.

CAVI

Now, assume that q_j for each μ_{kd} is $\mathcal{N}(\nu_{kd}, s_{kd}^2)$ for $k = 1, \ldots, K$ and $d = 1, \ldots, D$ and that q_j for each c_i is a categorical.

Consider the case where z_j is one of the categoricals,

$$
\begin{aligned}
&\arg\max_{q_j} \mathrm{ELBO}(p, \textstyle\prod_j q_j, \mathbf{x}) \\
&= \arg\max_{q_j} \mathbb{E}_{\mathbf{z}\sim\prod_j q_j} \left[\log p(\mathbf{z}, \mathbf{x}) - \log q_j(z_j)\right] \\
&= \arg\max_{q_j} \sum_k \left(q_{jk} \left(\sum_d \mu_{kd}\nu_{kd} - s_{kd}^2/2 - \nu_{kd}^2/2 \right) - q_{jk} \log q_{jk} \right)
\end{aligned}
$$

where q_{jk} is the probability that $z_{jk} = 1$ (equiv. $c_{ik} = 1$).

CAVI

Let's unpack what this looks like,

$$\arg\max_{q_j} \sum_k \left(q_{jk} \left(\sum_d \mu_{kd}\nu_{kd} - s_{kd}^2/2 - \nu_{kd}^2/2 \right) - q_{jk} \log q_{jk} \right)$$

Notice that it has the following form:

$$\arg\max_{q} \mathbb{E}_{c\sim q}[f(c)] + H(q).$$

where H is the entropy of the one-hot categorical $c \sim q$ and $f : \{0,1\}^K \to \mathbb{R}$ is some function.

Entropy-regularized objectives

We have reduced the inner optimization to a problem of the following form.

$$\arg\max_q \mathbb{E}_{c\sim q}[f(c)] + H(q).$$

Notice these two terms "compete"

- $\mathbb{E}_{c\sim q}[f(c)]$ encourages q to be "peaked" about $c^* = \arg\max_c f(c)$.
- $H(q)$ encourages q to be "broad"

Problems of this type will be a recurring theme throughout the course. They are sometimes called entropy regularized objectives. Actually the ELBO is itself an example.

Entropy-regularized objectives

To solve this problem, define

$$q^*(c) = \frac{\exp(f(c))}{\sum_{c'} \exp(f(c'))}$$

and notice

$$\begin{aligned}
&\arg\max_q \mathbb{E}_{c\sim q}[f(c)] + H(q) \\
&= \arg\max_q \mathbb{E}_{c\sim q}[f(c)] - \log\left(\sum_{c'} \exp(-f(c'))\right) + H(q) \\
&= \arg\max_q \mathbb{E}_{c\sim q}[\log q^*(c)] + H(q) \\
&= \arg\min_q KL(q(c) \parallel q^*(c))
\end{aligned}$$

This means that q^* is the optimum of our subproblem!

CAVI

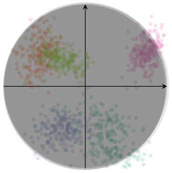
Going back to CAVI, we have just derived the solution of the subproblem for the q_j for the categoricals.

$$q_j^*(c) = \frac{\exp(\boldsymbol{\mu}^T c c^T \boldsymbol{\nu} - \left\| c^T \mathbf{s} \right\|^2 / 2 - \left\| c^T \boldsymbol{\nu} \right\|^2 / 2)}{\sum_{c'} \exp(\boldsymbol{\mu}^T c' c'^T \boldsymbol{\nu} - \left\| c'^T \mathbf{s} \right\|^2 / 2 - \left\| c'^T \boldsymbol{\nu} \right\|^2 / 2)}$$

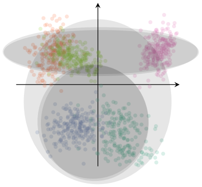
One can go through similar reasoning to derive the mean and variance of q_j^* for the Gaussians μ_{kd}.

Let's see a visualization of the algorithm.

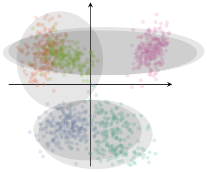
CAVI



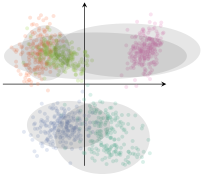
Initialization



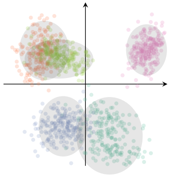
Iteration 20



Iteration 28



Iteration 35



Iteration 50

Variational inference and the evidence lower bound

- What does this have to do with pretty pictures? Well, you can show

$$\mathrm{ELBO}(p, q, \mathbf{x}) \leq \log p(\mathbf{x})$$

- The ELBO can be used as a surrogate objective for maximum likelihood estimation in very complex latent variable models, which we will look at.

$$\arg\max_{p \in \mathcal{P}} \max_{q \in \mathcal{Q}} \mathrm{ELBO}(p, q, \mathbf{x})$$

- Those fancy image models were trained using an ELBO.

Variational inference and this course

- This course will explore optimizing ELBOs (and related objectives) in a variety of settings.
- The emphasis will be on deep learning settings (i.e., we will assume far less structure)
- We will be exploring the following themes:
  - Using ELBO to train deep generative models (pretty pictures).
  - More exotic state spaces for $\mathbf{z}$, e.g., the space of functions.
  - Tighter variational bounds for maximum likelihood.
  - Variational objectives for other objectives (even supervised learning!).

Reinforcement learning

Reinforcement learning

- We just assumed that the data $\{x_i\}_{i=1}^n$ was collected once and i.i.d.
  - i.i.d. = independent and identically distributed, which is the most common assumption.
- Reinforcement learning is a subfield that makes very different assumptions on how data is collected.
  - Sometimes called control, but there are subtle differences in the communities that study this, etc.
- The main idea in reinforcement learning is that data is collected via an interactive process over a series of timesteps and the goal is to maximize a reward that is additive in time.
- The framework is useful when we have a desired outcome for some process, but we cannot characterize its optimal behaviour explicitly.

Atari

- Interaction: player pushes buttons and the games changes its state.
- You know the goal (maximize your points), but optimal behaviour is not always known explicitly.
- In 2013 DeepMind published a landmark paper that showed how to train neural networks to play a huge suite of Atari games.
  - Let's watch it play Breakout.

(Mnih et al., 2013)

Go

- Go was the last classical board games that humans still outperformed computers.
- Interaction: player picks a move and then the opponent picks a move.
- You know the goal (win), but optimal behaviour is not always known explicitly.
- DeepMind produced a Go bot (AlphaGo) that defeated the world champion Lee Sedol in 2016.
  - I was one of the founding members of this team!

CADE METZ BUSINESS 03.11.16 07:00 AM

THE SADNESS AND BEAUTY OF WATCHING GOOGLE'S AI PLAY GO

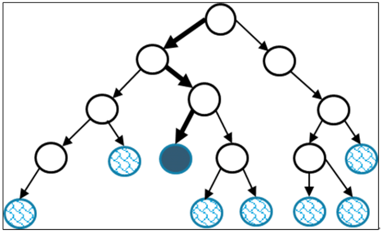
Wired

Data-driven Algorithm Design

- Many iterative algorithms include heuristic decisions at each iteration.
  - E.g., combinatorial solvers.
- Interaction: heuristic picks from a set of choices and the algorithm executes an iteration.
- Typically we have some goal (minimize run time), but the best heuristic for that goal is not known.
  - We can use reinforcement learning to optimize!

Data-driven Algorithm Design

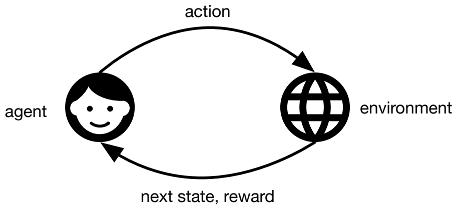
Consider branch-and-bound for discrete optimization.



(Shokry et al., 2018)

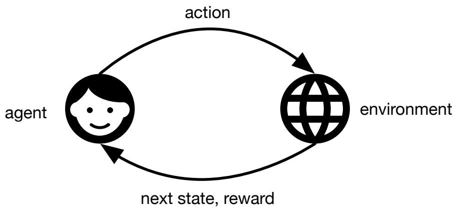
This method iteratively partitions a discrete optimization problem. Partition forms the leaves of a tree. Choosing which leaf to split next is a heuristic decision.

Reinforcement learning



- An agent (e.g. an Atari player, the player in Go, the heuristic) interacts with an environment (e.g. game of Breakout, the opponent in Go, the algorithm state)
- In each time step t,
  - in s_t agent picks an action a_t (e.g. keystrokes)
  - the environment transition to $s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$
  - the agent receives the next state s_{t+1} (e.g. positions of the ball and paddle) and a reward $r(s_t, a_t)$ (e.g. points)
- Let's assume, for now that there are finitely many possible actions a_t and finitely many states s_t.

Reinforcement learning



- An agent (e.g. an Atari player, the player in Go, the heuristic) interacts with an environment (e.g. game of Breakout, the opponent in Go, the algorithm state)
- The agent wants to learn a policy $\pi(a_t \mid s_t)$
  - Distribution over actions depending on the current state.

Markov Decision Processes

- The environment is represented as a Markov decision process $\mathcal{M}$.
- Markov assumption: all relevant information is encapsulated in the current state; i.e. the policy, reward, and transitions are all independent of past states given the current state
- Components of an MDP:
  - $\mathcal{S}$: State space. Discrete or continuous, but let's assume it is finite.
  - $\mathcal{A}$: Action space. We consider finite action space.
  - $p(s_{t+1}|s_t, a_t)$: Environment transition probability distribution.
  - $p(s_0)$: Initial state distribution.
  - $r(s_t, a_t)$: Bounded reward function (can be a random variable).
  - γ: Discount factor ($0 \leq \gamma < 1$).
- The agent operates in an MDP environment using a policy:
  - $\pi(a_t|s_t)$: a stochastic policy that the agent uses to choose action.
  - $\pi(s_t)$: a deterministic policy that the agent uses to choose action.

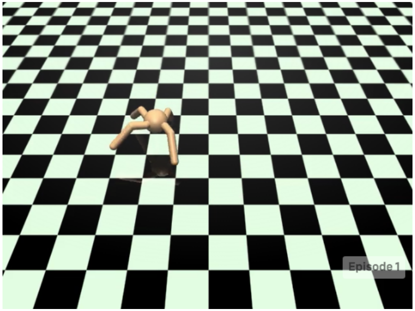
Markov Decision Processes

- Rollout, or trajectory $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T, a_T)$
- Probability of a rollout

$$p(\tau) = p(s_0)\,\pi(a_0 \mid s_0)\,p(s_1 \mid s_0, a_0) \cdots p(s_T \mid s_{T-1}, a_{T-1})\,\pi(a_T \mid s_T)$$

Markov Decision Processes

Continuous control in simulation, e.g. teaching an ant to walk



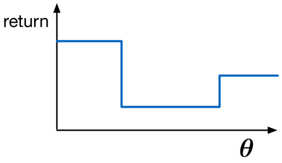
- State: positions, angles, and velocities of the joints
- Actions: apply forces to the joints
- Reward: distance from starting point
- Policy: output of an ordinary MLP, using the state as input
- More environments: `https://gym.openai.com/envs/#mujoco`

Markov Decision Processes

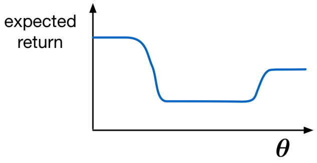
- Return for a rollout: $r(\tau) = \sum_{t=0}^{T} r(s_t, a_t)$
  - Note: This is assuming that T is fixed and finite. We'll discuss other cases shortly.
- Goal: maximize the expected return

$$\max_{\pi} \mathbb{E}_{\tau \sim p}[r(\tau)]$$

- The expectation is over both the environment's dynamics and the policy, but we only have control over the policy.
- The stochastic policy is important, since it makes R a continuous function of the policy parameters.
  - Reward functions and are often discontinuous (e.g. collisions)



deterministic policies



stochastic policies

Markov Decision Processes—Jargon

Reinforcement learning is not one problem. It is a collection of problems.

- Fully-observed. The policy fully observes the state s_t.
- Partially-observed. There is an observation distribution $p(o_t|s_t)$ and the agent observes $o_t \sim p(o_t|s_t)$ at time t. Policy $\pi(a_t|o_t)$ now depends on the observation o_t.

Markov Decision Processes—Jargon

Reinforcement learning is not one problem. It is a collection of problems.

- Bandits. $T = 1$, and we care about how much reward we get during training.
- Infinite horizon. $T = \infty$, and $r(\tau)$ may diverge, so we typically discount the return with $0 < \gamma < 1$:

$$r_\gamma(\tau) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$$

- Episodic. $1 < T < \infty$, and T may be random or fixed. $r_\gamma(\tau)$ may be discounted or not.

Markov Decision Processes—Jargon

Reinforcement learning is not one problem. It is a collection of problems.

- Stationary policy. π does not depend on t.
- Non-stationary policy. π_t does depend on t.

Markov Decision Processes—Jargon

- I hope you get the picture.
- *Technically*, a great deal depends on all of these choices (e.g., stationary policies are not optimal for finite horizon MDPs), and the theory can get a bit daunting. But the details can be a distraction. They key paradigm to remember is:
  - An interactive process between an agent and an environment.
  - The goal of the agent is to find a policy that maximizes its return, which is additive across time.

Reinforcement learning and this course

- We will focus on offline reinforcement learning and policy optimization
  - Offline RL: how can you use a fixed dataset of data to improve your policy?
  - Policy optimization: methods that attempt to directly optimize the policy (we will see alternatives next week).
- Why? These are among the most useful perspectives in practice, and very actively studied.
- In a bit more detail
  - How can we evaluate the quality of our policy without deploying the agent? Off-policy policy evaluation.
  - How can we directly optimize the policy? Policy optimization.
  - How can we optimize the policy using stored data? Offline policy optimization.
  - The relationship between tree search and policy optimization (AlphaGo).

A common problem

A common problem

- Consider the two problems that we encountered

$$\arg\max_q \mathbb{E}_{c\sim q}[f(c)] + H(q) \qquad \arg\max_\pi \mathbb{E}_{\tau\sim p}[r(\tau)]$$

- The only difference seems to be the temporal aspect of reinforcement learning and the entropy regularizer of variational inference.
    - However, entropy regularizers are often added to the reward in reinforcement learning!
    - However, many variational inference problems have time as a dimension and decompose additively over time!
- The main themes of this course are how to cope with restrictions on the family of q or π and how methods exploit specific structure in the model / environment.
- Next time: basic tools for solving these problems.