tabular reinforcement learning

a map of learning tasks

In the

Today we'll learn how to solve classical reinforcement learning problems. Compared to by-now supervised learning

motivating real-world examples include robotics, poker, and ad placement.

kinds of learning

How do we communicate patterns of desired behavior? We can teach:

by instruction: "to tell whether a mushroom is poisonous, first look at its gills..."
by example: "here are six poisonous fungi; here, six safe ones. see a pattern?"
by reinforcement: "eat foraged mushrooms for a month; learn from getting sick."

Machine learning is the art of programming computers to learn from such sources. We've so far focused on the most important case: learning from examples.

Initial

Learning by example is key to the other modes of learning. Today we'll discuss learning by reinforcement.

Online

Stateful (not independent) Stateless Extreme Stateless: only one input

The basic framework is that we have several situations and each permits several actions. Taking an action in a given situation leads to some (potentially noisy) reward. We would like to learn a how to act in any given situation so as to enjoy a high average reward.

	EXAMPLES	REWARDS
ONE	unsupervised	bandit
MANY	supervised	cond. bandit
SEQUENCE	imitation	classical RL

Figure 1: kinds of learning

bias-variance; exploitation-exploration

reinforcement –> explore vs exploit

online

stateful (fully observable / approximation of infinite state)

P vs NP

tabular vs leveraged; local gradient updates

bandits

unconditional

SETUP — Okay, let's say that each action $a \in \mathcal{A}$ in a known set of available actions leads to a reward sampled randomly independently from some unknown distribution $p_{r;\mathcal{A}}$. To keep things simple at the start, let's say $\mathcal{A}$ is finite and r takes values in $\{0, 1\}$. Thus, the world is determined by a probability p_a of nonzero reward for each action a. In fact, let's say there are only two actions: $a^\star = \text{argmax}_a p_a$ with reward $p^\star$ and a° with reward $p^\circ = p^\star - \Delta$. And let's say we go for some large number T of steps.

How high can we get our randomness-averaged time-totaled reward? It seems reasonable to compare this amount to the randomness-averaged time-totaled reward we always did the most rewarding action $a^\star$. We want to minimize the difference, i.e., our expected "**regret**".

Intuitively, if $\Delta \lesssim 1/\sqrt{T}$ we won't ever really know which action is best. In this case we don't expect to do better than $\sqrt{T}$ expected regret. So below we'll assume $1/\sqrt{T} \leq \Delta$.

Here's one strategy: we could sample each a say, $K = 10$ times, thereby estimating each $\hat{p}_a$. From then on, we'd stick to $\hat{a}^\star = \text{argmax}_a \hat{p}_a$. (Say $|\mathcal{A}|K \ll T$ so that this latter step dominates). Our expected regret then grows linearly with T:

Exercise: By the way, is $\hat{p}_{\hat{a}^\star}$ the same as $p_{\hat{a}^\star}$ on average?

$$\begin{aligned} \mathbb{E}\, \text{regret}_T \approx \mathcal{R} \triangleq &+ \Delta K && \leftarrow \textbf{explo\textit{ration} term} \\ &+ \Delta T \cdot \mathbb{P}\left[\hat{a}^\star \neq a^\star\right] && \leftarrow \textbf{explo\textit{itation} term} \end{aligned}$$

Intuitively, K trades off between the opportunity costs near the beginning and in the remainder of our timesteps. Larger K means we pay a slower start in return for more accurate estimates that'll inform us for the rest of time.

Our cost is at most

$$\Delta(K + T \exp(-K\Delta^2))$$

ORACLE CASE — Supposing we did know Δ and T, a best K would balance the marginal cost $\partial K/\partial K = 1$ of longer exploration against the marginal benefit $-\partial T \exp(-K\Delta^2)/\partial K = -T\Delta^2 \exp(-K\Delta^2)$ of more accurate exploitation:

$$K = \log(T\Delta^2)/\Delta^2$$

Then we incur cost°

$$\mathcal{R} = \frac{\log(T\Delta^2) + 1}{\Delta}$$

← Note that IF we over-explore by setting $K = \log(T)/\Delta^2$ — that is, by replacing $T\Delta^2$ by T — THEN our cost undergoes the same substitution to become $(\log(T) + 1)/\Delta$. With $\Delta = \rho/\sqrt{T}$, this cost grows with T like $\mathcal{R} \sim \log(T)\sqrt{T}$.

With $\Delta = \rho/\sqrt{T}$ for $1 \leq \rho$, this grows with T like $\mathcal{R} \sim \sqrt{T}$.

REALISTIC CASE — We don't know T, Δ. But, astonishingly, there is an algorithm that has expected regret nearly as low as when we *do* know T, Δ.

Intuitively, we want it to "explore" as many times K as if we knew T, Δ. But now, whether or not we do explore at timestep t may depend only on the length-t list of previous actions and rewards rather than on Δ and T.

Here's the clever insight: let's *estimate* Δ and T as follows: $T \approx t$ and $\Delta \approx |\hat{p}_\star - \hat{p}_\circ|$. After all, on average t is no more than a factor $1/2$ off from the true T — and since T appears in a log, it is plausible that we may neglect this difference.
TODO: "after all" heuristic for Δ

So, as long as the number $\hat{K}$ of 'explorations' so far is $\hat{K} < \log(T)/\Delta^2$, we'll want to do more exploration. Let's write $\hat{K} = \min(N_\circ, N_\star)$, the least number of experiencies we've had among actions. Then we want to try the empirically worse action $\hat{a}_\circ$ whenever: $\hat{p}_\star < \hat{p}_\circ + \sqrt{\log(T)/\min(N_\circ, N_\star)}$. With the understanding that $N_\star$ will typically be much bigger than $N_\circ$, the above condition is almost the same as this more symmetrical condition:

$$\hat{p}_\star + \sqrt{\log(t)/N_\star} < \hat{p}_\circ + \sqrt{\log(t)/N_\circ}$$

We now have a nice interpretation: at each point in time we choose an action with 'highest potential', where we judge an action a's potential not only by its empirical reward $\hat{p}_a$ but also by an uncertainty score or **explorer's bonus** $\sqrt{\log(t)/N_a}$. have high

TODO: Observe that most exploration usually happens near the very beginning of time.

As in the previous passage,° this incurs

$$\mathcal{R} \sim \frac{\log(T\Delta^2) + 1}{\Delta}$$

much cost, which is a $\log(T)$ factor within the oracle case!

← We may argue similarly as in the previous passage EXCEPT that we must be more careful because different actions are no longer independent!

CODE — Translating the realistic case algorithm to code is straightforward. We just have to worry about early-time edge cases where we would take logs or reciprocals of zero. A nice way to sweep these issues under the rug is to initialize our counts to small nonzero values.

```
actions = set([...])

tot_reward_by_action = {0.1 for a in actions} # We initialize to small, nonzero
nb_trials_by_action = {0.1 for a in actions}  #  values to avoid divide by zero.
T = sum(nb_trials_by_action.values()) # This relation we preserve for all time.

def reward_estimate(a): return tot_reward_by_action[a] / nb_trials_by_action[a]
def explorers_bonus(a): return np.sqrt(np.log(T) / nb_trials_by_action[a])
def next_action(a): return max(( reward_estimate(a)
                                +explorers_bonus(a), a) for a in actions)[1]
def learn_from(a, r):
    tot_reward_by_action[a] += r
    nb_trials_by_action[a] += 1
    T += 1
```

conditional

In the conditional case the reward may depend on some observed input s sampled i.i.d. from some unknown distribution on a known set $\mathcal{S}$. So the reward structure is determined by the symbol $p_{r|s;\mathcal{A}}$.

imitation

on-policy learning: TD

euler error

classical RL

on-policy learning: TD

off-policy learning: Q

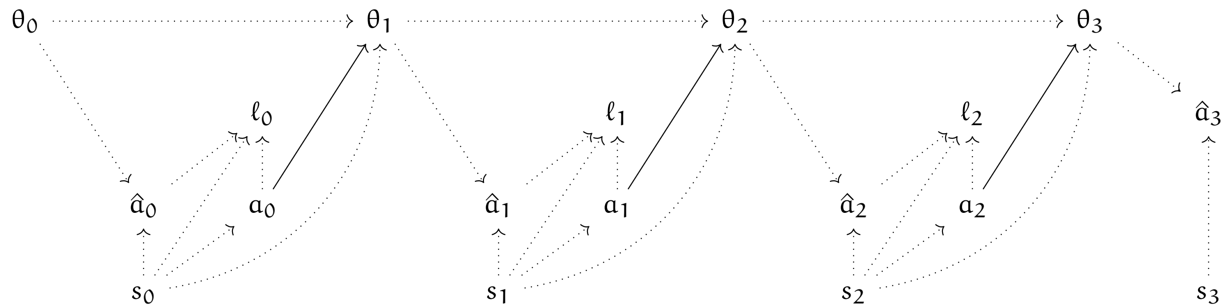
exploration strategies

bonus slide: data dependency diagrams

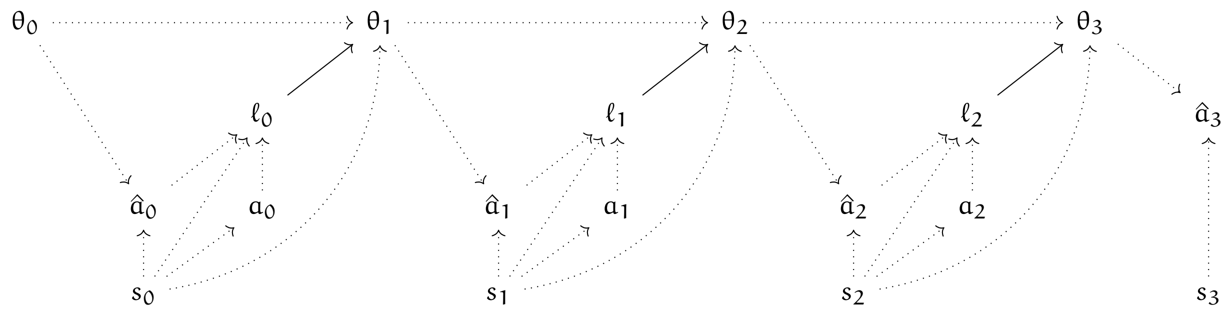
Let's visualize how weights and data affect each other in four kinds of (online)° learning problem. Solid and dashed arrows depict contrasts and commonalities. Here, θ stands for weights; s, for prompts or states; a, for true answers or ideal actions; $\hat{a}$, for guessed answers or actual actions; ℓ, loss.

← In **online** learning, we continuously learn as we receive a stream of data. Our goal is to experience small time-averaged loss.

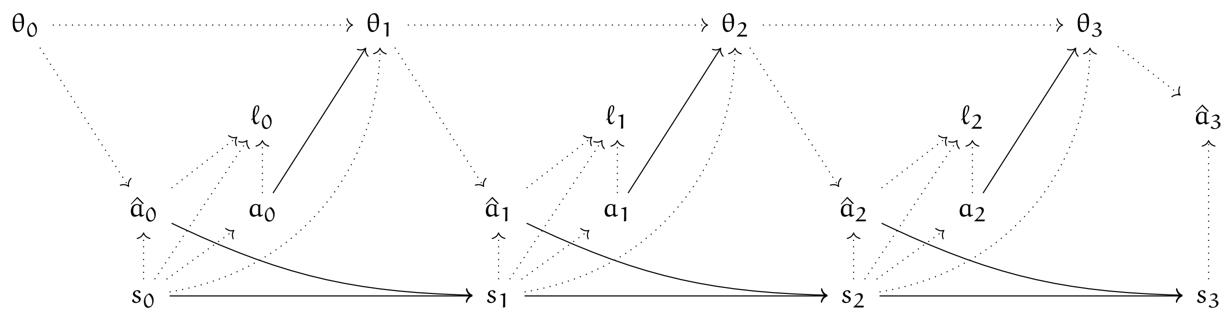
SUPERVISED — We learn from *examples* a. States are sampled independently.



CONDITIONAL BANDIT — We learn from *rewards* $-\ell$. States are sampled independently.



(SPECIAL CASE OF) IMITATION — We learn from *examples* a. The state evolves over time.



CLASSICAL REINFORCEMENT — We learn from *rewards* $-\ell$. The state evolves over time.