L20 – Bandit problems and RL “in the wild”

KAlg 15

What you should know

- Formally, reinforcement-learning problems are POMDPs
- Bandit problems are horizon-1 RL problems
- They have interesting special solutions

Reinforcement learning in the factory vs the wild

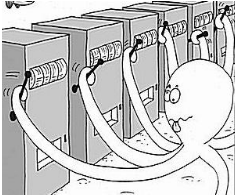
- One way that we use RL methods is in the "RL in the Matrix" setting[1]
  - We know a complete model of a complex MDP
  - We want a (near-optimal) policy for that MDP
  - MDP too complex for exact value iteration.
  - Use the MDP model to make a simulator (a generative transition model) and run an RL-like algorithm (ranging from fitted value iteration to Q learning) to get a policy
- RL methods were originally designed to model learning in a single individual, with a single "lifetime" that is trying to maximize its expected reward during its one life.
  - The problem of how to select actions is key: "Exploration" (trying actions that you have less information about) versus "Exploitation" (selecting actions that you think will have a good payoff.
  - You have to model your uncertainty about the way the world works. That's a belief.

[1] Lame movie reference

Bandit problems

Single-state MDP

- Action set $\mathcal{A}$ (usually discrete)
- Stochastic rewards $R : \mathcal{A} \to \mathcal{P}(\mathbb{R})$
- R is unknown!!
- Objective: select actions, conditioned on reward history, to maximize expected (possibly discounted) sum of future rewards.
- Usually we have some "prior" on R



Super-simple example: finite-state bandit

- Discrete $\mathcal{A} = \{1, \ldots, K\}$
- Binary rewards: $R(a) \in \{0, 1\}$ for all $a \in \mathcal{A}$
- Prior on $R(a)$:
    - Let $P(R(a_i) = 1) = p_i$
    - Limited possible number of probability values $p_i \in \{.1, .5, .9\}$
    - Uniform prior on those, independent for each action a_i:

$$P(p_i = .1) = P(p_i = .5) = P(p_i = .9) = 1/3$$

Super-simple bandit as a POMDP

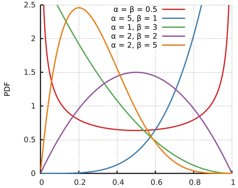
- $\mathcal{S} = \{.1, .5, .9\}^K$: that is, all possible choices of payoff probability for each action
- $\mathcal{A} = \{1, \ldots, K\}$ "arms" from bandit problem
- T is just the identity matrix (no actual state transitions)
- $R(s = (p_1, \ldots, p_K), a_i) = p_i$: expected reward of picking action i is its payoff probability (we get 1 with probability p_i and 0 otherwise)
- $\mathcal{O} = \{0, 1\}$: when we pull an arm, we observe what reward we get
- $O(s = (p_1, \ldots, p_K), a_i, 1) = p_i$
- γ (or finite horizon H)
- b_0 : uniform over $\mathcal{S}$

Solving super-simple bandit

- Because each observation depends only on the state of one arm, and the prior on the states can be factored, we can represent the belief state as K independent discrete distributions (one for the state of each arm).
- Belief update is standard Bayes filter update, only on the factor associated with the arm that was pulled.
- Belief space is two 3-simplices
- How to solve? Try exact solver or PBVI

More traditional Bernoulli bandit

- Payoff probability p_i for each arm in $[0, 1]$
- Uniform prior on p_i
- Now $\mathcal{S}$ is $[0, 1]^K$
- Standard POMDP methods don't apply
- Belief still factors per arm; but we need a continuous distribution over p_i. Use the **beta** distribution. It has two parameters–think of them as "pseudocounts" of number of positive and negative examples seen.
- Can still do forward-search type methods: expectimax, etc.



[2]

[2] Image credit By Horas based on the work of Krishnavedala - Own work, Public Domain, commons.wikimedia.org/w/index.php?curid=15404515

Approximation strategies and regret

The regret of a policy π is difference, in expectation, of the actual expected reward of executing the optimal policy (picking the best arm, always) and the expected reward of π.

- Thm: regret on trial of length n bounded below by $O(\log N)$.
- A UCB strategy of picking the arm with maximum

$$\hat{\mu}_i + \sqrt{\frac{2\log(1 + N\log^2 N)}{N_i}}$$

  where μ_i is the empirical average payoff for arm i so far, N is the total number of pulls of any arm and N_i is the total number of pulls of arm i, has $O(\log N)$ regret.
- Thompson sampling also has $O(\log N)$ regret:
  - Idea is to pick the arm that is most likely to be the best, according to the current belief
  - Implement by: drawing a sample, q_i, from the Beta posterior distribution on p_i for each arm i
  - Select arm with highest q_i

RL in an MDP as a POMDP

We can take this same view of reinforcement learning more generally: Bayesian RL:

- The state is completely observable, but R and T are unknown
- Agent has a belief over R and T—usually independent distributions on each parameter
- Exploration/exploitation is even trickier
- UCB-style exploration “bonuses” can be helpful but they have to be propagated around the MDP (like values) to motivate moving toward unexplored areas.