# A Brief Introduction to Bandits

CSCI2951-f
Ron Parr
Brown University

## One-armed bandits

- Rrepeatable (iid) processes w/constant payoff amount, unknown prob (can usually generalize to unknown payoff amounts)
- Examples (some w/variable payoff):
  - Trials of different drugs
  - Products to suggest to users
  - Routing paths for data
  - Financial portfolios
- Goal: Pick arms in a “smart” way

- Note: entire books & classes on bandit algorithms and extensions thereof (we just scratch the surface here)

## Different goals

- Figure out the optimal are in the limit
- Figure out the optimal arm in a finite time (no guaranteed method)
- Some PAC criterion (identify nearly optimal arm WHP)
- Maximize expected reward over a finite horizon
- Maximize expected discounted reward in the limit
- Minimize regret

## Methods for updating payoff estimates

- Maximum likelihood

- Bayesian

# Maximum likelihood

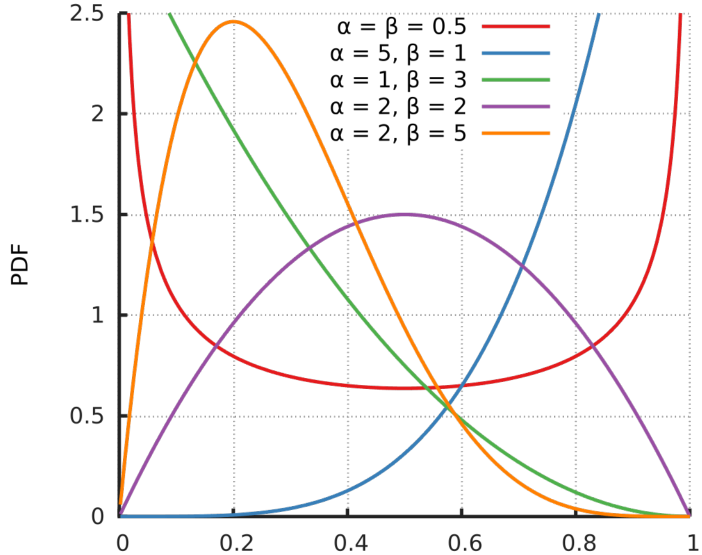
- Think of arm "a" as a Bernouli random variable w/unknown $p_a$
- Count number of payoffs: $w_a$
- Count number of pulls: $l_a$
- ML estimate of payoff: $p_a = w_{a/}(w_a + l_a)$
- Pros: Easy to compute
- Cons:
  - Behavior for small/no pulls
  - No incorporation of prior knowledge

# Bayesian approach

- Prior distribution on possible payoff probs for each arm
- beta($\alpha$,$\beta$) is conjugate for binomial distribution
- Expectation is: $\alpha/\alpha+\beta$
- Posterior given a positive example is beta($\alpha$+1,$\beta$)
- Posterior given a negative example is beta($\alpha$,$\beta$+1)

- Interpretation:
  - $\alpha$ and $\beta$ can be thought of as the number of previous positive/negative (heads/tails) examples we have seen
  - Used as a prior, it reflects a bias towards a particular value, and encodes the strength of this bias

# Visualizing the Beta distribution




$$\frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathrm{B}(\alpha,\beta)}$$

where $\mathrm{B}(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$



# Bayesian approach summary

- Advantages:
  - No harder to work with than maximum likelihood
  - Reasonable behavior for low sample size
  - Incorporates prior knowledge
  - Converges to ML estimate in the limit
- Cons: Where does prior knowledge come from?

- Extension to multiple outcomes:
  - Binomial -> multinomial
  - Beta -> dirichlet

## Simple strategies

- ε greedy

- Softmax

## ε–greedy

- Choose greedy action w.p. 1-ε
- Choose random action w.p. ε

- Advantage: Simple, widely used in RL
- Disadvantages:
  - Not very smart
  - How to pick ε

## Softmax

- Given values X1…Xk
- Choose index i with probability:

$$\frac{e^{\lambda X_i}}{\sum_{j=1}^{k} e^{\lambda X_j}}$$

- Uniform random for $\lambda = 0$
- Hard max as $\lambda \rightarrow \infty$

## Softmax pro/con

- Advantages:
  - Random choices favor (seemingly) better actions
  - Tunable between uniform and hard max
- Disadvantages:
  - Somewhat more expensive/complicated than $\varepsilon$-greedy
  - How to pick $\lambda$?

## Limiting properties of simple approaches

- So long as every arm is tried infinitely often ($\varepsilon$>0,$\lambda$< $\infty$)
- Estimates of payoff probabilities will converge to true estimates

- Comments:
  - Very weak statement
  - Doesn’t say anything about how much time is spend suboptimally

## PAC approaches

- Goal: Choose an $\varepsilon$ optimal arm w/prob 1-$\delta$
- Main tool: Hoeffding inequality
  - Given iid X1…Xm with empirical mean p, true mean $\theta$
  - True mean $\tau$ is in inside: [p-z/$\sqrt[2]{m}$, p+z/$\sqrt[2]{m}$] w.p. 1-$\delta$
  - z = $\sqrt[2]{1/2\ln(2/\delta)}$

- Take c samples of each arm $c = 2\epsilon^2 \ln(\frac{2k}{\delta})$
- Use union bound to show that this suffices

## PAC approach summary

- Similar arguments can be used for strategies for
  - Choosing suboptimal arm bounded number of times WHP
  - Achieve average reward that is close to optimal WHP
- Nice approach overall – simple to execute
- Cost of achieving guarantees can still be high
- Some probability of making lots of costly mistakes remains

## Dynamic programming/MDP approach

- Consider some finite horizon
- Number of possible outcomes is determined by number of steps (but exponential in number of steps)
- Define a state as counts of each outcome
- Define reward as payoff
- Policy that maximizes expected (discounted) reward is solution to the finite horizon MDP

## MDP Approach Pros/Cons

- Pro: Solution is optimal for finite horizon
- Con: Exponential size makes it impractical for long horizons and/or large numbers of arms

## Gittins indices

- Surprising result:
  - Finite horizon MDP formulation is intractable for long horizons
  - Infinite horizon discounted approach has a quirky, but efficient
- Idea behind Gittins indices
  - Compute an index (Gittins index) for each arm
  - Function of discount and distribution over possible payoffs given current knowledge
  - Computation of Gittens index also gives an optimal time to stick with each arm
  - Pick arm with highest Gittens index, and stick with it for recommended time
  - After time is up, recompute indices and pick a new arm

## Gittins index comments

- Viewed as a very complicated and cool result
- Computation is Gittens indices is not trivial
- Considered brittle: Works for maximizing discounted sum of rewards, but technique does not generalize to slight changes in problem setting or optimality criterion

## Regret Minimization

- Regret is the difference between actual returns and what you could have gotten if you picked the best arm from the beginning
- Methods discussed so far do not provide bounds on regret
- Choosing an epsilon optimal arm could have regret that grows linearly with the number of time steps

# UCB1

**Deterministic policy:** UCB1.
**Initialization:** Play each machine once.
**Loop:**

– Play machine $j$ that maximizes $\bar{x}_j + \sqrt{\frac{2 \ln n}{n_j}}$, where $\bar{x}_j$ is the average reward obtained from machine $j$, $n_j$ is the number of times machine $j$ has been played so far, and $n$ is the overall number of plays done so far.

Exploration bonus

From Auer et al., who show that UCB1 has regret **logarithmic in n**

# Thompson sampling

- For each arm, compute the probability that it is optimal given your current distribution over payoffs
- Pick an arm to play by sampling from this distribution

- Regret is logarithmic in sqrt(KT log T)

## Extensions of the bandit framework

- Many!

- Arguably most relevant to us is a contextual bandit:
  - Each bandit has a payoff that is dependent upon a context vector
  - Example: Customer profile
  - Context does not change as a function of choices (not an MDP)

## Conclusions

- Bandits are the gateway drug to MDPs
- Simplest case is essentially a single state

- Different views of optimality criteria lead to different algorithms