CMSC 858G: Bandits, Experts and Games Lecture #5 (Cot 3, 2016)

Instructor: Alex Slivkins

Discussion: practical aspects of IID bandits

Where we are:

- covered so far:
  - IID bandits: upper & lower bounds
  - Bayesian bandits via Thompson Sampling
  - 5 algorithms => 5 general techniques
  - basic & self-contained, except:
    - more complicated analysis for UCB1 and the sqrt{T} LB with better constants
    - proof for log(T) lower bound
    - log(T) upper bound for Thompson Sampling
- Coming up next
  - start with simple model, push in different directions
  - constrained function classes: Lipschitz, linear, convex
  - adversarial rewards (full feedback and bandit feedback)
  - contextual bandits
  - later: dynamic pricing (& similar problems),
    connections to game theory & mechanism design

Algorithms for IID bandits: practical performance

- Thompson Sampling is as good as anything else (and applied in practice)
- Doubling trick: bad in practice, blows up regret by constant factor
- UCB with decreased confidence radius $r\_t(a)$
  - log(T)->log(t) (also removes the need to know T)
  - plug in estimate of reward variance $Var(a)$
    - recall: small & known variance => can plug it into the confidence radius;
      instead, one can estimate the variance …
    - UCB-tuned: $r_t(a) = \sqrt{\frac{\ln t}{n_t(a)}\ \mathrm{V}_t^{UCB}(a)}$, where $\mathrm{V}_t^{UCB}(a)$ is approx. UCB on $Var(a)$
      From the original UCB1 paper. Good performance in simulations, no provable bounds.
    - UCB-V: $r_t(a) = \sqrt{\frac{2\ \ln t}{n_t(a)}\ V_t(a)} + \frac{c \log t}{n_t(a)}$, where $V_t(a)$ estimates $Var(a)$
      Use an estimate instead of UCB, but add a correction term.
      In a follow-up paper; comes with theoretical guarantees.
  - just replace $const \times \log t$ with 1! Worked pretty well in simulations (for some generalizations of IID bandits).
  - UCB2: $r_{t(a)} = \sqrt{\frac{\alpha \log(t/n_t(a))}{n_t(a)}}$, and each chosen arm is played $\beta\ n_t(a)$ times in a row
    From the original UCB1 paper; explains "1" in UCB1!
    … for carefully chosen constants α, β

- Successive Elimination is not good in practice:
  sampling uniformly from active arms suffices for theory, but you want to sample better arms more often
    - Successive Elimination with "better" arms selection: e.g., based on Thompson Sampling: published in follow-up work, works OK in simulations
- eps-greedy: (very) good in some regimes, but very sensitive to the choice of eps
- ϵ_t-greedy, $t = \min(1, \frac{cK}{dT})$ Needs $d \approx \min_{a:\Delta(a)>0} \Delta(a)$
  Performs very well with the right d.
  But very sensitive to the choice of constant c -- needs diff constants for diff instances!

Evaluation on simulated data *[more complicated than it seems]*

- Need to try many different "regimes" for reward function μ
    - 2 arms: {small/medium/large Δ} x {small/medium/large μ_1}
    - K arms: {fraction of good arms} x {# "types" of arms} x {all values shifted up/down}
    - deviations from IID - much more complicated, will discuss with adversarial rewards
- different "tunings" of the same algorithm
    - ideally, one tuning for all regimes
    - … but if we know smth about typical problem instances, may be ok to tune to the instance
- which time horizon do we care about? what if one algo is better initially but worse later?

Simulate on real data *[more complicated than it seems]*

- ideal: full feedback data -- but where do we get such data?
  Can take real data from different problems, and "fake" a bandit problem
    - multi-class classification datasets => contextual bandits with 0-1 rewards
        - omit contexts => fake an instance of IID bandits (with 0-1 rewards)
    - repeated auctions with bids => can use bids as customers' "private values" for dynamic pricing
        - caveat: in repeated auctions, one may have very different participants compared to dynamic pricing,
          so typical "private values" may be different
    - data from recommender systems
        - users and songs/restaurants/movies that they chose
        - data point = (user, <user features>, chosen item, <rating for this item>)
          reward = 1 or "rating" for all chosen items, 0 otherwise
          => can use it to create an instance of contextual bandits
        - caveat: data may depend heavily on the "menu" offered to the users, might not reflect their true preferences
- minimally: need enough samples from each arm to estimate the mean reward for this arm
    - good: probably ok to have less samples from bad arms
    - bad: only estimates, not the true values; inserts IID assumption into the data;
      (also, this approach is not suitable to simulate contextual bandits)
    - ugly: data collection needs to *explore*!
      so one needs to deploy [something like] a bandit algorithm just to collect the data.
- counterfactual evaluation:
    - what would have happened if you ran this algorithm when collecting the data?
    - again, data collection needs to explore

    - … and record the data very carefully, we'll discuss this more in the class on "contextual bandits".
- available datasets
    - multi-class classification – lots of publicly available datasets, widely used.
    - recommender systems: movies, songs, restaurants, shared bookmarks (some data available)
    - Yahoo news dataset: essentially, contextual bandits.
      Probably the only "real" bandit dataset available publicly.
    - medical trial data: lots of medical trials, a few are available publicly
      (… and I've seen a paper that simulates bandits on that data)

Real-world applications *[more complicated than it seems]*

- most common: A/B testing
    - essentially, Explore-First with uniform-at-random arms selection
    - pros: easier to implement in practice, easier to understand, does not rely on IID assumption
    - cons: inefficient like "Explore-First".
- Thompson Sampling:
    - published: at Microsoft (old version of the ad platform), Google Analytics (ad targeting)
    - rumors: Twitter, Criteo (ad targeting), Netflix, LinkedIn
- versions of UCB1 -- anecdotal evidence, at least for some small-scale deployments
- Contextual bandits: at Microsoft (MSN News, Bing, Ads), Yahoo (News, possibly also Ads), LinkedIn
- many deployments not publicized (trade secrets, engineers don't care to publish, afraid of bad PR)

Barriers for adoption: in practice, one might not have …

- … the right feedback:
    - might not know how to define "rewards"
    - rewards not always observed and/or arrive too late
- … the right algorithm [yet] (because the setting is more complicated than IID bandits)
- … the right infrastructure: it may be difficult to …
    - … insert a bandit algorithm into the existing system
    - … implement the logging in the right way
      (logging is mainly done for debugging and charging, and ML is an afterthought)
    - … have a sufficiently fast feedback loop
- … enough data points for bandits to make a difference
- … buy-in from management:
    - do we really need to explore? are we not afraid to explore?
    - why would it help to go beyond A/B testing?
    - inertia: why change? e.g., we already *have* A/B testing …
- … the manpower and/or expertise

For all these reasons, it helps to have "ML system" = {algorithms & infrastructure}, not just algorithms

    - Ideally: one system for many applications.
    - Multi-world Testing Decision Service: a system for contextual bandits developed at MSR-NYC