Stat 991 - Topics in Deep Learning - Lecture 1

Edgar Dobriban

University of Pennsylvania

January 24, 2019

Logistics

- See syllabus.

Overview

Sequential decision-making: from bandits to deep reinforcement learning

- Bandits
- Policies and regret analysis
- Comparing policies
- Adversarial bandits
- Software and Miscellanea

Setup

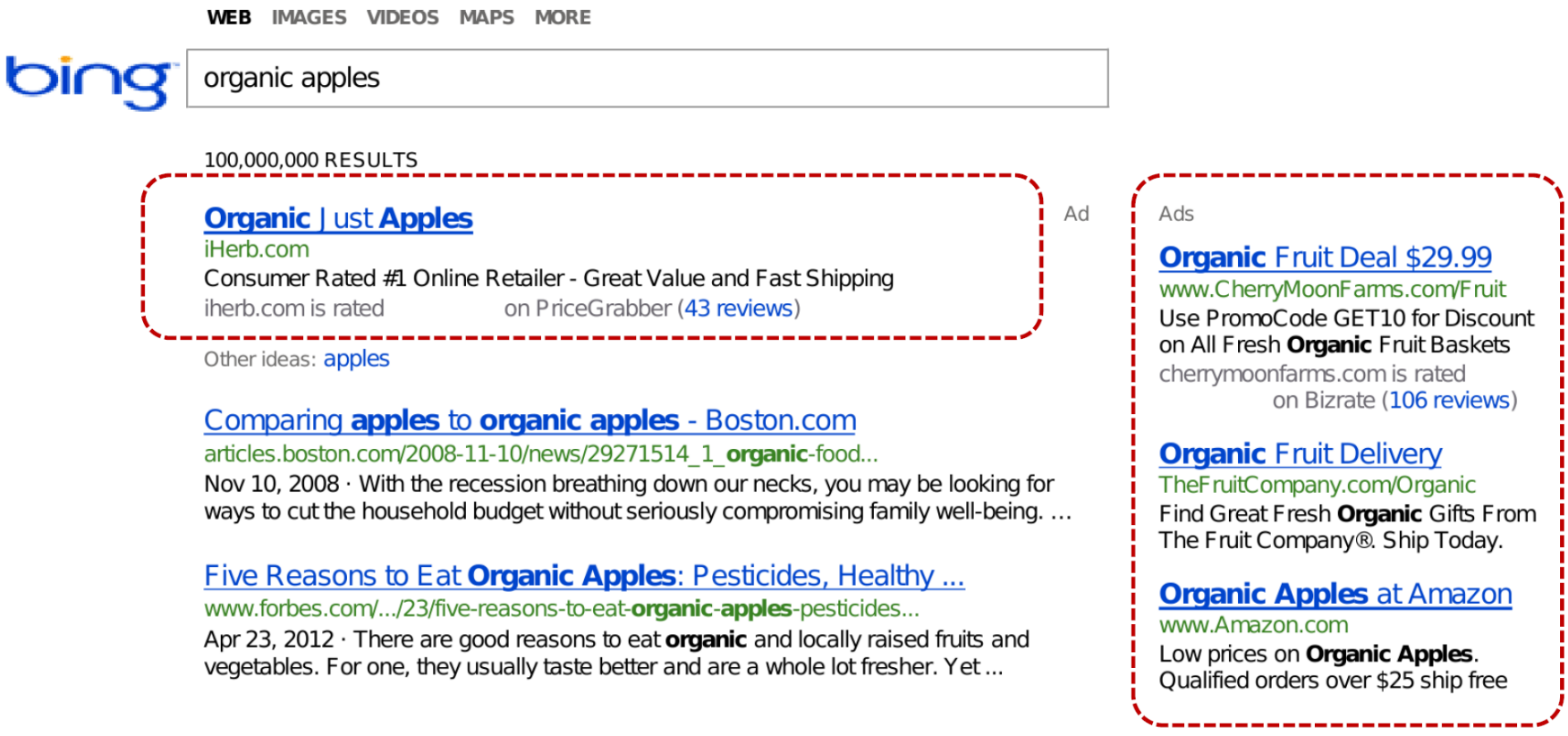
- Question: How to learn from experience that accumulates gradually over time?
- Specifically: How to make decisions?

Bandits - a clean model

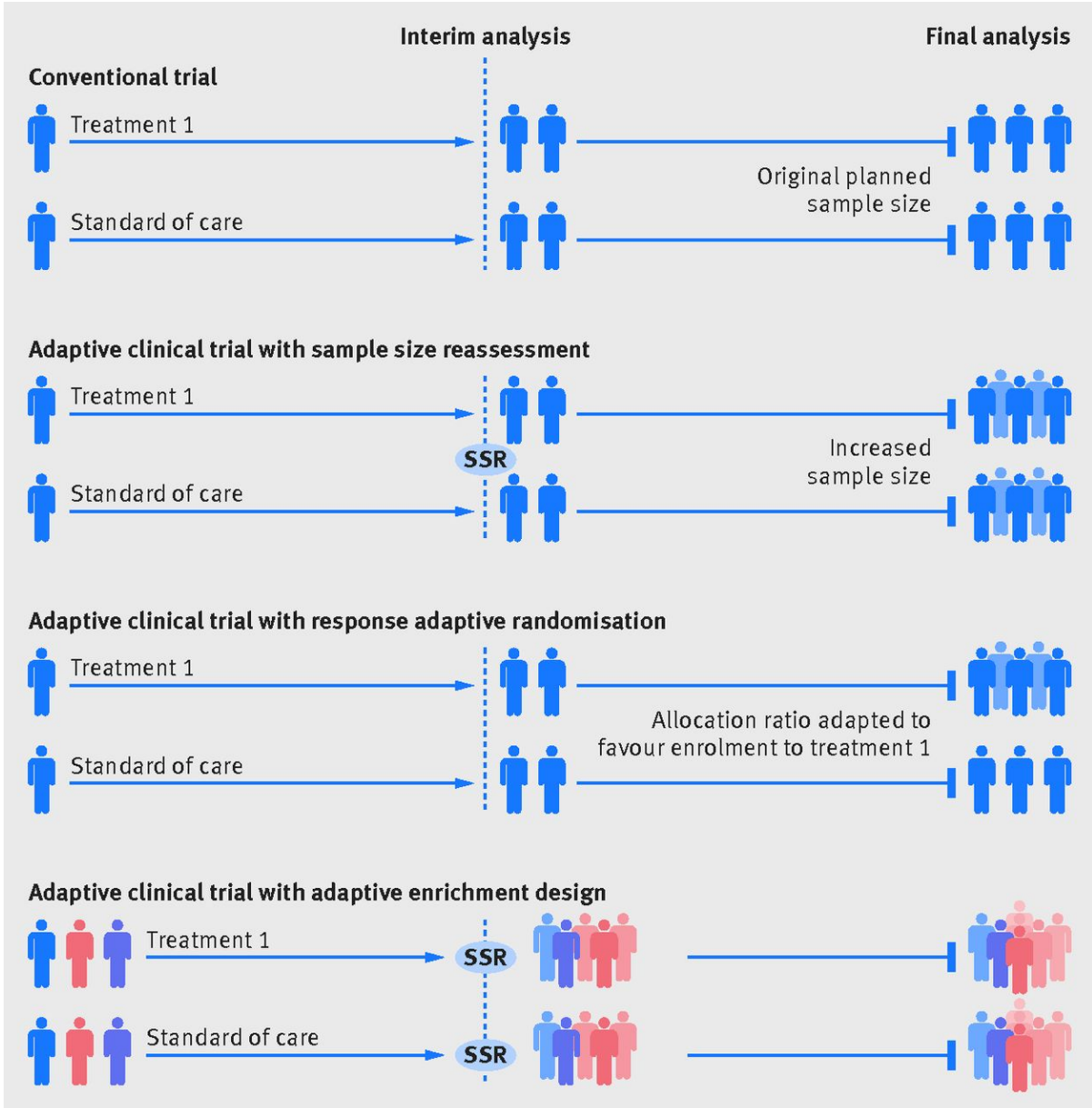
Time	1	2	3	4	5	6	7	8	9	10	11	12
Left arm	$1	$0			$1	$1	$0					
Right arm			$1	$0								

Five rounds to go. Which arm would you play next?

Online advertising



Adaptive clinical trials



Robotics

Examples

- Healthcare
  - Adaptive clinical trials, e.g., oncology
  - mHealth, e.g., smoking, stress, exercise
- Web
  - online ads, e.g., Microsoft, Google
  - dynamic pricing
  - A/B testing (different versions of website)
  - recommender systems: news, products

Examples

- Robotics
  - Planning, search, control
- Game playing
  - Classical games: Chess, Go
  - Computer games
- ...

Overview

Sequential decision-making: from bandits to deep reinforcement learning

- Bandits
- Policies and regret analysis
- Comparing policies
- Adversarial bandits
- Software and Miscellanea

Setup

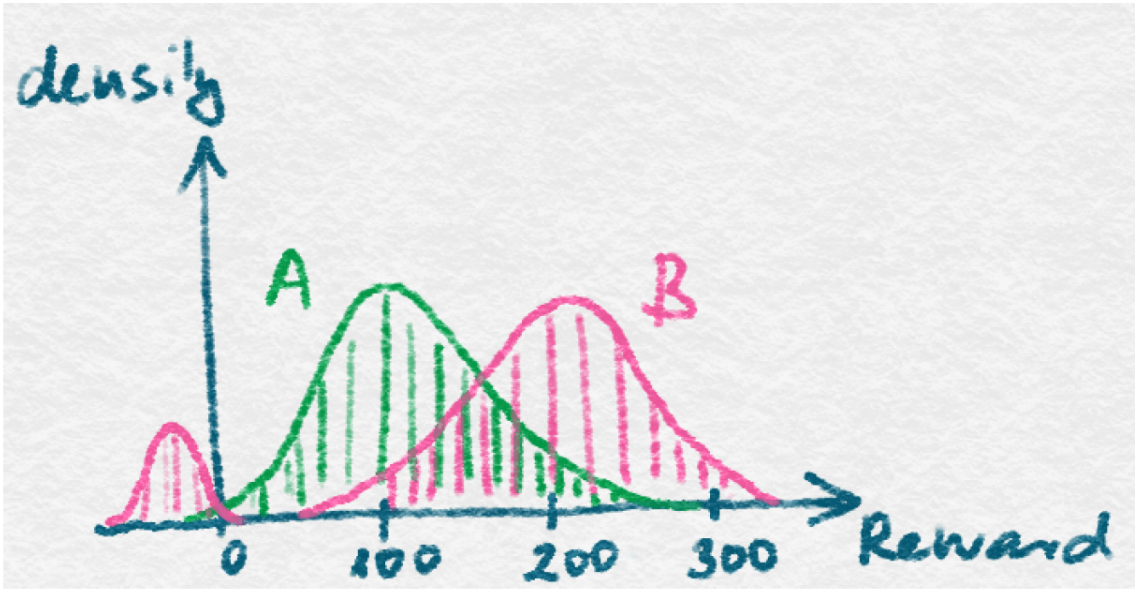
- *Time* $t = 1, 2, \ldots, T$
- Need to choose an *arm* $A_t \in 1, 2, \ldots, K$,
- Observe *reward* R_t, which depends on the arm chosen
- Goal: maximize *returns*

$$\max \sum_{t=1}^{T} R_t$$

Setup (2)

- Simplest setting: rewards are random, and distribution only depends on the arm. $R_t | A_t = k \sim P_k$ (independently of the past).
- known as *Stochastic bandits*
- At each timestep, we have available the entire *history* of previous actions and rewards $A_1, R_1, A_2, R_2, \ldots, A_{t-1}, R_{t-1}$
- Our *policy* (denoted π) chooses an action, in a possibly randomized way

Rewards



How to act? Any ideas?

- Recall: we have available the entire history of previous actions and rewards $A_1, R_1, A_2, R_2, \ldots, A_{t-1}, R_{t-1}$
- we need to choose arms in $1, \ldots, K$ to maximize sum of rewards. Rewards have a specific distribution for each arm.

Some possible policies

Some possible policies

- Initially explore all arms, then choose best (exploit) - (ETC)

Some possible policies

- Initially explore all arms, then choose best (exploit) - (ETC)
- Initially explore uniformly, then explore according to how promising the arms are - (Thompson sampling, Boltzmann/softmax, ε-greedy)

Some possible policies

- Initially explore all arms, then choose best (exploit) - (ETC)
- Initially explore uniformly, then explore according to how promising the arms are - (Thompson sampling, Boltzmann/softmax, ε-greedy)
- Estimate an upper confidence bound on the mean reward for each arm, and be greedy - UCB

Notes

- Balance *exploration* and *exploitation*
- Separating them is suboptimal
- In practice the other algorithms are quite similar (if tuned)

Measuring success–the notion of regret

- In the stochastic case, assume $R_t \in [0, 1]$ (or sub-Gaussian)
- Focus only on the *mean* rewards $\mu_k = \mathbb{E}_{P_k} R_t$
- An oracle policy would choose the best arm at each step:

$$k^* = \arg\max_k \mu_k$$

and $\mu_{k^*} = \mu^*$

- A more refined comparison between policies is given by the notion of *regret*: the difference between the optimal action and the given policy:

$$r_T = \max_k \mathbb{E}_{A_t=k} \sum_{t=1}^{T} R_t - \mathbb{E}_\pi \sum_{t=1}^{T} R_t$$

Or

$$r_T = T\mu^* - \mathbb{E}_\pi \sum_{t=1}^{T} R_t$$

Regret

- Regret $r_T = T\mu^* - \mathbb{E}_\pi \sum_{t=1}^{T} R_t$
- Minimizing regret is equivalent to maximizing return. But this is a more fine-grained measure, allows more careful comparison of policies
- Minimize regret - important life lesson :)

Overview

Sequential decision-making: from bandits to deep reinforcement learning

- Bandits
- Policies and regret analysis
- Comparing policies
- Adversarial bandits
- Software and Miscellanea

Explore then commit (ETC)

- ETC policy
  1. *Explore*: Choose each arm m times
  2. *Exploit*: Choose best arm afterwards (for $T - Km$ times)
- **Theorem**: Suppose $K = 2$, and let $\Delta = |\mu_1 - \mu_2|$. The regret of ETC is

$$r_T \leq m\Delta + T\Delta \exp\left(-\frac{m\Delta^2}{4}\right)$$

- Optimal $m = \lceil \frac{4}{\Delta^2} \log(\frac{T\Delta^2}{4}) \rceil$ - after this many steps, probability of error is small
- Achieves regret

$$r_T \leq \frac{4}{\Delta}\left[\log(\frac{T\Delta^2}{4}) + 1\right] + \Delta$$

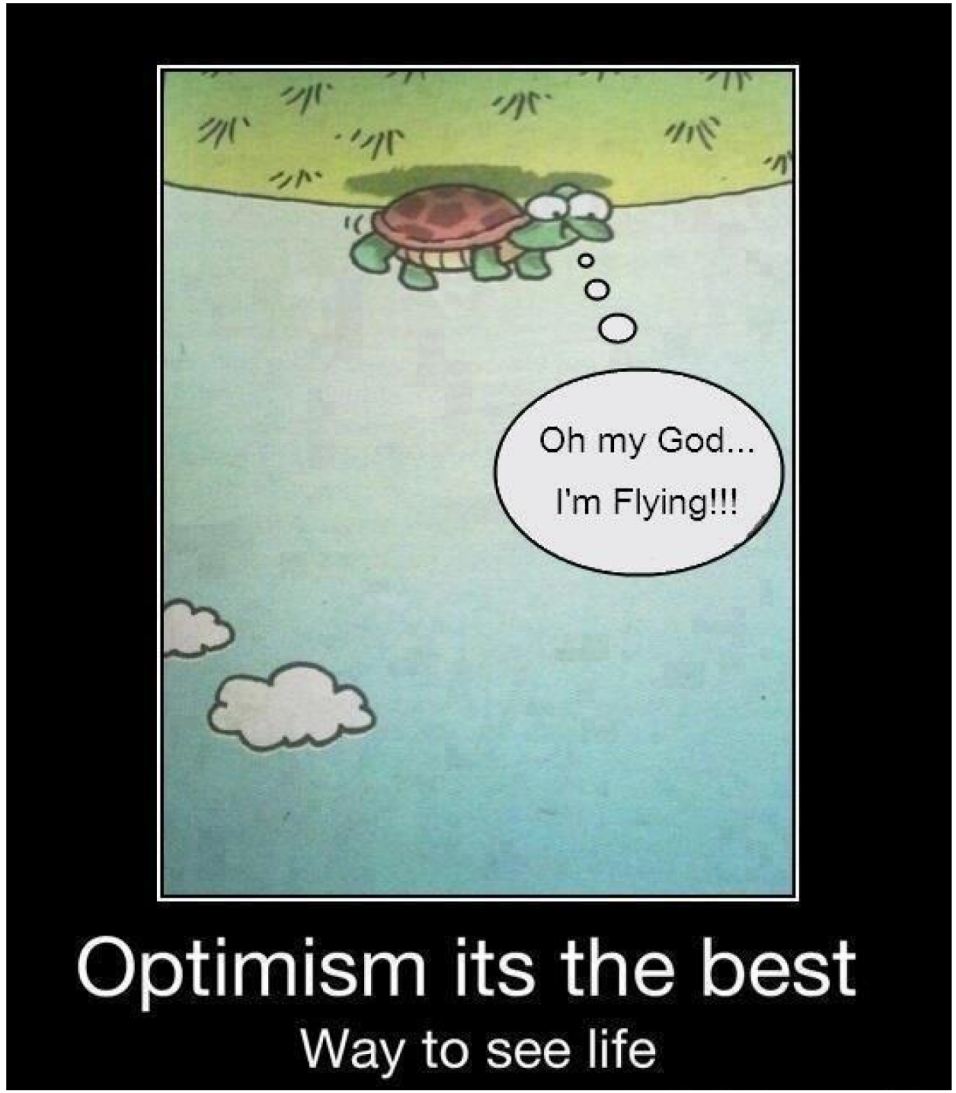
- Worst case $\Delta \sim T^{-1/2}$, worst case regret

$$r_T = O(T^{1/2})$$

Notes on ETC

- We say that ETC attains regret $T^{1/2}$
- Proof: See tutorial by Lattimore and Szepesvari
- Optimal up to order for $K = 2$ (but not the right constant)
- Issues
  - Non-adaptive: need to know T, Δ (hard for $K > 2$)
  - Spend a lot of time not learning

Optimism Principle



Optimism Principle - UCB Algorithm

- Upper Confidence Bound (UCB): Estimate an upper confidence bound on the mean reward for each arm, and be greedy
- Let $T_k(t)$ be the number of times action k was chosen until time t
- Define the *index* (or UCB) of arm k at time t:

$$\gamma_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{2\log(t^3)}{T_k(t)}}$$

- UCB algorithm:
  1. Choose each arm once
  2. Afterwards, at time $t+1$, choose arm maximizing index $\gamma_k(t)$

Analyzing UCB

- **Theorem**: Let $\Delta_i = \mu^* - \mu_i \geq 0$ be gap. The problem-dependent regret of UCB is

$$r_T = O\left(\sum_{i:\Delta_i>0} \Delta_i + \frac{\log T}{\Delta_i}\right)$$

- Worst case regret

$$r_T = O(\sqrt{KT \log T})$$

Notes on UCB

- May take different power $\log(t^c)$
- May take into account variance of rewards (UCB-Tuned)

ε-greedy

- ε-greedy: Let $\varepsilon > 0$ (say 0.05) be a small exploration parameter
  1. (Explore) With probability ε choose an action uniformly at random
  2. (Exploit) Wp $1 - \varepsilon$ choose best action
- Note: May decay $\varepsilon = \varepsilon_t \to 0$
- Initialize $\hat{\mu}_k(0) = 0$ (or 1 - optimistic initial values)

Boltzmann sampling / softmax

- Boltzmann sampling: Let $\tau > 0$ (say 0.1) be a temperature parameter
  1. Initialize $\hat{\mu}_k(0) = 0$ (or 1)
  2. At time $t+1$, select action k with probability proportional to

$$P(A_t = k) \sim \exp\left[\frac{\hat{\mu}_k(t)}{\tau}\right]$$

  3. Update $\hat{\mu}_k(t+1)$

Thompson sampling

- Thompson sampling is a Bayesian method. Put priors on each arm's reward distribution.
  1. At time t, select action k with probability (assuming continuous reward distributions)
     $$P(A_t = k) = P(R_k > R_j, \forall j \neq k)$$
     where $P()$ is the posterior
  2. Update posterior of chosen action
- Choose arm with probability that it is best

Thompson sampling

- Thompson sampling for Bernoulli bandits is convenient because of conjugate priors.
- Suppose $P_k \sim Bernoulli(\mu_k)$
- Put prior $\mu_k \sim Beta(a, b)$ (eg 1,1)
- If we choose action k, and sample $R_1 = 1$, the posterior becomes $\mu_k \sim Beta(a + 1, b)$
- More generally, after t steps, let S_t be the number of successes by pulling arm t, and let F_t be the number of failuers. Then the posterior is

$$\mu_k \sim Beta(a + S_t, b + F_t)$$

Overview

Sequential decision-making: from bandits to deep reinforcement learning

- Bandits
- Policies and regret analysis
- Comparing policies
- Adversarial bandits
- Software and Miscellanea

Which policy to use

- Saw: UCB, ε-greedy, Boltzmann/softmax, Thompson sampling
- Which policy to use?
- In practice they are quite similar (if tuned)

Experiments

- Experimental evaluation of several algorithms performed by Velmorel, Mohri (2005); Kuleshov, Precup (2014)
- Protocol from Kuleshov, Precup (2014)
  - $K = 2, 5, 10, 50$, $T = 1000$ (plateau afterwards),
  - Distributions normal $P_k \sim \mathcal{N}(\mu_k, \sigma^2)$; μ_k sampled from $U[0, 1]$; $\sigma^2 = 0.01, 0.1, 1$
  - Also try other distributions P_k (only mean, variance matters)
  - Policies: UCB, UCB-Tuned, ε-greedy, Boltzmann/softmax, Pursuit, Reinforcement comparison (see book by Sutton-Barto)
  - Tune hyperparameters optimally [In practice, how??]

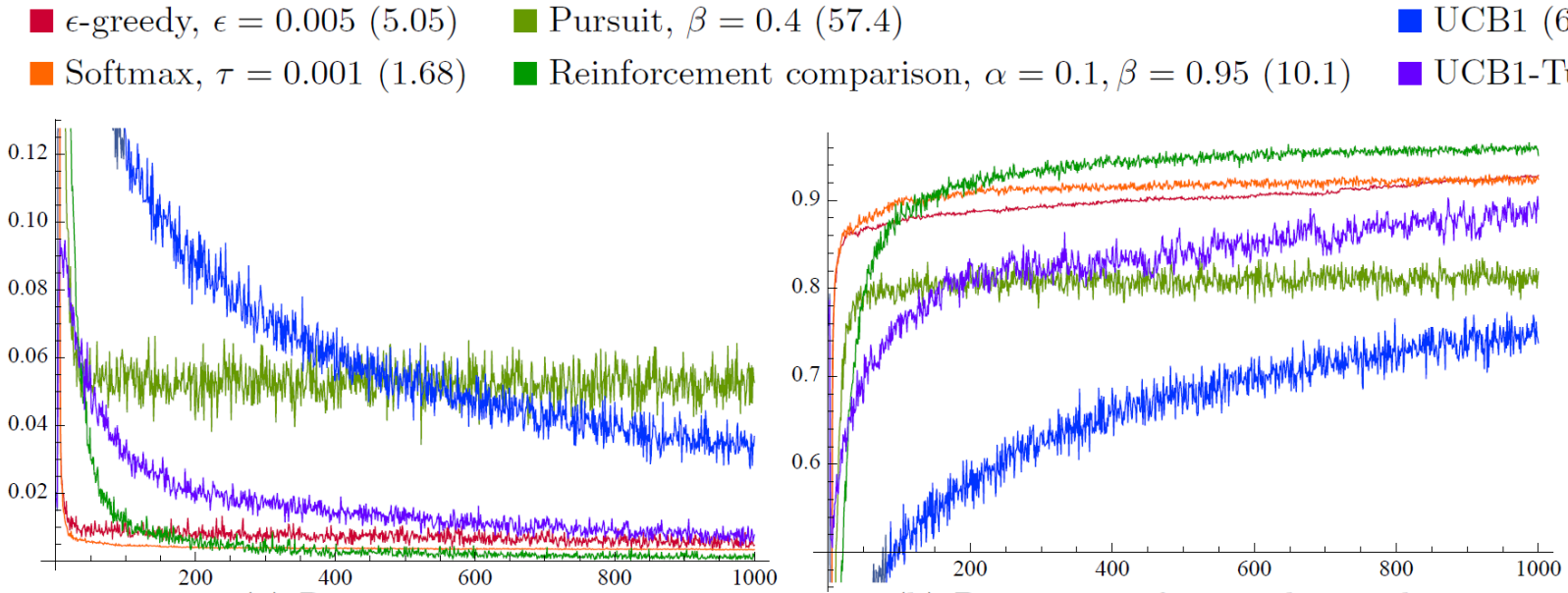
Experiments

- Kuleshov & Precup (2014)

  The most striking observation is that the simplest algorithms, ε-greedy and Boltzmann exploration, outperform their competitors on almost all tasks. Both heuristics perform very similarly, with softmax usually being slightly better.

K=5

$\sigma = 0.01$



(a) Regret per turn

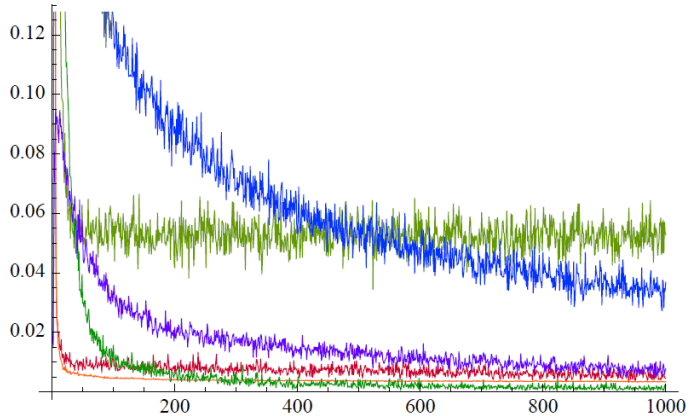
(b) Percentage of optimal arm plays

K=5

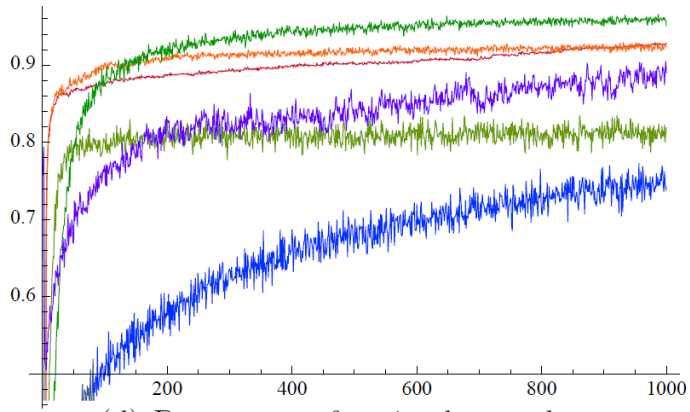
$\sigma = 0.1$

ϵ-greedy, $\epsilon = 0.001$ (8.93) — Pursuit, $\beta = 0.25$ (55.8) — UCB1 (67.4)

Softmax, $\tau = 0.01$ (5.63) — Reinforcement comparison, $\alpha = 0.1, \beta = 0.95$ (10.3) — UCB1-Tuned (17.5)



(c) Regret per turn

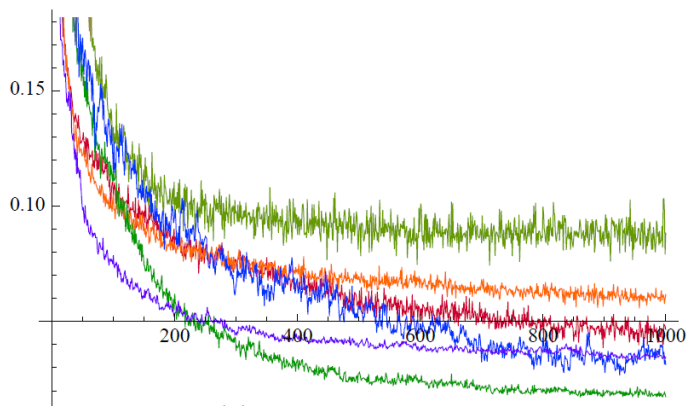


(d) Percentage of optimal arm plays

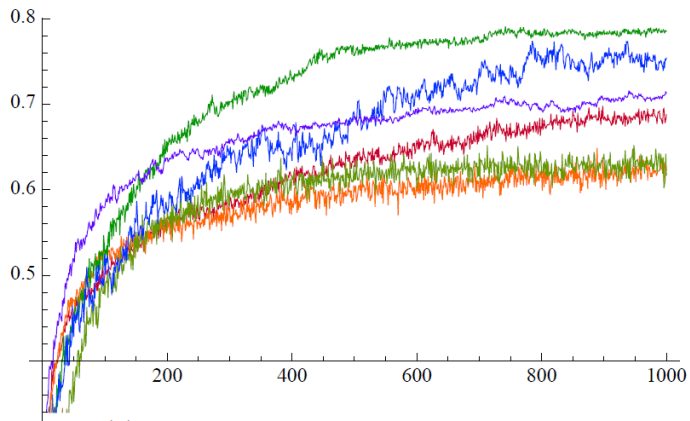
$\sigma = 1$

ϵ-greedy, $\epsilon = 0.05$ (71.192) — Pursuit, $\beta = 0.05$ (106) — UCB1 (69.5)

Softmax, $\tau = 0.1$ (77.4) — Reinforcement comparison, $\alpha = 0.01, \beta = 0.9$ (46.9) — UCB1-Tuned (50.7)



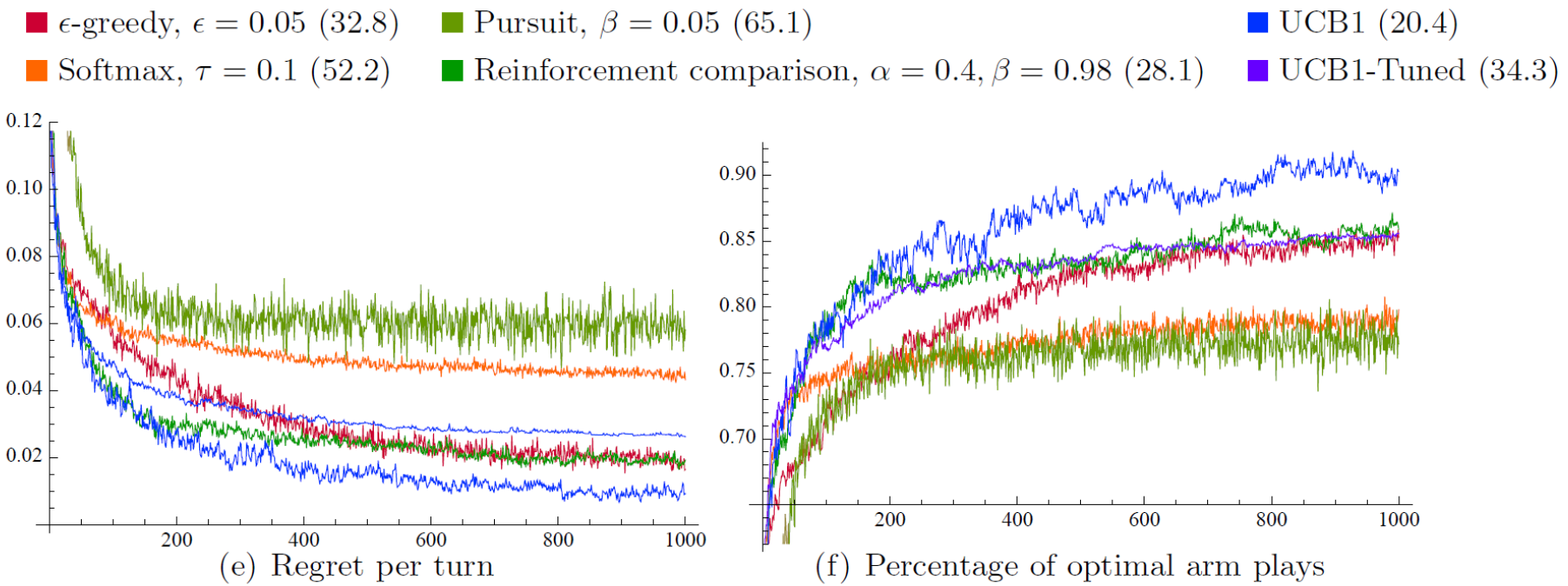
(e) Regret per turn



(f) Percentage of optimal arm plays

K=2

$\sigma = 1$



(e) Regret per turn

(f) Percentage of optimal arm plays

Figure 1: Empirical Results for 2 arms, with different values of the variance

Experiments

- Kuleshov & Precup (2014)

  In particular, softmax outperforms the other algorithms in terms of total regret on all tasks, except on the high-variance setting ($\sigma = 1$) for small and medium numbers of arms (K = 2,5,10). On these settings, softmax comes in second behind the UCB1-Tuned algorithm. In some sense, the fact that UCB1-Tuned was specifically designed to be sensitive to the variance of the arms justifies the fact that it should be superior at high values of the variance.

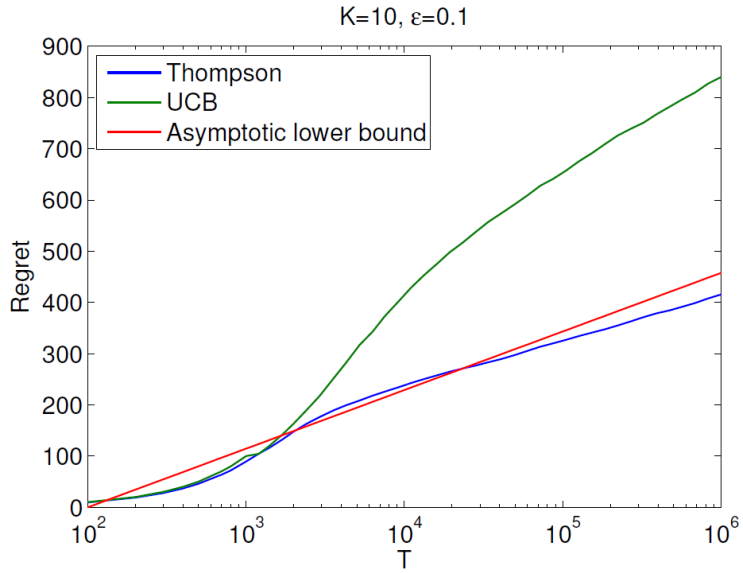
- Softmax: good for low variance
- UCB: Good for high variance
- But not that conclusive, and how to tune?

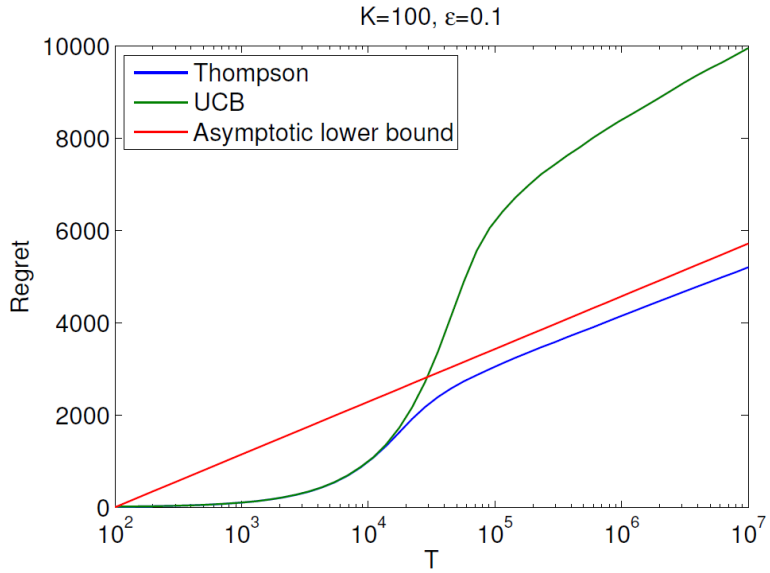
Thompson sampling?

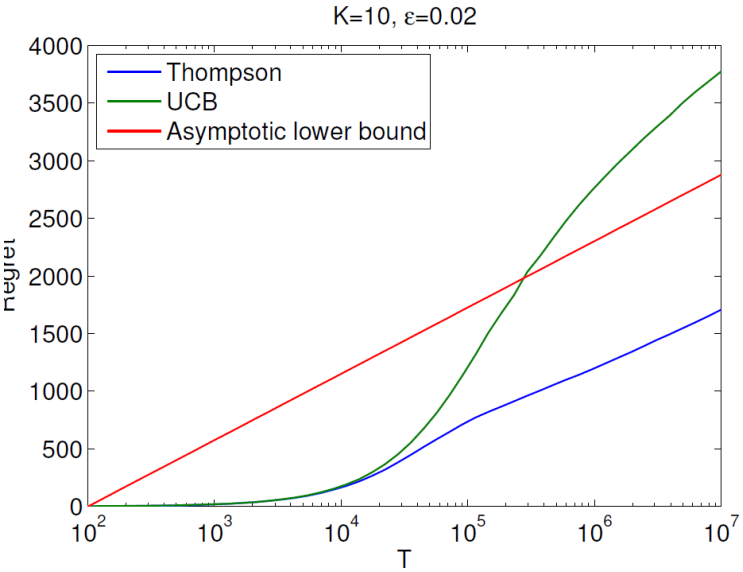
- Empirical evaluation by Chapelle and Li (2011)
- Thompson sampling works better than UCB
- Protocol: K Bernoulli arms, one with $\mu_1 = 1/2$, rest with $\mu_i = 1/2 - \varepsilon$
- Prior: $\mu_i \sim Beta(1, 1)$
- Lower bound from Lai and Robbins (1985)

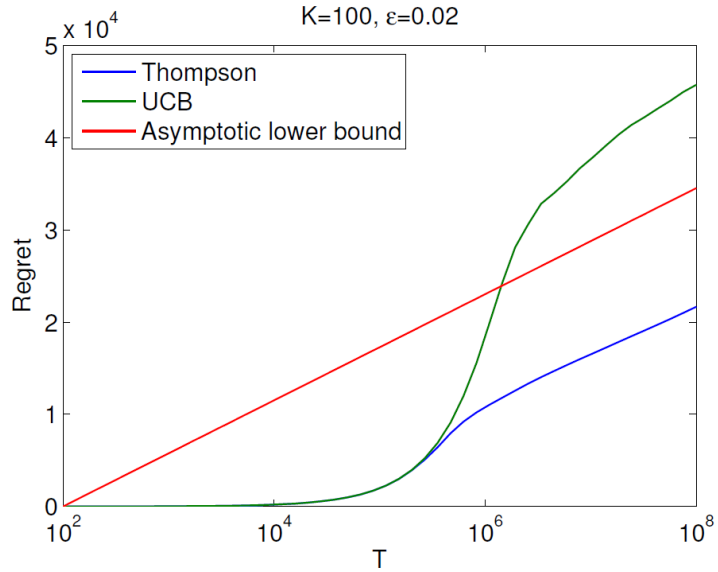
$$r_T \geq \log(T) \cdot \left[\sum_{i=1}^{K} \frac{\Delta_i}{D_{KL}(\mu_i \| \mu^*)} + o(1) \right]$$

Thompson sampling









Experiments

- Chapelle and Li (2011)

  We can thus conclude that in these simulations, Thompson sampling is asymptotically optimal and achieves a smaller regret than the popular UCB algorithm. It is important to note that for UCB, the confidence bound is tight; we have tried some other confidence bounds, including the one originally proposed in [3], but they resulted in larger regrets.

- Theory: TS is asy optimal, attains Lai-Robbins lower bound for Bernoulli bandits and several priors (Agrawal and Goyal, 2012; Agrawal and Goyal, 2013a; Kauffmann et al., 2012).
- In some cases worst-case regret $\sqrt{KT \log(T)}$
- See tutorial by Russo, Van Roy et al, 2018

Overview

Sequential decision-making: from bandits to deep reinforcement learning

- Bandits
- Policies and regret analysis
- Comparing policies
- Adversarial bandits
- Software and Miscellanea

Adversarial bandits

- At the beginning, adversary secretly chooses fixed rewards $R_1, R_2, \ldots, R_T$, say each in $[0,1]^K$. So $R_1(k)$ is the reward for choosing action k at the first time step
- As before: We choose actions A_t based on history, and obtain reward $R_t(A_t)$
- Hard: the adversary may choose rewards without any structure, or to "fool" us
- Regret of policy π is

$$r_T = \max_k \sum_{t=1}^{T} R_t(k) - \mathbb{E}_\pi \sum_{t=1}^{T} R_t(A_t)$$

Adversarial bandits - Example

- Your friend secretly chooses rewards $x_1 \in \{0, 1\}$ and $x_2 \in \{0, 1\}$.
- You implement a strategy to select $A \in \{1, 2\}$ and receive reward x_A.
- The regret is $R = \max\{x_1, x_2\} - \mathbb{E}x_A$.

Adversarial bandits - policies

- Any suggestions on strategies?
- The rewards are non-random, can be arbitrary, adversarial and may fool us

Adversarial bandits

- How can we avoid being fooled?
- If we consider a deterministic strategy, there is always an adversarial example that has maximal regret T
- So we need to consider randomized strategies
- A uniform random strategy already guarantees regret $T/2$. Can we do better?
- Yes - there are policies with regret $O(\sqrt{KT})$

Boltzmann sampling aka Exp3 - Exponential-weights for Exploration and Exploitation

- Let $\tau > 0$ (say 0.1) be a temperature parameter
  1. Initialize $\hat{\mu}_k(0) = 0$ (or 1)
  2. At time $t+1$, select action k with probability proportional to

$$P(A_t = k) \sim \exp\left[\frac{\hat{\mu}_k(t)}{\tau}\right]$$

  3. Update $\hat{\mu}_k(t+1) = [k\hat{\mu}_k(t) + \hat{R}_t(k)]/(k+1)$, for some unbiased estimator $\hat{R}_t$ of R_t

Estimators of reward

- Key intermediate step: Find unbiased estimator $\hat{R}_t$ of R_t. Problem: we do not observe R_t
- Denote P_t the probability distribution over actions at time t
- *Importance weighted* estimator $\hat{R}_t$, has zeros for actions not taken, and

$$\hat{R}_t(k) = \frac{I(A_t = k)R_t(k)}{P_t(k)}$$

- We see $\mathbb{E}[\hat{R}_t(k)|A_1, R_1, \ldots, A_{t-1}, R_{t-1}] = R_t(k)$
- Loss-based importance-weighted estimator: preferable because takes values in $(-\infty, 1]$:

$$\hat{R}_t(k) = 1 - \frac{I(A_t = k)}{P_t(k)}(1 - R_t(k))$$

Regret of Exp3

- Theorem: Let $\tau = \sqrt{K/(T \log K)}$. Then the regret of the Exp3 algorithm with the loss-based weights for adversarial bandits is

$$r_T \leq \sqrt{2TK \log(K)}$$

A different perspective

- Regret is (S^k is simplex)

$$r_T = \max_k \mathbb{E}\left[\sum_{t=1}^{T} R_t(k) - R_t(A_t)\right]$$
$$= \max_{p \in S^k} \mathbb{E}\left[\sum_{t=1}^{T} p^\mathsf{T} R_t - P_t^\mathsf{T} R_t\right]$$

- because for a vector U, we have $\max_k U(k) = \max_{p \in S^k} p^\mathsf{T} U$, and we use this for $U = \sum_{t=1}^{T} R_t$
- recall we denote P_t the probability distribution over actions at time t
- *Online linear optimization on the simplex*

Online linear optimization on the simplex

- Find distributions P_t sequentially, such that the following is small

$$r_T = \max_{p \in S^k} \mathbb{E}\left[\sum_{t=1}^{T} (p - P_t)^\mathsf{T} R_t\right]$$

- Online linear optimization on the simplex, except R_t is not observed
- For unbiased estimator $\hat{R}_t$, we see $\mathbb{E}[\hat{R}_t(k) | A_1, R_1, \ldots, A_{t-1}, R_{t-1}] = R_t(k)$, so

$$r_T = \max_{p \in S^k} \mathbb{E}\left[\sum_{t=1}^{T} (p - P_t)^\mathsf{T} R_t\right] = \max_{p \in S^k} \mathbb{E}\left[\sum_{t=1}^{T} (p - P_t)^\mathsf{T} \hat{R}_t\right]$$

Algorithm

- *Follow the regularized leader*: for each $t = 1, \ldots,$ and for a fixed learning rate η, define P_t as

$$P_t = \arg\max_p \eta \cdot p^{\mathsf{T}} \sum_{s=1}^{t-1} \hat{R}_s + F(p)$$

- F is a regularizer, eg entropy $F(p) = -\sum_i p_i \log(p_i)$
- Theorem: $r_T \leq \sqrt{2TK \log(K)}$

Notes

- There is a more general picture: Online lin optimization. Allow choosing probability distributions. Two main algos: Mirror descent + Follow the regularized leader
- Exponential weights algorithm is a special case of mirror descent (constraint set A: simplex, F negentropy)
- In some cases the two algorithms coincide. (see Lattimore and Szepesvari for more)

Notes

- Adversarial bandits are in general challenging than stochastic bandits – but in this case they are equally difficult
- Variants: reactive adversary

Overview

Sequential decision-making: from bandits to deep reinforcement learning

- Bandits
- Policies and regret analysis
- Comparing policies
- Adversarial bandits
- Software and Miscellanea

Software

- Not that widely implemented
- Links to a few software packages (in Python) have been provided in the syllabus
- Experimental evaluations on real data tricky - how do we find the reward for the actions not taken?

Terminology

- Sequential experimental design
- Online learning
- Active/Interactive learning
- Reinforcement learning