A Tutorial on Thompson Sampling

Zheng Wen
Research scientist, Google DeepMind
zhengwen@google.com

MLRS2023, Bangkok, Thailand

August 8, 2023

Goal and Caveats

- **Goal:** discuss the most important insights about Thompson sampling (TS) and data-efficient reinforcement learning
- by “most important”, it means
  - it will not cover all about TS
  - it will reflect my understanding about what are important
- by “insights”, it means
  - some of the discussion will be a little bit high-level and handwavy
  - more rigorous discussions/analyses exist in literature

Outline

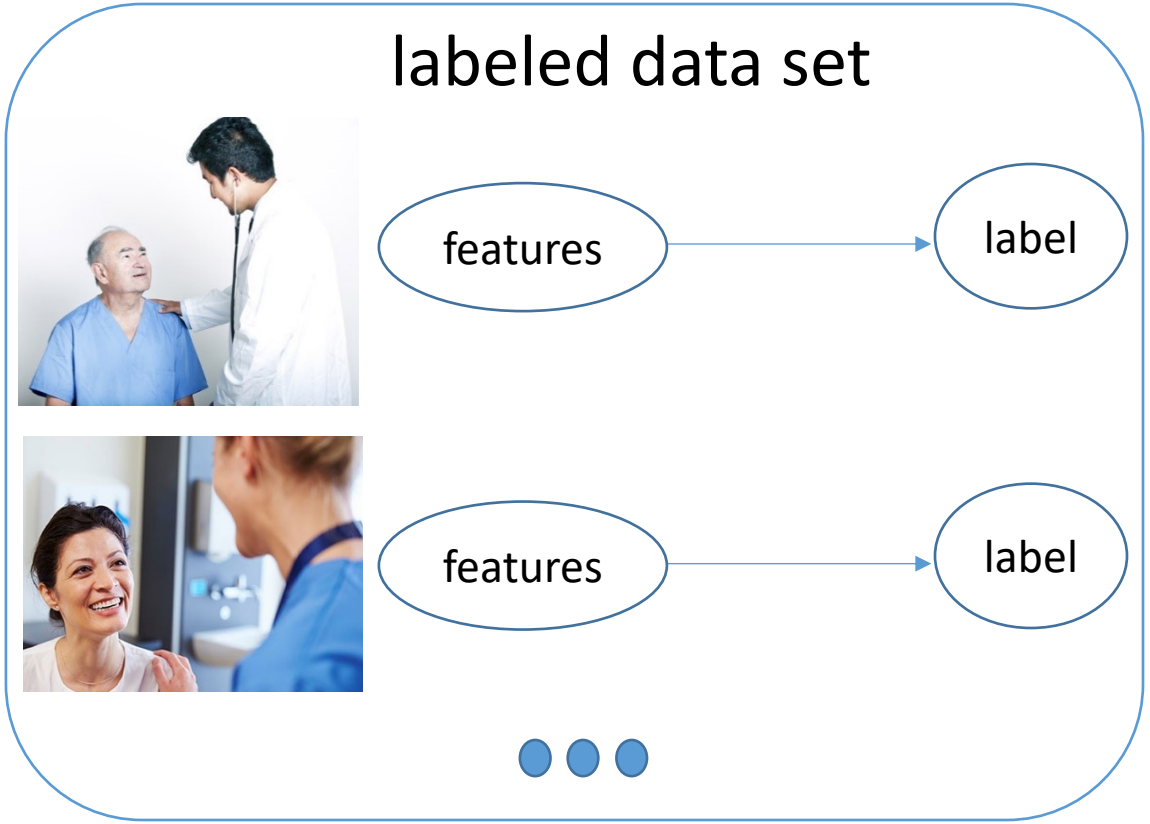
1. **reinforcement learning and contextual bandits**
2. Thompson sampling (TS): algorithm and practical considerations
3. analysis techniques of TS: a high-level overview
4. discussion and some advanced research topics

Machine Learning Paradigms

- three classical **machine learning paradigms:**

Supervised learning	Unsupervised learning	Reinforcement learning
learn from a labeled dataset	learn from an unlabeled dataset	learn while interacting with an environment

Supervised Learning

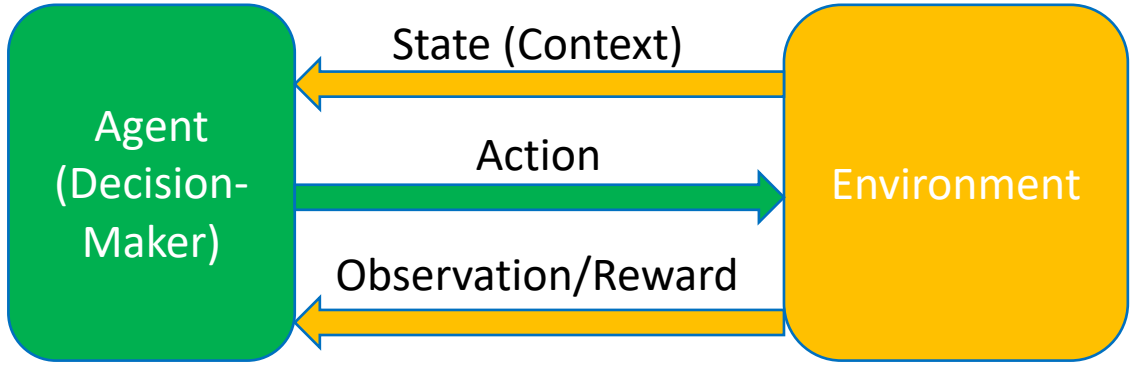


Main Challenges

Generalization

Reinforcement Learning

- **reinforcement learning (RL):** learning to optimize while interacting with an unknown (dynamic) environment
  - actions influence rewards, state transitions, and the agent's observations
  - a standard model for the environment: Markov decision process (MDP)
    - but we might also consider more general models



Reinforcement Learning

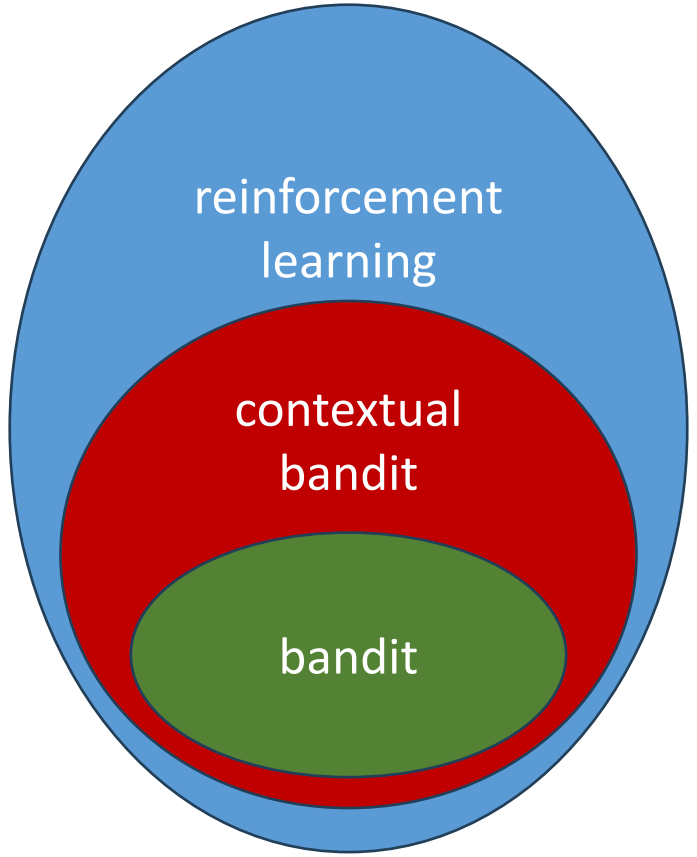
- **what is new in RL?**
  - **exploration-exploitation tradeoff:**
    - since actions influence both reward and observations
    - the agent can choose an action to maximize its immediate reward (exploitation)
    - or can choose an action to gain some useful information (exploration)
  - **delayed consequences:**
    - since actions might influence state transitions
    - the agent can plan to receive a big reward later, while sacrificing its immediate rewards

Alternative Names

- variants of RL problems have been studied in many related fields
  - different names have been proposed
  - “reinforcement learning” is the name used in the AI/ML community
  - motivated by the reinforcement theory in psychology
- in operations research community:
  - RL = approximate dynamic programming with unknown (MDP) models
  - a.k.a neuro-dynamic programming or data-driven dynamic programming
- in control theory community:
  - some RL problems are referred to as adaptive control problems

Bandits and Contextual Bandits

- **contextual bandits:** RL problems in which actions do not influence state transitions
  - e.g. the state (context) follows a Markov chain
  - no delayed consequences
  - learn to optimize immediate rewards at each state (context)
  - still has exploration-exploitation tradeoff
- **bandits:**
  - RL problems with only one state
  - learn to choose an optimal action



Contextual Bandits

features → treatment → reward

features → treatment → reward

...

Main Challenges

Generalization+ Exploration

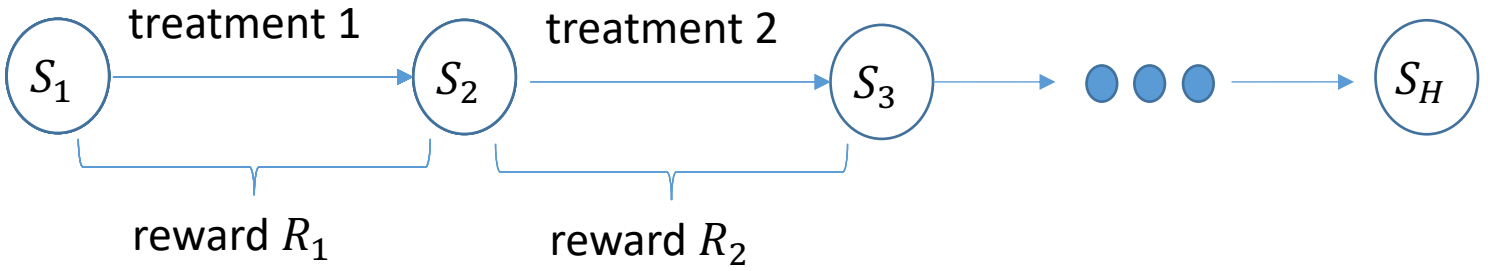
Contextual Bandits

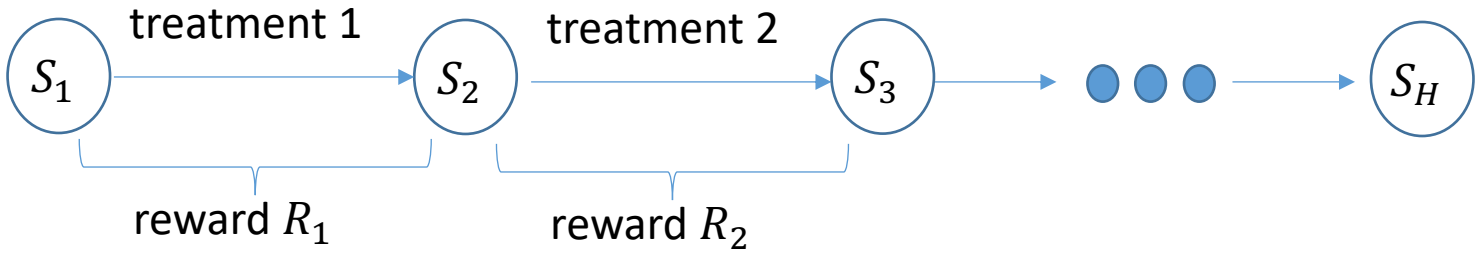
features → treatment → reward

features → treatment → reward

- generalization is still a challenge
- the observation at one patient-treatment (state-action) pair might provide useful information for
  - treatment effect of the same treatment for a similar patient
  - treatment effect of a similar treatment for the same patient

Reinforcement Learning





Main Challenges

Generalization + Exploration

+ Delayed Consequences

can we learn an optimal adaptive treatment plan?

Exploration-Exploitation Tradeoff

- exploration-exploitation tradeoff plays a key role in RL agent design
  - under-exploration: agent can get stuck at sub-optimal policies
  - over-exploration: incur huge exploration cost
- many exploration schemes have been proposed:
  - epsilon-greedy
  - Boltzmann exploration
  - upper confidence bound (UCB)
  - Thompson sampling (TS)
  - information directed sampling (IDS)
  - …

Bernoulli Bandit

- let's use Bernoulli bandit to illustrate the exploration schemes
- **Bernoulli bandit with *K* arms (actions):**
  - each action a has a mean reward $r(a) \in [0, 1]$
  - at each time t, if the agent chooses action A_t, then it will observe and receive a binary reward $R_{t+1} \sim \text{Bern}\big(r(A_t)\big)$. Notice that $R_{t+1} \in \{0, 1\}$.
- the agent does not know $r(a)$
  - might have a prior over it
- **Goal:** maximize the expected total reward in T time steps

Greedy Algorithms

- a simple and natural algorithm for online decision-making
- at each time t
  - estimate a reward model $\bar{r}_t(a)$ from historical data
    - can be any reasonable point estimate
  - select an action that is optimal under the estimated reward model

$$A_t \in \operatorname{argmax}_a \bar{r}_t(a)$$

- "greedy" in the sense that it tries to maximize the immediate reward
- "exploitation-only"
- can easily get stuck at sub-optimal actions if unlucky

Epsilon-Greedy Exploration

- an improvement of the greedy algorithm
- at each time t
  - estimate a reward model $\bar{r}_t(a)$ from historical data
  - with prob. ε, sample A_t uniformly randomly from the K actions
  - with prob. $1-\varepsilon$, select $A_t \in \operatorname{argmax}_a \bar{r}_t(a)$
- ε controls the "level of exploration"
- can learn the optimal action as $t \to \infty$
- tend to be data inefficient since it tries actions that have no chance to be optimal

Boltzmann Exploration

- another exploration scheme based on the point estimate
- at each time t
  - estimate a reward model $\bar{r}_t(a)$ from historical data
  - select action $A_t = a$ with probability

$$P(A_t = a) = \frac{\exp(\bar{r}_t(a)/\eta)}{\sum_{a'} \exp(\bar{r}_t(a')/\eta)}$$

- η is the "temperature" and controls the level of exploration
- can learn the optimal action as $t \to \infty$
- also tend to be data inefficient since it is based on point estimates
  - it does not take uncertainty into consideration

Upper-Confidence Bound (UCB)

- an exploration scheme considering uncertainty
- at each time t
  - estimate a confidence interval $[l_t(a), u_t(a)]$ for $r(a)$ from historical data
  - choose action $A_t \in \operatorname{argmax}_a u_t(a)$
- considers uncertainty
- is data efficient, i.e. learns the optimal action quickly
  - if the confidence intervals are constructed correctly
  - usually, the confidence intervals are constructed based on some concentration inequalities (e.g. Hoeffding's inequality)

Upper-Confidence Bound (UCB)

- why does UCB work?
- intuitively,

  UCB = (point estimate) + (confidence interval size)

- UCB is one index balancing exploration and exploitation
- an action with large UCB:
  - either you believe the action has large reward (exploitation)
  - or you are uncertain about the action's reward (exploration)
- **"optimistic in the face of uncertainty"**

Upper-Confidence Bound (UCB)

- for settings with generalization, UCB requires to construct a confidence set rather than a confidence interval for each action
- why not use UCB?
  - it has been widely used
  - researchers have observed that Thompson sampling tends to perform better in most practical problems
- why TS tends to perform better?
  - some theory has been developed
  - **Intuition:** the performance of UCB highly depends on how the confidence interval/set is constructed
  - in many cases, the constructed confidence interval/set tends to be too conservative

Outline

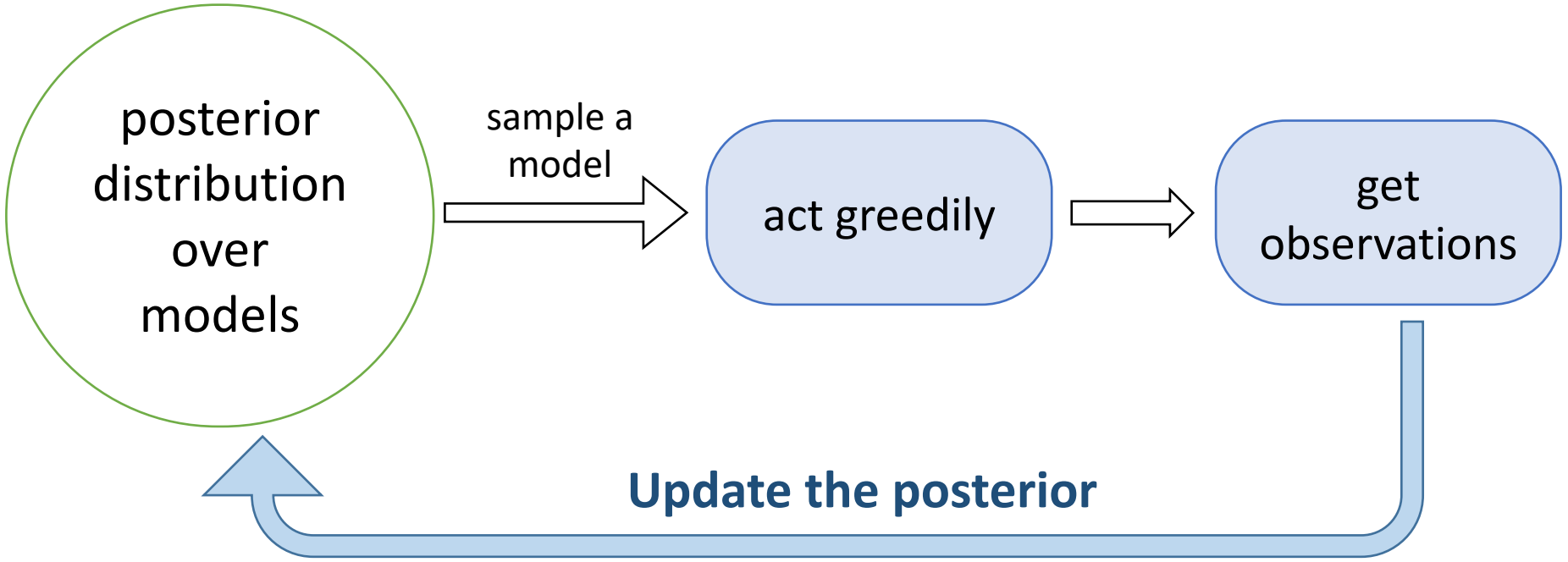
1. reinforcement learning and contextual bandits
2. **Thompson sampling (TS): algorithm and practical considerations**
3. analysis techniques of TS: a high-level overview
4. discussion and some advanced research topics

Why Thompson Sampling (TS)?

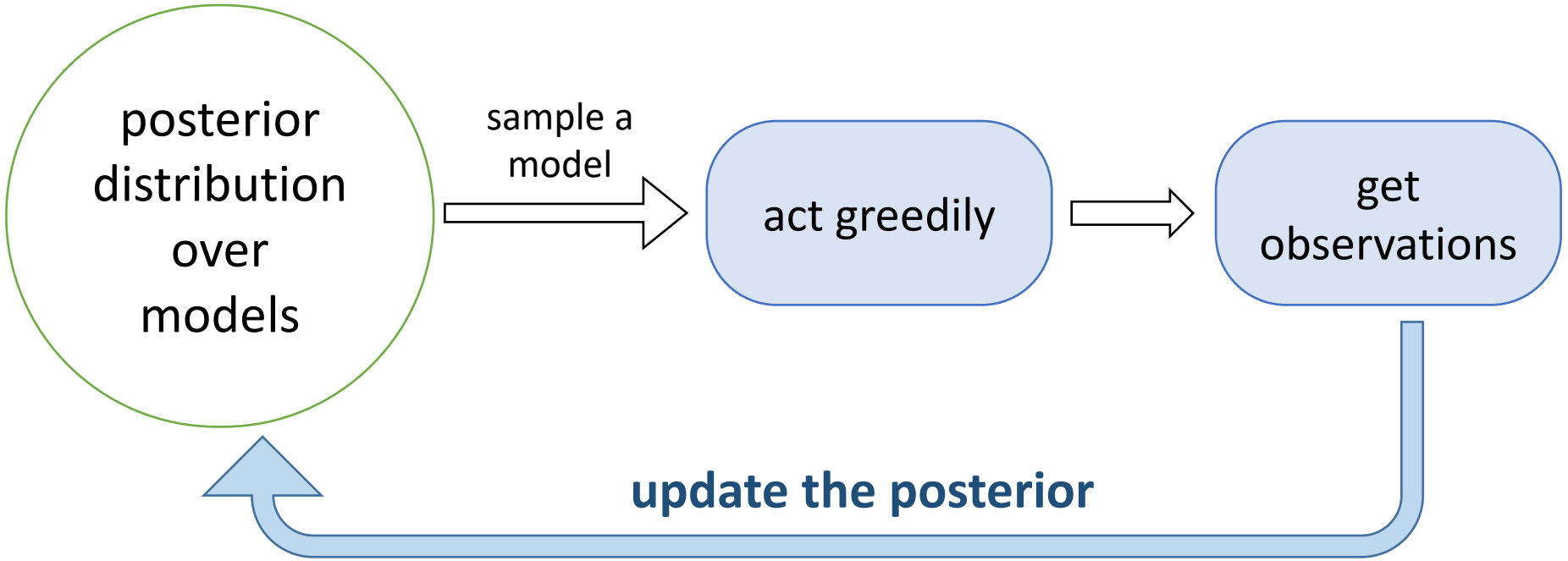
in most practical problems, it is observed that Thompson sampling significantly outperforms other exploration schemes, such as epsilon-greedy and UCB, assuming that similar efforts have been spent on fine-tuning the algorithms.

Thompson Sampling (TS)

- maintains a probability distribution over models and selects actions according to their probabilities of being optimal



Thompson Sampling (TS)



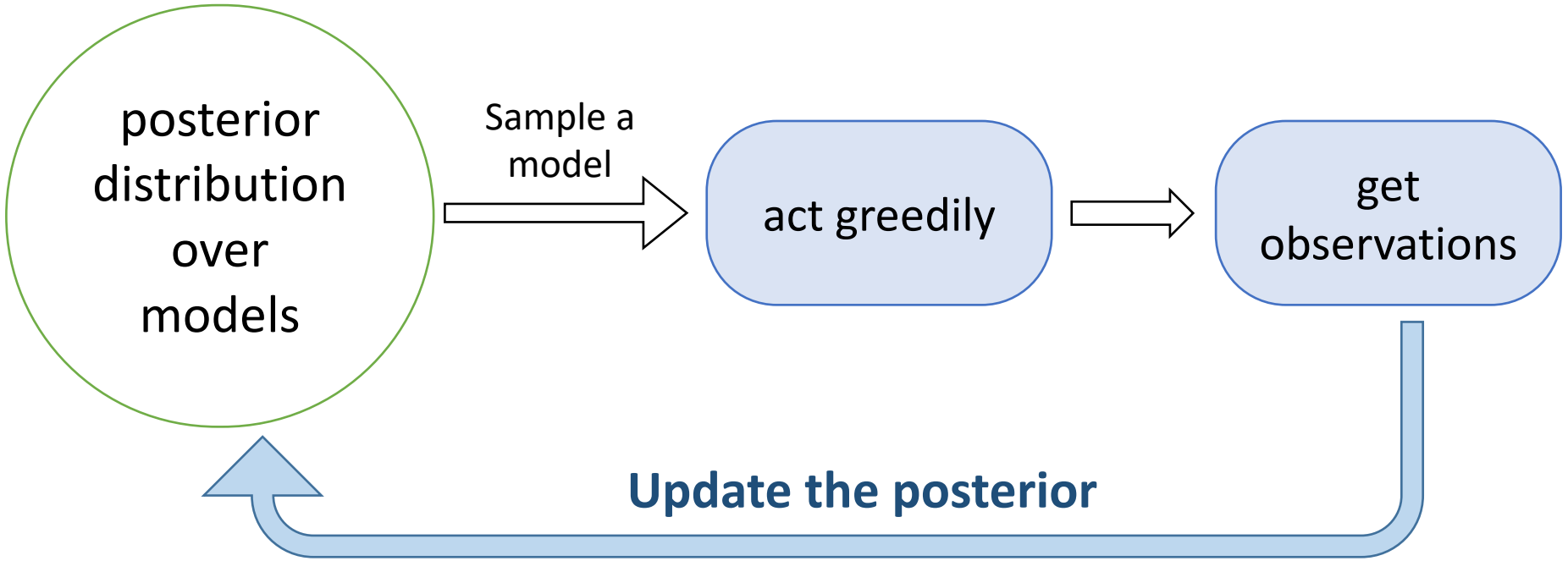
- "model" can be MDP model, state-action value function, or optimal policy
  - correspond to model learning, value learning, and policy learning in RL
- **in contextual bandits:** model learning = value learning
  - since actions do not influence state transitions

Why does Thompson Sampling Work?

Key Insight: TS samples actions according to the posterior probability that they are optimal

- roughly speaking: Thompson sampling
  - tries all promising actions
  - while gradually discarding those are believed to underperform
- TS balances exploration and exploitation

Computational Tractability



- TS requires sampling from posteriors and updating them based on new observations

Computational Tractability

- in general, both sampling from the posterior and updating the posterior can be computationally intractable
- related to approximate Bayes inference and deep Bayesian learning
  - variational inference
  - Markov chain Monte Carlo (MCMC)
  - Bayesian neural network
  - and others
- however, TS will be computationally efficient if we have a conjugate prior

Conjugate Prior

- conjugate prior:
  - if the posterior distribution is in the same probability distribution family as the prior distribution
  - usually, the posterior distribution can be efficiently computed
- **Example 1:** Beta-Bernoulli bandit with K actions
  - each action a has a mean reward $r(a) \in [0, 1]$
  - at each time t, if the agent chooses action A_t, then it will observe and receive a binary reward $R_{t+1} \sim \text{Bern}\big(r(A_t)\big)$
  - $r(a)$ is unknown, and the prior over $r(a)$ is independent Beta distribution
  - the posterior is also independent Beta distribution
    - and can be efficiently computed

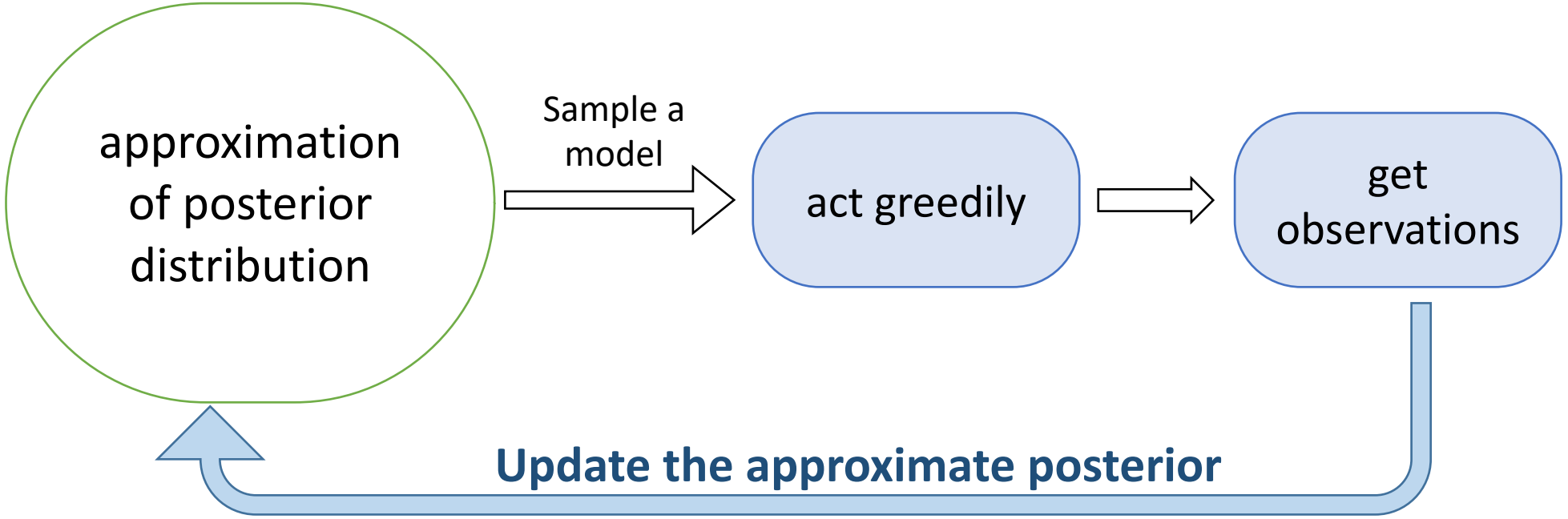
Conjugate Prior

- **Example 2:** linear Gaussian bandit
  - each action $a \in \mathcal{R}^d$ is a d-dimensional vector
  - at each time t, if the agent chooses action A_t, then it will observe and receive a reward

$$R_{t+1} = A_t^T \theta + w_{t+1},$$

    where $w_{t+1} \sim \mathrm{N}(0, \sigma^2)$ is an independent Gaussian noise
  - $\theta \in \mathcal{R}^d$ is unknown, and the prior over it is a Gaussian distribution
  - the posterior over θ is also a Gaussian distribution

Approximate Thompson Sampling



- maintain an approximation of posterior distribution
- sample from this approximation
- update this approximation
- many approximation approaches have been proposed

Approximate Thompson Sampling

- **Laplace approximation**
  - approximate a posterior with a Gaussian distribution
- **Markov chain Monte-Carlo (MCMC)**
  - Gibbs sampling, Langevin/Hamiltonian Monte-Carlo
- **statistical bootstrapping**
  - randomize history instead of computing posterior
- **ensemble sampling**
  - a particle based algorithm
- **epistemic neural network (ENN)**
  - a unified framework for uncertainty modeling

Laplace Approximation

- **Laplace Approximation**
  - approximate a posterior with a Gaussian distribution
- a useful approximation under some technical conditions
  - the posterior is unimodal (has only one peak)
  - log posterior is strictly concave around its mode
- how to compute the Laplace approximation?
  - compute the mode of the posterior, and use it as the mean of the Gaussian distribution
  - compute the Hessian matrix of the negative log posterior at the mode
  - choose the inverse of the Hessian matrix as the covariance matrix

Laplace Approximation

- approximate TS with Laplace approximation is successful in logistic bandits
  - a model widely used in recommendation system
  - we will see an example later
- in general, Laplace approximation is not a good idea if the posterior is multi-modal
  - if the posterior has multiple peaks
  - e.g. Gaussian mixture

Markov Chain Monte-Carlo

- construct a Markov chain whose (unique) stationary distribution is the posterior distribution
  - running Monte-Carlo on this Markov chain will eventually sample from the posterior distribution
- different versions of MCMC algorithms correspond to different Markov chains
  - the Markov chain can be discrete-time or continuous-time
  - the continuous-time Markov chain is also known as stochastic differential equation (SDE)
- convergence rate of the Markov chain is crucial

Statistical Bootstrap

- standard way to compute a point estimate
  - define a loss function $l(\theta)$, where θ encodes the unknown parameters
    - one standard choice of $l(\theta)$ is the negative log posterior
  - compute a point estimate $\theta^* \in \operatorname{argmax}_\theta l(\theta)$
- **statistical bootstrap:**
  - randomly perturb the loss function $l(\theta)$ to $l(\theta; w)$, where *w* is random
  - examples:
    - add random noises to the observations
    - randomly reweigh the log likelihood terms
  - compute $\theta^*(w) \in \operatorname{argmax}_\theta l(\theta; w)$
  - since w is random, $\theta^*(w)$ is also random

Statistical Bootstrap

- we hope that, if the random perturbations are done correctly, the distribution of $\theta^*(w)$ is "close" to the posterior distribution
- **theory:** for some special cases, we know how to perturb the loss so that the distribution of $\theta^*(w)$ matches the posterior
  - Beta-Bernoulli case
  - linear Gaussian case
- the optimal statistical bootstrap for more general cases is unclear
- practitioners just try some "standard" statistical bootstraps
  - e.g. add Gaussian noises to the observations

Ensemble and ENN

- **ensemble:** use an ensemble of particles (point estimates) to approximate the posterior
  - can approximate any posterior
  - however, the number of required particles might be intractably large
- **epistemic neural network (ENN):**
  - a unified framework for uncertainty modeling
  - an ensemble is an ENN
  - however, there exist ENNs that achieve similar performance as a large ensemble, but with much smaller size
  - will discuss it at the end of this tutorial

How to Choose Prior?

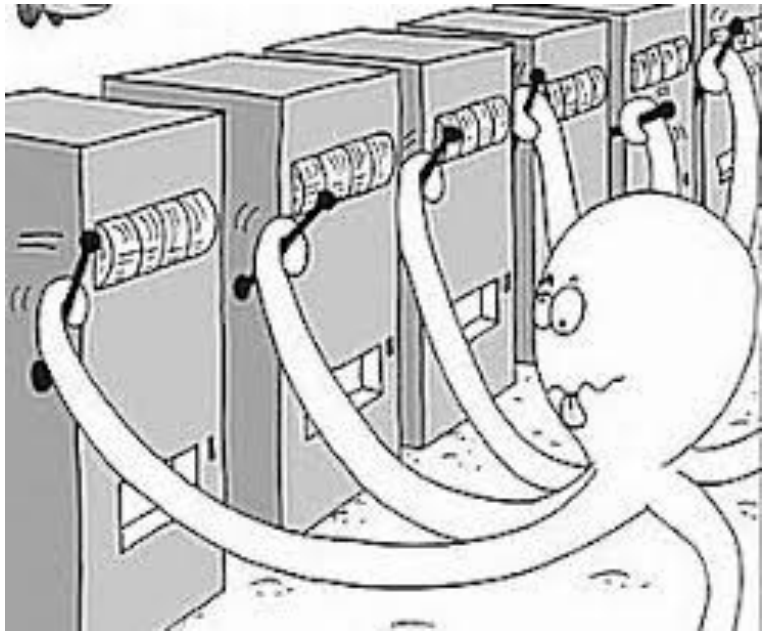
- recall that TS maintains a posterior distribution over models
  - which is initialized by a prior distribution over models
- how to choose this prior distribution in practice?
  - should encode your prior information into this prior distribution
  - prior information might include
    - domain knowledge/expert advice
    - information learned from offline datasets
    - information transferred from other tasks
  - **no prior information:** try using weak prior

Non-Stationary Problems

- vanilla version of TS is developed under the assumption that the environment is time-invariant
- in practice, the environment might be time-varying
  - interesting research problem
- a simple heuristic: sliding-window TS
  - use the prior and only the observations in the past W steps to compute the posterior
  - W is the window size
- more active approaches: use change detection to actively detect changes

Experiment: Bernoulli Bandits

- K arms (actions)
- Each arm has a Bernoulli reward
  - arms' mean rewards are unknown
- No context, standard bandit problem
- **Performance metric:** per-episode regret
  - The difference between the mean reward of an optimal arm and the arm selected by the algorithm



Experiment: Bernoulli Bandits

- **Comparison:** Greedy vs TS
- Beta distribution is the conjugate prior of Bernoulli reward

Algorithm 1 BernGreedy(K, α, β)

```
for t = 1, 2, ... do
    #estimate model:
    for k = 1, ..., K do
        θ̂_k ← α_k / (α_k + β_k)
    end for

    #select and apply action:
    x_t ← argmax_k θ̂_k
    Apply x_t and observe r_t

    #update distribution:
    (α_{x_t}, β_{x_t}) ← (α_{x_t} + r_t, β_{x_t} + 1 − r_t)
end for
```

Algorithm 2 BernTS(K, α, β)

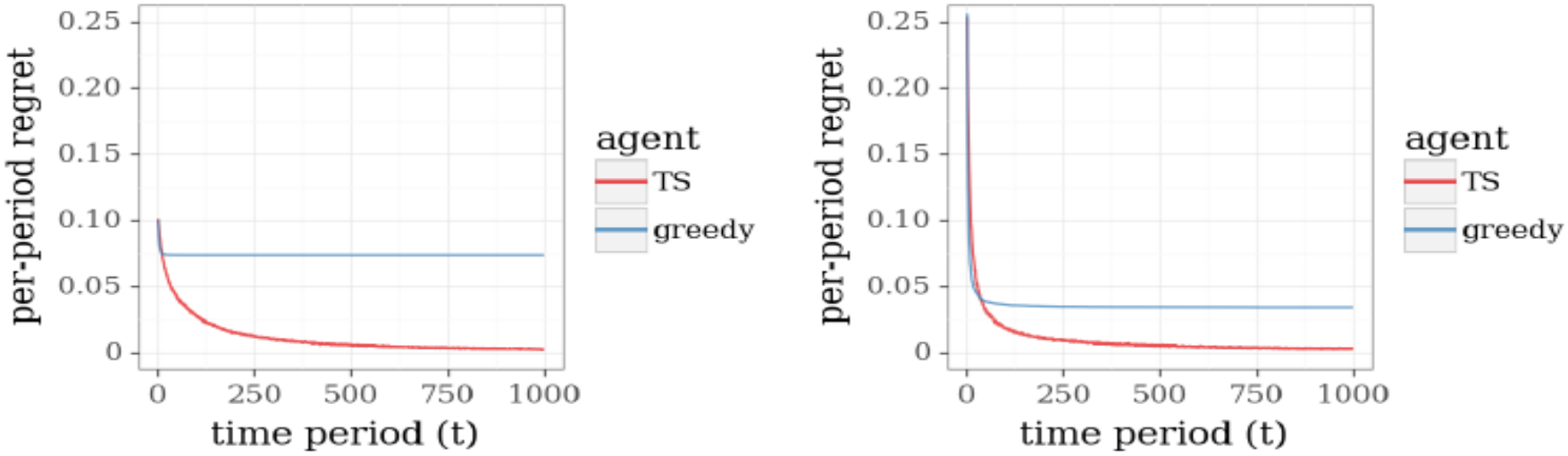
```
for t = 1, 2, ... do
    #sample model:
    for k = 1, ..., K do
        Sample θ̂_k ~ beta(α_k, β_k)
    end for

    #select and apply action:
    x_t ← argmax_k θ̂_k
    Apply x_t and observe r_t

    #update distribution:
    (α_{x_t}, β_{x_t}) ← (α_{x_t} + r_t, β_{x_t} + 1 − r_t)
end for
```

Experiment: Bernoulli Bandits

- TS outperforms greedy
- Per-episode regret of greedy does not decrease to 0, since it does not explore



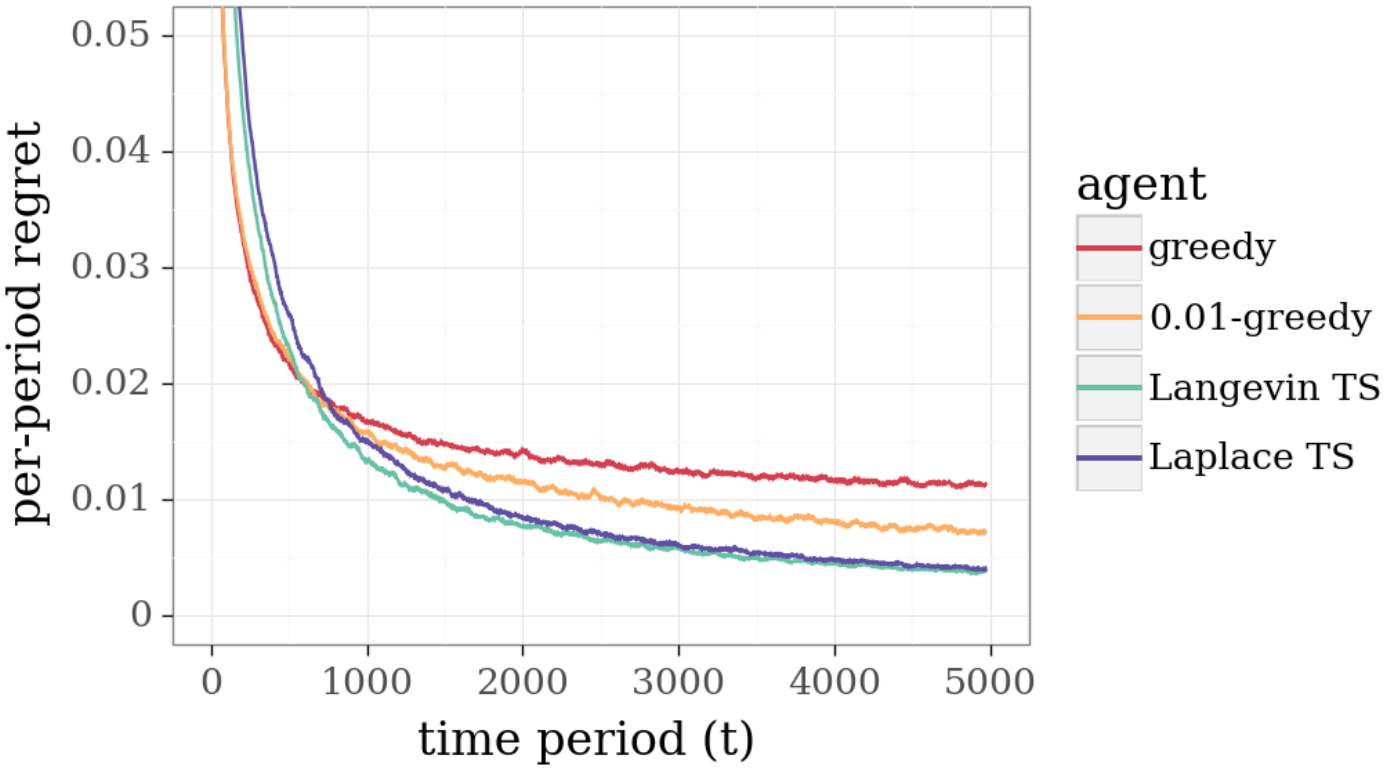
(a) $\theta = (0.9, 0.8, 0.7)$ (b) average over random θ

News Article Recommendation

- in each round t, recommender system
  - observes a feature vector z_t associated with the t^{th} user
  - chooses a news article x_t to display from a set of k articles
  - then observes a binary reward $r_t \in \{0, 1\}$ indicating whether the user likes this article
- contextual bandit with generalized linear model
  - each article x is associated with an unknown parameter vector $\boldsymbol{\theta}_x$
  - when recommending article x_t to user with feature z_t, the user will like this article with probability $g(\theta_{x_t}^T z_t)$, where g is the logistic function

News Article Recommendation

- approximate TS outperforms greedy and epsilon-greedy



Outline

1. reinforcement learning and contextual bandits
2. Thompson sampling (TS): algorithm and practical considerations
3. **analysis techniques of TS: a high-level overview**
4. discussion and some advanced research topics

Regret Bounds via UCB

- UCB is a class of widely used algorithms for bandit/RL
- in the contextual bandit setting, at each step
  - construct an upper confidence bound (UCB) for each context-action pair
  - typically, UCB = (best point estimate) + (confidence interval size)
  - then, choose the arm with the highest UCB
- UCB: one index balancing exploration and exploitation
  - large UCB: either large point estimate (exploitation), or large confidence interval size (exploration)
  - **"optimistic in the face of uncertainty"**
- UCB is better understood in theory, but works worse in practice

Regret Bounds via UCB

- one idea to analyze TS: to show TS is “similar” to UCB
- **Insight one:**
  - UCB is optimistic all the time, but TS is optimistic with significant probability
    - “significant prob”: prob. bounded away from 0
  - with high prob, once a while TS will choose an optimistic action
    - TS behaves like a “slower version” of UCB
  - leads to worst-case regret bounds for variants of TS
  - however, it does not explain why TS usually performs better than UCB

Regret Bounds via UCB

- **Insight two:**
  - typical regret decomposition under UCB:

$$\begin{aligned}\mu(x^*,\theta)-\mu(\overline{x}_t,\theta) &= \mu(x^*,\theta)-U_t(\overline{x}_t)+U_t(\overline{x}_t)-\mu(\overline{x}_t,\theta)\\ &\leq \underbrace{\mu(x^*,\theta)-U_t(x^*)}_{\text{pessimism}}+\underbrace{U_t(\overline{x}_t)-\mu(\overline{x}_t,\theta)}_{\text{width}}.\end{aligned}$$

optimal action

(uncertain) model parameter

action chosen by algorithm

  - if we can develop a similar regret decomposition for TS
    - then we can reuse many techniques in UCB analyses to establish regret bounds for TS

Regret Bounds via UCB

- **a key insight:** under TS

$$\mathbb{E}[U_t(x_t)] = \mathbb{E}[\mathbb{E}[U_t(x_t)|\mathbb{H}_{t-1}]] = \mathbb{E}[\mathbb{E}[U_t(x^*)|\mathbb{H}_{t-1}]] = \mathbb{E}[U_t(x^*)]$$

- similar regret decomposition for TS

$$\begin{aligned}\mathbb{E}\left[\mu(x^*,\theta) - \mu(x_t,\theta)\right] &= \mathbb{E}\left[\mu(x^*,\theta) - U_t(x_t)\right] + \mathbb{E}\left[U_t(x_t) - \mu(x_t,\theta)\right] \\ &= \underbrace{\mathbb{E}\left[\mu(x^*,\theta) - U_t(x^*)\right]}_{\text{pessimism}} + \underbrace{\mathbb{E}\left[U_t(x_t) - \mu(x_t,\theta)\right]}_{\text{width}}.\end{aligned}$$

- bayes regret bound for TS can be established

Regret Bounds via Information Theory

- **Information Ratio:** for any problem and any learning algorithm

$$\Gamma_t = \frac{(\mathbb{E}\left[\mu(x^*, \theta) - \mu(x_t, \theta)\right])^2}{I\left(x^*; (x_t, y_t) | \mathbb{H}_{t-1}\right)}.$$

Performance loss

Information Gain

- information ratio can be viewed as a "cost-benefit ratio" for any learning problem
  - cost: performance loss
  - benefit: information gain
- other versions of information ratio exist
  - different order of performance loss, rather than second order
  - different measures of information

Regret Bounds via Information Theory

- general Bayes regret bound for any learning algorithm:

$$\mathbb{E}\left[\text{Regret}(T)\right] \leq \sqrt{\overline{\Gamma} H(x^*) T}$$

  for any $\overline{\Gamma}$ such that $\Gamma_t \leq \overline{\Gamma}$ for all t

- this bound depends on the prior uncertainty of the optimal action
- sufficient to bound information ratio for TS
  - we can bound information ratio for TS in many classical bandit problems

Summary: How to Analyze TS?

- **Analysis via UCB**
  - show TS is "similar" to UCB
  - use existing analysis techniques for UCB algorithms
  - the UCB function will appear in the analysis
    - though it does not show up in the TS algorithm

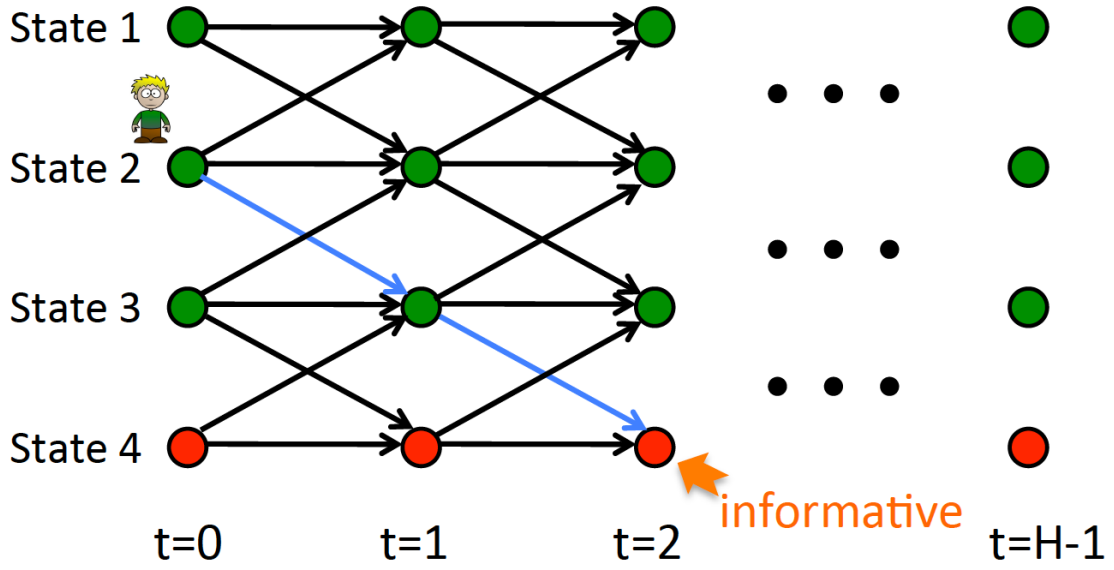
- **Information-Theoretic Analysis**
  - use the general information-theoretic regret bound
  - bound the information ratio for TS

Outline

1. reinforcement learning and contextual bandits
2. Thompson sampling (TS): algorithm and practical considerations
3. analysis techniques of TS: a high-level overview
4. **discussion and some advanced research topics**
   - **Thompson sampling in general RL problems**
   - limitations of TS
   - approximate TS based on epistemic neural networks

Exploration in RL

- exploration in general RL is trickier
  - sometimes need to plan to explore (deep exploration)
  - switching policies too frequently might hurt deep exploration
  - for TS: when to resample a model?



Episodic RL

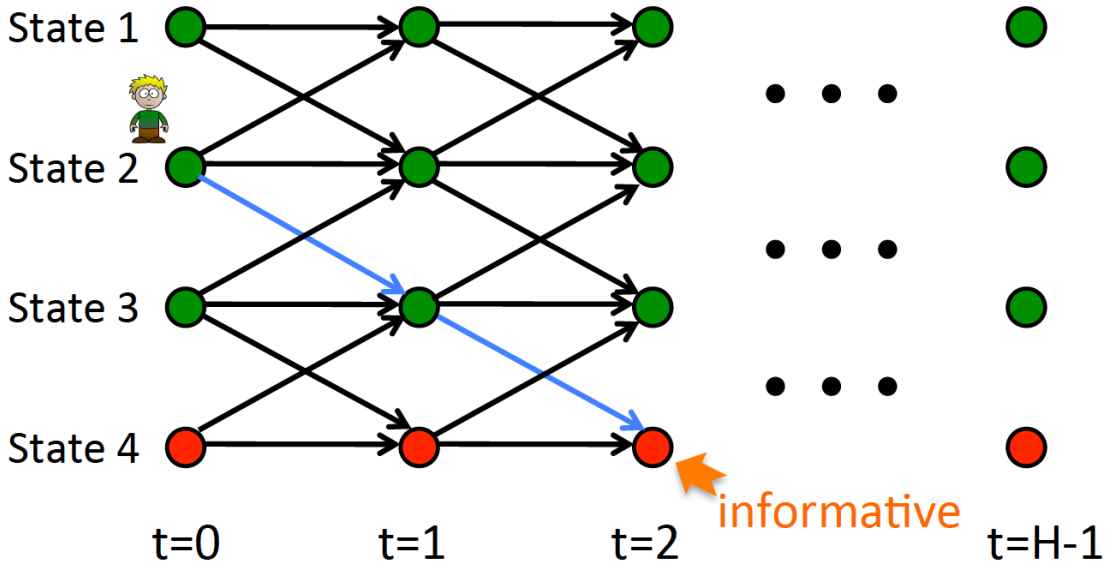
- **episodic RL:**
  - environment is a finite-horizon MDP (S, A, P, R, H, ρ)
    - S: state space, A: action space
    - P: transition model, R: reward model
    - H: finite horizon, ρ: initial state distribution
  - the agent repeatedly interacts with that MDP
  - each interaction is referred to as an episode
- agent does not know the transition model P or the reward model R
  - but has a prior over them
- **Goal:** maximize the expected total reward in T episodes

TS for Episodic RL

- **TS for episodic RL:** once a while
  1. update the posterior based on the new observations
  2. resample a model from the posterior
  3. compute an optimal policy $\hat{\pi}$ under the sampled model
  4. choose actions based on $\hat{\pi}$ until the next posterior update
- **Question:** when and how frequent should we update the posterior and resample a model?
  - should we resample a model every time step?
  - or should we resample a model every episode?

TS for Episodic RL

- recall the requirement for deep exploration
  - agent's exploration strategy should be consistent within one episode
  - better to resample every episode, rather than every time step
  - resample at the beginning of each episode



Outline

1. reinforcement learning and contextual bandits
2. Thompson sampling (TS): algorithm and practical considerations
3. analysis techniques of TS: a high-level overview
4. **discussion and some advanced research topics**
   - Thompson sampling in general RL problems
   - **limitations of TS**
   - approximate TS based on epistemic neural networks

Limitations of TS

- TS is effective for a broad range of problems
  - however, there are contexts for which TS is inappropriate
  - we will highlight some of them
- **problems that do not require exploration**
  - e.g. actions only influence rewards, but not observations
  - **example:** learn to select a portfolio made of publicly traded financial securities
- **problems that do not require exploitation**
  - e.g. classical simulation optimization problem
  - a.k.a. “pure-exploration” problems

Limitations of TS

- **time-sensitive learning problems**
  - TS aims to minimize the exploration cost to learn an optimal strategy
  - in time-sensitive setting, better to exploit a sub-optimal strategy
  - **example:** Bernoulli bandit with K actions
    - the agent will interact with the Bernoulli bandit for *T* time steps
    - time-sensitivity: $K \gg T$
    - in this case, trying to learn the optimal action is not a good idea
    - instead, we should aim to learn a near-optimal action

Limitations of TS

- **problems with complicated "information structure"**
  - TS only chooses strategies that might be optimal
  - in particular, chooses a strategy based on its posterior probability of being optimal
  - what should we do if a strategy is never optimal, but provides a lot of useful information?
    - TS does not perform well in this setting
    - an alternative solution: **information-directed sampling (IDS)**

Outline

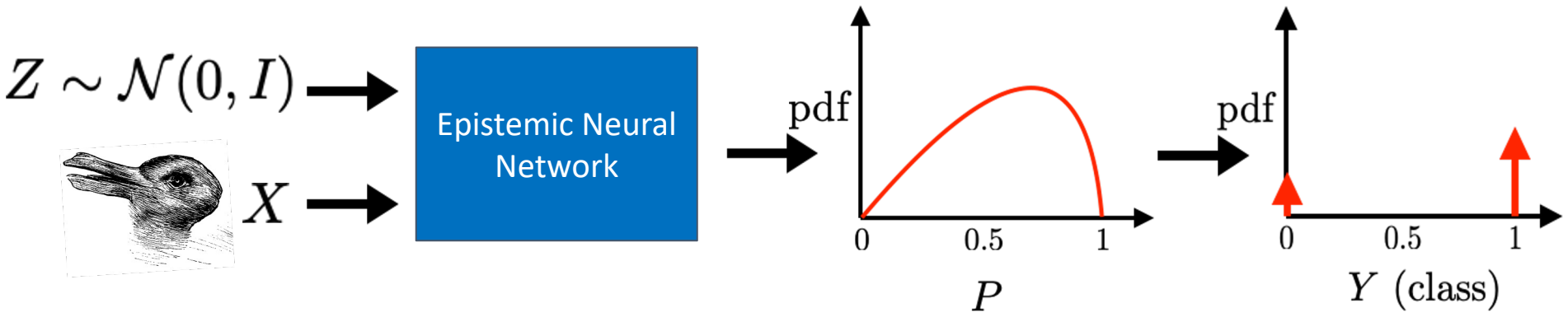
1. reinforcement learning and contextual bandits
2. Thompson sampling (TS): algorithm and practical considerations
3. analysis techniques of TS: a high-level overview
4. **discussion and some advanced research topics**
   - Thompson sampling in general RL problems
   - limitations of TS
   - **approximate TS based on epistemic neural networks**

Epistemic Neural Network

- traditional neural networks do not model epistemic uncertainty
- **epistemic neural network (ENN):**
  - a unified way of thinking about uncertainty in deep learning
- many existing uncertainty modeling approaches can be viewed as ENNs:
  - ensemble agents
  - dropout
  - Bayesian neural networks (BNN)
- **epinet:** a new computationally scalable ENN

Epistemic Neural Network

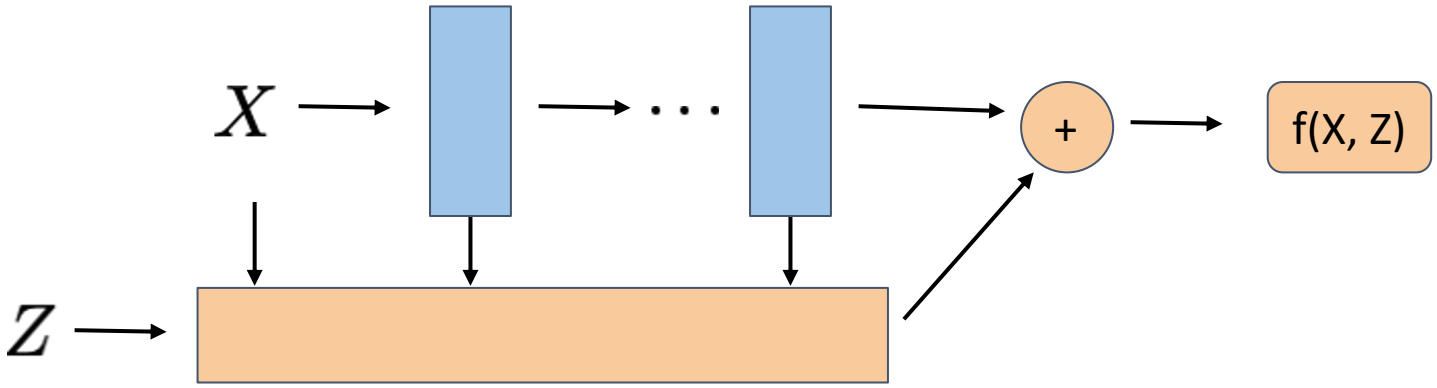
- **Example:** ENN for a binary image classification problem
  - to classify if an image is a rabbit



  - Z is the ENN index, which is drawn from a fixed reference distribution
  - for a fixed X, a reference distribution induces an output distribution
    - which further induces a class distribution

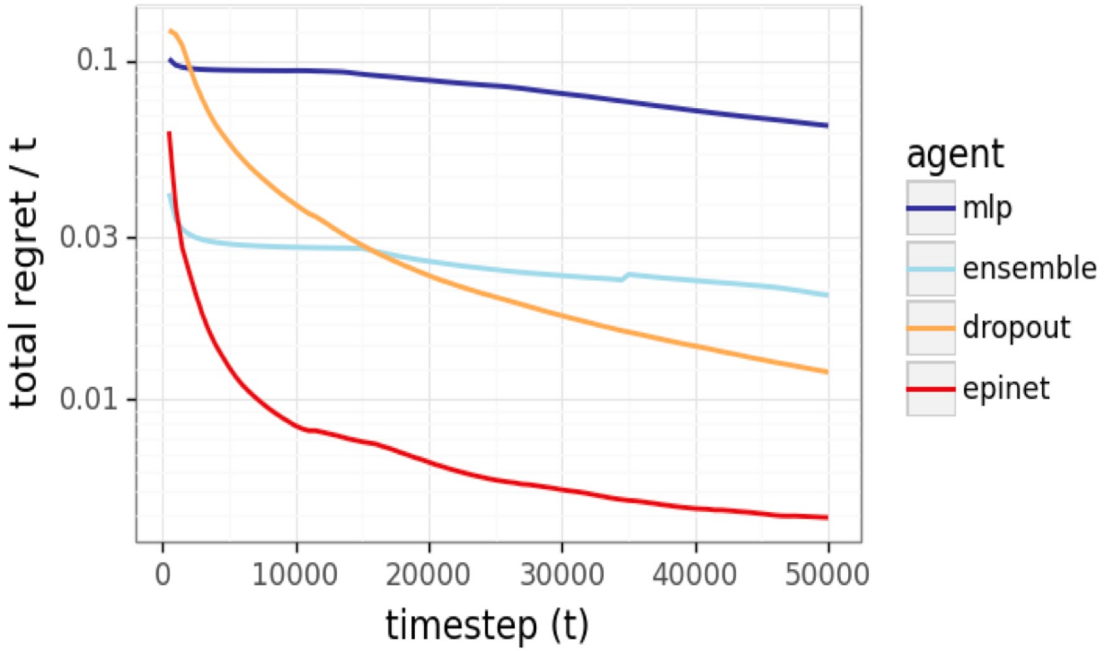
Epinet: a New ENN

- a new computationally scalable ENN
- performs as well as large ensemble, but with much less compute
- **main idea:** uncertainty network with intermediate layer features



Approximate TS via ENN

- many ENNs allow incremental updates and efficient sampling
  - can be used in approximate TS
- appropriate for deep bandit/RL problems



Further Reading

- Russo et al., *"A Tutorial on Thompson Sampling"*
- *Li et al., "A Contextual Bandit Approach to Personalized News Article Recommendation"*
- Agrawal & Goyal, *"Analysis of Thompson sampling for the multi-armed bandit problem"*
- Osband et al., *"Epistemic Neural Networks"*
- Lu et al., *"Reinforcement Learning, Bit by Bit"*