CS885 Reinforcement Learning
Lecture 8b: May 25, 2018

Bayesian and Contextual Bandits
[SutBar] Sec. 2.9

Outline

- Bayesian bandits
  - Thompson sampling
- Contextual bandits

Multi-Armed Bandits

- Problem:
  - N bandits with unknown average reward $R(a)$
  - Which arm a should we play at each time step?
  - Exploitation/exploration tradeoff

- Common frequentist approaches:
  - ϵ-greedy
  - Upper confidence bound (UCB)

- Alternative Bayesian approaches
  - Thompson sampling   we explicitly model our uncertainty
  - Gittins indices

Bayesian Learning

we model the uncertainty of this random variable using a parameterized distribution. If we knew the true distribution, we could just sample from it to get R(a)

- Notation:
  - r^a: random variable for a's rewards
  - $\Pr(r^a; \theta)$: unknown distribution (parameterized by θ)
  - $R(a) = E[r^a]$: unknown average reward

prior reflects our belief of what the distribution looks like

- Idea:
  - Express uncertainty about θ by a prior $\Pr(\theta)$
  - Compute posterior $\Pr(\theta | r_1^a, r_2^a, \ldots, r_n^a)$ based on samples $r_1^a, r_2^a, \ldots, r_n^a$ observed for a so far.

using these samples, we can update our belief about theta. In the long run, we will converge on the true theta

- Bayes theorem:

$$\Pr(\theta | r_1^a, r_2^a, \ldots, r_n^a) \propto \Pr(\theta) \Pr(r_1^a, r_2^a, \ldots, r_n^a | \theta)$$

posterior

date likelihood. The likelihood that we would have draw our samples given theta

prior

Distributional Information

if we knew theta, the prediction would be given by Pr(r_a; theta). But since we don't know theta, we need to consider all many possible thetas, which are each weighted by the posterior. So the integral takes a weighted combination over the distributions based on different possible thetas, each one weighted by the posterior.

- Posterior over θ allows us to estimate
  - Distribution over next reward r^a

predict the next reward

$$\Pr(r^a | r_1^a, r_2^a, \ldots, r_n^a) = \int_\theta \Pr(r^a; \theta) \Pr(\theta | r_1^a, r_2^a, \ldots, r_n^a)\, d\theta$$

posterior

we actually aren't interested in predicting the next reward. We want to predict the average reward R(a)!

  - Distribution over $R(a)$ when θ includes the mean

$$\Pr(R(a) | r_1^a, r_2^a, \ldots, r_n^a) = \Pr(\theta | r_1^a, r_2^a, \ldots, r_n^a) \text{ if } \theta = R(a)$$

- To guide exploration:
  - UCB: $\Pr\left(R(a) \leq bound(r_1^a, r_2^a, \ldots, r_n^a)\right) \geq 1 - \delta$
  - Bayesian techniques: $\Pr\left(R(a) | r_1^a, r_2^a, \ldots, r_n^a\right)$

Coin Example

- Consider two biased coins C_1 and C_2

$$R(C_1) = \Pr(C_1 = head)$$
$$R(C_2) = \Pr(C_2 = head)$$

- Problem:
  - Maximize # of heads in k flips
  - Which coin should we choose for each flip?

Bernoulli Variables

- r^{C_1}, r^{C_2} are Bernoulli variables with domain $\{0,1\}$

- Bernoulli dist. are parameterized by their mean
  - i.e. $\Pr(r^{C_1};\theta_1) = \theta_1 = R(C_1)$
    $\Pr(r^{C_2};\theta_2) = \theta_2 = R(C_2)$

Beta distribution

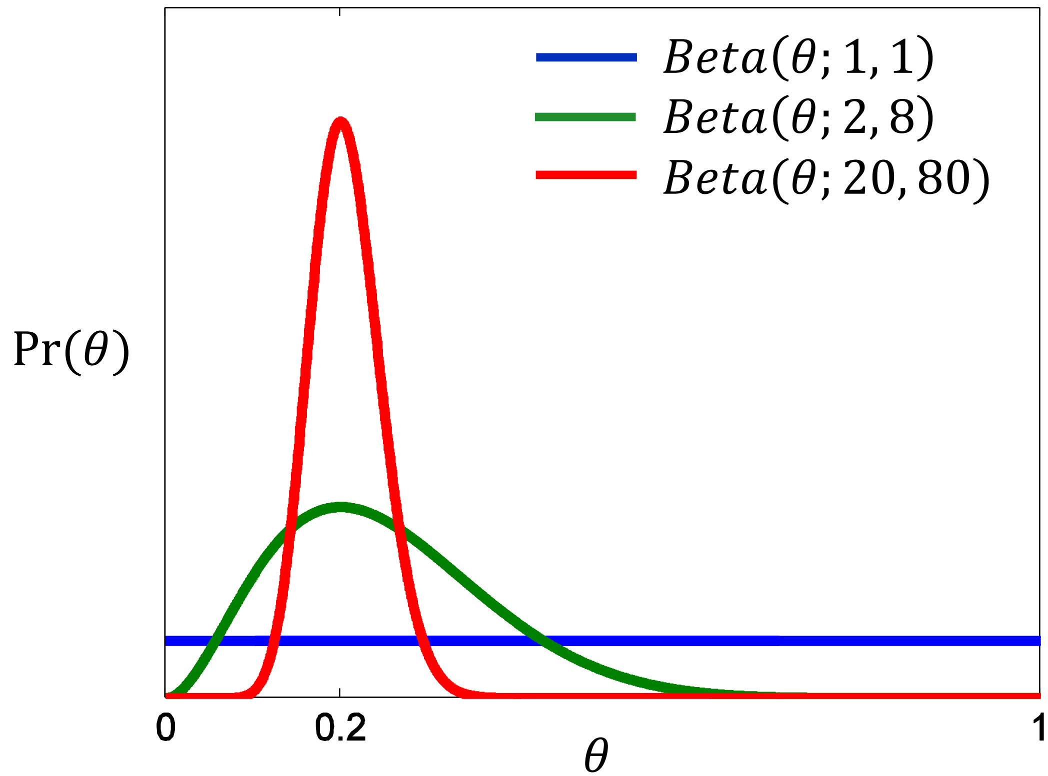
Since we don't know theta, we create a distribution over theta. Because theta is a Bernoulli variable that ranges for 0 to 1, we will represent the distribution as a Beta distribution.

- Let the prior $\Pr(\theta)$ be a Beta distribution

$$Beta(\theta;\alpha,\beta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}$$

- $\alpha - 1$: # of heads
- $\beta - 1$: # of tails

- $E[\theta] = \alpha/(\alpha+\beta)$



Belief Update

- Prior: $\Pr(\theta) = Beta(\theta; \alpha, \beta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}$

likelihood. For bernoulli, the likelihood that we get head given theta is just theta

prior

- Posterior after coin flip:

$$
\begin{aligned}
\Pr(\theta|head) &\propto \quad \Pr(\theta) \qquad \Pr(head|\theta) \\
&\propto \theta^{\alpha-1}(1-\theta)^{\beta-1} \qquad \theta \\
&= \theta^{(\alpha+1)-1}(1-\theta)^{\beta-1} \\
&\propto Beta(\theta; \alpha+1, \beta)
\end{aligned}
$$

exponent rules

posterior, which is also a beta distribution

$$
\begin{aligned}
\Pr(\theta|tail) &\propto \quad \Pr(\theta) \qquad \Pr(tail|\theta) \\
&\propto \theta^{\alpha-1}(1-\theta)^{\beta-1} \quad (1-\theta) \\
&= \theta^{\alpha-1}(1-\theta)^{(\beta+1)-1} \\
&\propto Beta(\theta; \alpha, \beta+1)
\end{aligned}
$$

exponent rules

Thompson Sampling

samples possible means

recall this is the distribution of mean rewards

- Idea:
  - Sample several potential average rewards:
    $R_1(a), \ldots R_k(a) \sim \Pr(R(a)|r_1^a, \ldots, r_n^a)$ for each a
  - Estimate empirical average
    $\hat{R}(a) = \frac{1}{k}\sum_{i=1}^{k} R_i(a)$
  - Execute $argmax_a \, \hat{R}(a)$

- Coin example
  - $\Pr(R(a)|r_1^a, \ldots, r_n^a) = \text{Beta}(\theta_a; \alpha_a, \beta_a)$
    where $\alpha_a - 1 = \#heads$ and $\beta_a - 1 = \#tails$

Thompson Sampling Algorithm Bernoulli Rewards

ThompsonSampling(h)

$V \leftarrow 0$

For $n = 1$ to h

Sample $R_1(a), \ldots, R_k(a) \sim \Pr(R(a)) \;\; \forall a$

for each arm, retrieve k samples from the prior distribution

$\hat{R}(a) \leftarrow \frac{1}{k}\sum_{i=1}^{k} R_i(a) \quad \forall a$

for each arm, find the average of the samples

$a^* \leftarrow \text{argmax}_a \, \hat{R}(a)$

Execute a^* and receive r

$V \leftarrow V + r$

Update $\Pr(R(a^*))$ based on r

calculate posterior

Return V

we first create posterior and then same from that. This creates exploration!

Comparison

Thompson Sampling

- Action Selection

$$a^* = \text{argmax}_{\text{a}}\ \hat{R}(a)$$

- Empirical mean

$$\hat{R}(a) = \frac{1}{k}\sum_{i=1}^{k} R_i(a)$$

- Samples

$$R_i(a) \sim \Pr(R_i(a) | r_1^a \dots r_n^a)$$

$$r_i^a \sim \Pr(r^a; \theta)$$

- Some exploration

i refers to ith sample that you get from your posterior distribution

Greedy Strategy

- Action Selection

$$a^* = \text{argmax}_{\text{a}}\ \tilde{R}(a)$$

- Empirical mean

$$\tilde{R}(a) = \frac{1}{n}\sum_{i=1}^{n} r_i^a$$

- Samples

$$r_i^a \sim \Pr(r^a; \theta)$$

- No exploration

i refers to the ith timestep

Sample Size

- In Thompson sampling, amount of data n and sample size k regulate amount of exploration

- As n and k increase, $\hat{R}(a)$ becomes less stochastic, which reduces exploration
  - As $n \uparrow$, $\Pr(R(a)|r_1^a \ldots r_n^a)$ becomes more peaked
  - As $k \uparrow$, $\hat{R}(a)$ approaches $E[R(a)|r_1^a \ldots r_n^a]$

- The stochasticity of $\hat{R}(a)$ ensures that all actions are chosen with some probability

often, we want to get a sample size as large as possible to get a good estimate. But in this case, we want a small sample size to increase exploration!

Analysis

- Thompson sampling converges to best arm
- Theory:
  - Expected cumulative regret: $O(\log n)$
  - On par with UCB and ϵ-greedy
- Practice:
  - Sample size k often set to 1

Contextual Bandits

- In many applications, the context provides additional information to select an action
  - E.g., personalized advertising, user interfaces
  - Context: user demographics (location, age, gender)

- Actions can also be characterized by features that influence their payoff
  - E.g., ads, webpages
  - Action features: topics, keywords, etc.

you can categorize ads and webpages by topics, keywords, etc

so we can think over the arms as being parameterized by features

Contextual Bandits

- Contextual bandits: multi-armed bandits with states (corresponding to contexts) and action features
- Formally:
  - S: set of states where each state s is defined by a vector of features $\boldsymbol{x}^s = (x_1^s, x_2^s, \dots, x_k^s)$
  - A: set of actions where each action a is associated with a vector of features $\boldsymbol{x}^a = (x_1^a, x_2^a, \dots, x_l^a)$
  - Space of rewards (often $\mathbb{R}$)

- No transition function since the states at each step are independent
- Goal find policy $\pi: \boldsymbol{x}^s \to a$ that maximizes expected rewards $E(r|s,a) = E(r|\boldsymbol{x}^s, \boldsymbol{x}^a)$

Approximate Reward Function

- Common approach:
  - learn approximate average reward function $\tilde{R}(s,a) = \tilde{R}(\boldsymbol{x})$ (where $\boldsymbol{x} = (\boldsymbol{x}^s, \boldsymbol{x}^a)$) by regression

    we want to approximate the average reward for taking action a (parameterized as x_a)
    given the state s (paramterized as x_s)

- Linear approximation: $\tilde{R}_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$

- Non-linear approximation: $\tilde{R}_{\boldsymbol{w}}(\boldsymbol{x}) = neuralNet(\boldsymbol{x};\boldsymbol{w})$

goal: find best set of weight to allow us to accurately model the reward function

Bayesian Linear Regression

we have uncertainty around our weights for our model, so let's represent this using a Gaussian prior

- Consider a Gaussian prior:

$$pdf(\boldsymbol{w}) = N(\boldsymbol{w}|\boldsymbol{0}, \lambda^2\boldsymbol{I}) \propto exp\left(-\frac{\boldsymbol{w}^T\boldsymbol{w}}{2\lambda^2}\right)$$

- Consider also a Gaussian likelihood:

$$pdf(r|\boldsymbol{x}, \boldsymbol{w}) = N(r|\boldsymbol{w}^T\boldsymbol{x}, \sigma^2) \propto exp\left(-\frac{(r - \boldsymbol{w}^T\boldsymbol{x})^2}{2\sigma^2}\right)$$

- The posterior is also Gaussian:

$$pdf(\boldsymbol{w}|r, \boldsymbol{x}) \propto pdf(\boldsymbol{w}) \Pr(r|\boldsymbol{x}, \boldsymbol{w})$$

$$\propto exp\left(-\frac{\boldsymbol{w}^T\boldsymbol{w}}{2\lambda^2}\right) exp\left(-\frac{(r - \boldsymbol{w}^T\boldsymbol{x})^2}{2\sigma^2}\right)$$

$$= N(\boldsymbol{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

if you multiply these together and rearrange the terms, you can rewrite this in the form of a Gaussian

where $\boldsymbol{\mu} = \sigma^{-2}\boldsymbol{\Sigma}\boldsymbol{x}r$ and $\boldsymbol{\Sigma} = (\sigma^{-2}\boldsymbol{x}\boldsymbol{x}^T + \lambda^{-2}\boldsymbol{I})^{-1}$

Predictive Posterior

- Consider a state-action pair $(\boldsymbol{x}^s, \boldsymbol{x}^a) = \boldsymbol{x}$ for which we would like to predict the reward r

- Predictive posterior:

$$pdf(r|\boldsymbol{x}) = \int_{\boldsymbol{w}} N(r|\boldsymbol{w}^T\boldsymbol{x}, \sigma^2)\, N(\boldsymbol{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma})\, d\boldsymbol{w}$$
$$= N(r|\sigma^2\boldsymbol{x}^T\boldsymbol{\mu}, \boldsymbol{x}^T\boldsymbol{\Sigma}\boldsymbol{x})$$

- UCB: $\Pr\left(r < \sigma^2\boldsymbol{x}^T\boldsymbol{\mu} + c\sqrt{\boldsymbol{x}^T\boldsymbol{\Sigma}\boldsymbol{x}}\right) > 1 - \delta$

  where $c = 1 + \sqrt{\ln(2/\delta)/2}$

- Thomson sampling: $\tilde{r} \sim N(r|\sigma^2\boldsymbol{x}^T\boldsymbol{\mu}, \boldsymbol{x}^T\boldsymbol{\Sigma}\boldsymbol{x})$

Upper Confidence Bound (UCB) Linear Gaussian

UCB(h)

$V \leftarrow 0,\ pdf(\boldsymbol{w}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = N(\boldsymbol{w}|\boldsymbol{0},\lambda^2\boldsymbol{I})$

Repeat until $n = h$

Receive state $\boldsymbol{x}^s$

For each action $\boldsymbol{x}^a$ where $\boldsymbol{x} = (\boldsymbol{x}^s,\boldsymbol{x}^a)$ do

$confidenceBound(a) = \sigma^2\boldsymbol{x}^T\boldsymbol{\mu} + c\sqrt{\boldsymbol{x}^T\boldsymbol{\Sigma}\boldsymbol{x}}$

$a^* \leftarrow \mathrm{argmax_a}\ confidenceBound(a)$

Execute a^* and receive r

$V \leftarrow V + r$

update $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ based on $\boldsymbol{x} = (\boldsymbol{x}^s,\boldsymbol{x}^{a^*})$ and r

Return V

Thompson Sampling Algorithm Linear Gaussian

ThompsonSampling(h)

$V \leftarrow 0$; $pdf(\boldsymbol{w}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = N(\boldsymbol{w}|\boldsymbol{0},\lambda^2\boldsymbol{I})$

For $n = 1$ to h

Receive state $\boldsymbol{x}^s$

For each action $\boldsymbol{x}^a$ where $\boldsymbol{x} = (\boldsymbol{x}^s, \boldsymbol{x}^a)$ do

Sample $R_1(a), \ldots, R_k(a) \sim N(r|\sigma^2\boldsymbol{x}^T\boldsymbol{\mu}, \boldsymbol{x}^T\boldsymbol{\Sigma}\boldsymbol{x})$

$\hat{R}(a) \leftarrow \frac{1}{k}\sum_{i=1}^{k} R_i(a)$

$a^* \leftarrow \text{argmax}_a \hat{R}(a)$

Execute a^* and receive r

$V \leftarrow V + r$

update $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ based on $\boldsymbol{x} = (\boldsymbol{x}^s, \boldsymbol{x}^{a^*})$ and r

Return V

Industrial Use

- Contextual bandits are now commonly used for
  - Personalized advertising
  - Personalized web content
    - MSN news: 26% improvement in click through rate after adoption of contextual bandits (https://www.microsoft.com/en-us/research/blog/real-world-interactive-learning-cusp-enabling-new-class-applications/)