Lecture 9: Exploration and Exploitation

David Silver

Outline

Exploration vs. Exploitation Dilemma

- Online decision-making involves a fundamental choice:

  Exploitation Make the best decision given current information

  Exploration Gather more information Doing something else (new things)

- The best long-term strategy may involve short-term sacrifices
- Gather enough information to make the best overall decisions

State-action Exploration vs. Parameter Exploration (We will focus on state-action exploration)

State-action exploration

- Systematically explore state space / action space
- e.g. Pick different action A each time S is visited

Parameter exploration

- Parameterize policy $\pi(A|S,u)$ - Disadvantage : doesn't know about state/action space
- e.g. Pick different parameters and try for a while - Advantage: consistent exploration

Examples

- Restaurant Selection
  - **Exploitation** Go to your favourite restaurant
  - **Exploration** Try a new restaurant
- Online Banner Advertisements
  - **Exploitation** Show the most successful advert
  - **Exploration** Show a different advert
- Oil Drilling
  - **Exploitation** Drill at the best known location
  - **Exploration** Drill at a new location
- Game Playing
  - **Exploitation** Play the move you believe is best
  - **Exploration** Play an experimental move

Principles

Three approachs
to the exploration problem
(and there're more)

- Naive Exploration
  - Add noise to greedy policy (e.g. ϵ-greedy)
- Optimistic Initialisation
  - Assume the best until proven otherwise
- Optimism in the Face of Uncertainty

fundamental principle

  - Prefer actions with uncertain values
- Probability Matching
  - Select actions according to probability they are best
- Information State Search
  - Lookahead search incorporating value of information

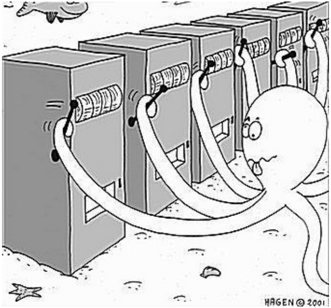
* Random exploration
- Explore random actions
(e.g. ε-greedy, softmax)
* Optimism in the face of uncertainty
- Estimate uncertainty on value
- Prefer to explore states/actions with highest uncertainty
* Information state space
- Consider agent's information as part of its state
- Lookahead to see how information helps reward

correct but computation is very difficult because out state space blows up to something. So it is massively more complicated and difficult than before.

The Multi-Armed Bandit

Simplified simple MDP : thrown away the state space and transition matrix

- A multi-armed bandit is a tuple $\langle \mathcal{A}, \mathcal{R} \rangle$
- $\mathcal{A}$ is a known set of m actions (or "arms")
- P[R=r | A=a]
  $\mathcal{R}^a(r) = \mathbb{P}[r|a]$ is an unknown probability distribution over rewards
- At each step t the agent selects an action $a_t \in \mathcal{A}$
  one arm
- The environment generates a reward $r_t \sim \mathcal{R}^{a_t}$
- The goal is to maximise cumulative reward $\sum_{\tau=1}^{t} r_\tau$



one step MDP

Regret

- The *action-value* is the mean reward for action a,

$$Q(a) = \mathbb{E}\left[r | a\right]$$

true payout (experience)
(we don't have any state)

- The *optimal value* V^* is

$$V^* = Q(a^*) = \max_{a \in \mathcal{A}} Q(a)$$

- The *regret* is the opportunity loss for one step

$$l_t = \mathbb{E}\left[V^* - Q(a_t)\right]$$

- The *total regret* is the total opportunity loss

$$L_t = \mathbb{E}\left[\sum_{\tau=1}^{t} V^* - Q(a_\tau)\right]$$

- Maximise cumulative reward $\equiv$ minimise total regret

Counting Regret

- The *count* $N_t(a)$ is expected number of selections for action a
- The *gap* Δ_a is the difference in value between action a and optimal action a^*, $\Delta_a = V^* - Q(a)$
- Regret is a function of gaps and the counts

optimal value

Q : payout of the machine actually we picked

$$L_t = \mathbb{E}\left[\sum_{\tau=1}^{t} V^* - Q(a_\tau)\right]$$

The amount that we lose at every step is the difference.

We don't know v*.

$$= \sum_{a \in \mathcal{A}} \mathbb{E}\left[N_t(a)\right]\left(V^* - Q(a)\right)$$

count that we pick a specific action(bandit machine) a

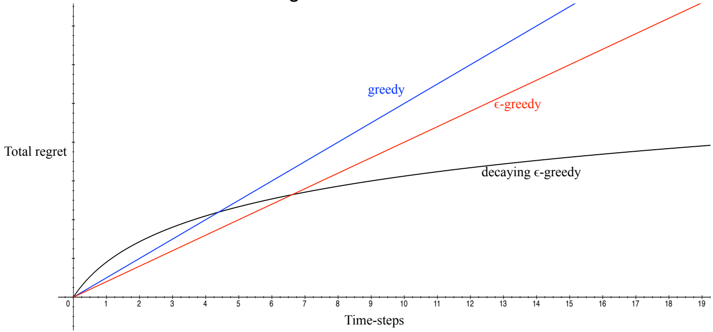
$$= \sum_{a \in \mathcal{A}} \mathbb{E}\left[N_t(a)\right]\Delta_a$$

- A good algorithm ensures small counts for large gaps
- Problem: gaps are not known!

We don't know v*.

Linear or Sublinear Regret

The real question here is what the total regret looks like over time.



- If an algorithm forever explores it will have linear total regret
- If an algorithm never explores it will have linear total regret
- Is it possible to achieve sublinear total regret? The answer is YES.

Greedy Algorithm

estimated value

- We consider algorithms that estimate $\hat{Q}_t(a) \approx Q(a)$
- Estimate the value of each action by Monte-Carlo evaluation

$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{t=1}^{T} r_t \mathbf{1}(a_t = a)$$

- The *greedy* algorithm selects action with highest value

$$a_t^* = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ \hat{Q}_t(a)$$

- Greedy can lock onto a suboptimal action forever
- $\Rightarrow$ Greedy has linear total regret

optimal 하지 않은 action을 계속 하기 때문에 total reward는 계속 증가한다.
(greedy한 action이 정해지므로 같은 action만 반복하고 linear하게 증가)

ϵ-Greedy Algorithm

- The *ϵ-greedy* algorithm continues to explore forever
  - With probability $1 - \epsilon$ select $a = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ \hat{Q}(a)$
  - With probability ϵ select a random action
- Constant ϵ ensures minimum regret

$$l_t \geq \frac{\epsilon}{\mathcal{A}} \sum_{a \in \mathcal{A}} \Delta_a$$

- $\Rightarrow$ ϵ-greedy has linear total regret

Optimistic Initialisation

Initialize values values to maximum possible, Q(a) = r_max
Then act greedily, A_t = argmax_{a \in A} Q_t (a)
Encourages exploration of unknown values
But a few unlucky samples can lock out optimal action forever
=> Optimistic greedy has linear total regret

- Simple and practical idea: initialise $Q(a)$ to high value
- Update action value by incremental Monte-Carlo evaluation
- Starting with $N(a) > 0$

$$\hat{Q}_t(a_t) = \hat{Q}_{t-1} + \frac{1}{N_t(a_t)}(r_t - \hat{Q}_{t-1})$$

- Encourages systematic exploration early on
- But can still lock onto suboptimal action
- $\Rightarrow$ greedy + optimistic initialisation has linear total regret
- $\Rightarrow$ ϵ-greedy + optimistic initialisation has linear total regret

Decaying ϵ_t-Greedy Algorithm

- Pick a decay schedule for $\epsilon_1, \epsilon_2, \ldots$
- Consider the following schedule

This is impossible schedule.
We cannot do this in practice
because we use advance knowledge v*
which we don't know.

$$c > 0$$
$$d = \min_{a|\Delta_a>0} \Delta_i$$
$$\epsilon_t = \min \left\{1, \frac{c|\mathcal{A}|}{d^2 t}\right\}$$

- Decaying ϵ_t-greedy has *logarithmic* asymptotic total regret!
- Unfortunately, schedule requires advance knowledge of gaps
- Goal: find an algorithm with sublinear regret for any multi-armed bandit (without knowledge of $\mathcal{R}$)

The only problem is that we don't know in advance what the schedule should be.

Lower Bound

- The performance of any algorithm is determined by similarity between optimal arm and other arms
- Hard problems have similar-looking arms with different means
- This is described formally by the gap Δ_a and the similarity in distributions $KL(\mathcal{R}^a || \mathcal{R}^a *)$

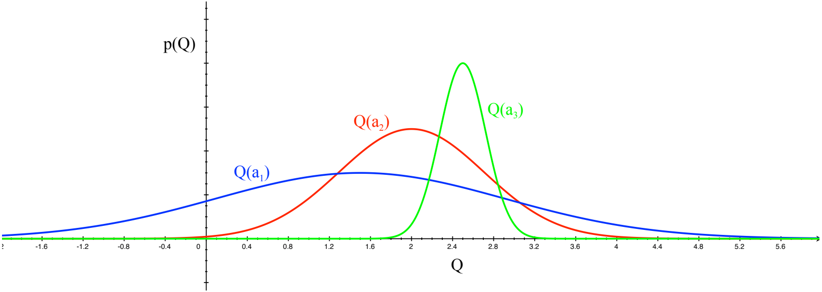
Theorem (Lai and Robbins)

Asymptotic total regret is at least logarithmic in number of steps

$$\lim_{t \to \infty} L_t \geq \log t \sum_{a | \Delta_a > 0} \frac{\Delta_a}{KL(\mathcal{R}^a || \mathcal{R}^{a^*})}$$

Optimism in the Face of Uncertainty

Main principle which we're going to use in next section

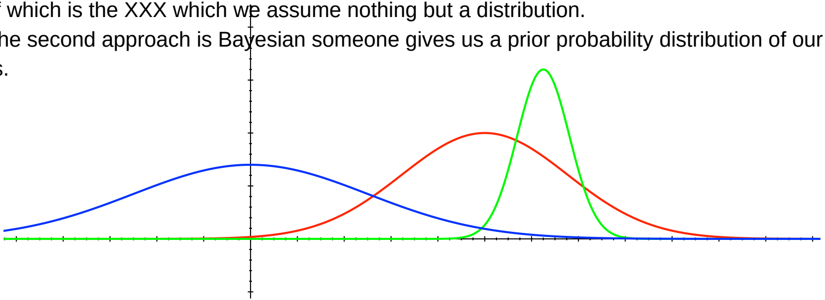


- Which action should we pick?
- The more uncertain we are about an action-value
- The more important it is to explore that action
- It could turn out to be the best action

The difficulty is that so far we've only talked about ways to estimate the mean (q values) and we haven't talked about ways to estimate the uncertainty

Optimism in the Face of Uncertainty (2)

So, we're gonna talk about two different approaches now to solving this approach,
one of which is the XXX which we assume nothing but a distribution.
And, the second approach is Bayesian someone gives us a prior probability distribution of our q values.



- After picking blue action
- We are less uncertain about the value
- And more likely to pick another action
- Until we home in on best action

Upper Confidence Bounds

The general idea that we're going to use is something called UCB (upper confidence bounds).

```
High probability confidence interval is where Q value can possibly be. We're going to pick
the thing with the highest upper confidence value. It basically means that true action
value Q(a) is less than upper confidence bound (95% confidence range(interval)).
```

- Estimate an upper confidence $\hat{U}_t(a)$ for each action value
- Such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with high probability
- This depends on the number of times $N(a)$ has been selected
  - Small $N_t(a) \Rightarrow$ large $\hat{U}_t(a)$ (estimated value is uncertain)
  - Large $N_t(a) \Rightarrow$ small $\hat{U}_t(a)$ (estimated value is accurate)
- Select action maximising Upper Confidence Bound (UCB)

Algorithm is really simple.

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \left(\hat{Q}_t(a) + \hat{U}_t(a) \right)$$

U value shrinks down in confidence range where Q value actually is, and eventually, U value shrinks to zero.

Optimism in the Face of Uncertainty에 의하면 best action (most potential action)은 variance가 큰 action을 선택하는 것이다. 즉, action에 대한 분포가 크면 초기에 exploration할 수 있는 action이 다양하다는 의미이고, 학습을 통해 uncertainty를 줄이면서 variance를 감소시킬 수 있다. 기대할 수 있는 optimal Q의 upper bound를 높게 잡는 것이다.

Hoeffding's Inequality

Theorem (Hoeffding's Inequality)

Let $X_1, ..., X_t$ be i.i.d. random variables in [0,1], and let $\overline{X}_t = \frac{1}{\tau}\sum_{\tau=1}^{t} X_\tau$ be the sample mean. Then

$$\mathbb{P}\left[\underbrace{\mathbb{E}\left[X\right]}_{\text{true mean}} > \underbrace{\overline{X}_t + u}_{\text{empirical mean}}\right] \leq e^{-2tu^2}$$

- We will apply Hoeffding's Inequality to rewards of the bandit
- conditioned on selecting action a

$$\mathbb{P}\left[Q(a) > \hat{Q}_t(a) + U_t(a)\right] \leq e^{-2N_t(a)U_t(a)^2}$$

Calculating Upper Confidence Bounds

- Pick a probability p that true value exceeds UCB
- Now solve for $U_t(a)$

$$e^{-2N_t(a)U_t(a)^2} = p$$

$$U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$$

- Reduce p as we observe more rewards, e.g. $p = t^{-4}$
- Ensures we select optimal action as $t \to \infty$

$$U_t(a) = \sqrt{\frac{2\log t}{N_t(a)}}$$

The denominator is the count, which means as we pick things more and more this bonus term is going to get pushed down to 0(zero).
As actions we try very often, we're going to have very large bonus term.

And, now what we will do is picking something like a schedule, what we actually want to do is to make sure that we actually guarantee that we can pick the optimal action as we continue.

So, the second thing we do is to schedule our p value. We don't fix it like 95%, instead what we do is that we slowly increase this thing over time to be more and more confident we've included our true value.

UCB1

You can just use this algorithm, empirical fact. This works very well in practice.
This is the one algorithm of many extensions and all kinds of different approaches.

- This leads to the UCB1 algorithm

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \left(Q(a) + \sqrt{\frac{2 \log t}{N_t(a)}} \right)$$

It only depends on time t.

Theorem

The UCB algorithm achieves logarithmic asymptotic total regret

gap

$$\lim_{t \to \infty} L_t \leq 8 \log t \sum_{a | \Delta_a > 0} \Delta_a$$

It doesn't have KL term and distribution term.

Example: UCB vs. ϵ-Greedy On 10-armed Bandit

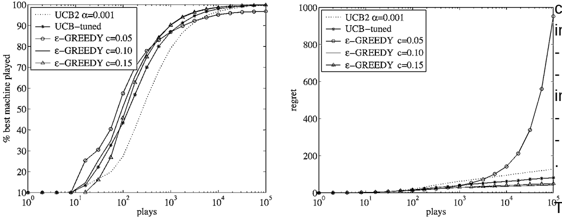


Figure 9. Comparison on distribution 11 (10 machines with parameters 0.9, 0.6, . . . , 0.6).

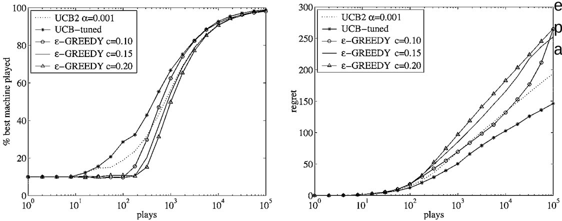


Figure 10. Comparison on distribution 12 (10 machines with parameters 0.9, 0.8, 0.8, 0.8, 0.7, 0.7, 0.7, 0.6, 0.6, 0.6).

Upper confidence bounds can be applied to other inequalities:
- Bernstein's inequality
- Empirical Bernstein's inequality
- Chernoff inequality
- Azuma's inequality
. . .

These ideas can be continued, so the inequality is that you can think of examples of general kind of procedure for generating algorithms.

The surprise is again that epsilon greedy does well. HOWEVER, the difficulty is that if it were wrong then it's disaster.

Bayesian Bandits

- So far we have made no assumptions about the reward distribution $\mathcal{R}$
  - Except bounds on rewards

e.g. indep. Gaussians:
w = [\mu_1, \sigma_1^2, ... , \mu_k, \sigma_k^2] for a \in [1, k]

Bayesian methods compute posterior distribution over w, p[w I R_1, ... , R_t]

- Bayesian bandits exploit prior knowledge of rewards, $p[\mathcal{R}]$

Consider a distribution p[Q|w] over action-value function with parameter w.

- They compute posterior distribution of rewards $p[\mathcal{R} \mid h_t]$
  - where $h_t = a_1, r_1, ..., a_{t-1}, r_{t-1}$ is the history
- Use posterior to guide exploration

Posteroir Distribution

  - Upper confidence bounds (Bayesian UCB)
  - Probability matching (Thompson sampling)
- Better performance if prior knowledge is accurate

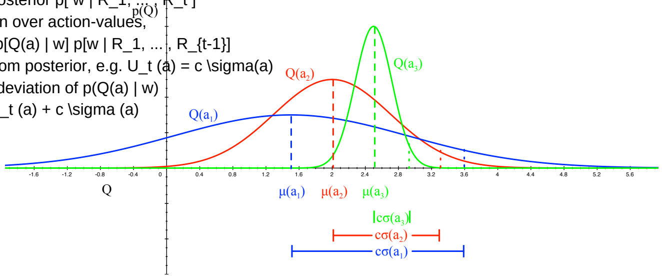
If prior knowledge is wrong, you'd probably better of using the UCB approach we just saw, which is robust different distribution exceptions.

Bayesian UCB Example: Independent Gaussians

How to can we use these Bayesian idea to compute our UCB?

- Assume reward distribution is Gaussian, $\mathcal{R}_a(r) = \mathcal{N}(r; \mu_a, \sigma_a^2)$

Use Bayes law to compute posterior p[w | R_1, ... , R_t]
Compute posterior distribution over action-values,
p[Q(a) | R_1, ... , R_{t-1}] = p[Q(a) | w] p[w | R_1, ... , R_{t-1}]
Estimate upper confidence from posterior, e.g. U_t (a) = c \sigma(a)
where \sigma(a) is standard deviation of p(Q(a) | w)
Pick action that maximizes Q_t (a) + c \sigma (a)



- Compute Gaussian posterior over μ_a and σ_a^2 (by Bayes law)

$$p\left[\mu_a, \sigma_a^2 \mid h_t\right] \propto p\left[\mu_a, \sigma_a^2\right] \prod_{t \mid a_t = a} \mathcal{N}(r_t; \mu_a, \sigma_a^2)$$

- Pick action that maximises standard deviation of $Q(a)$

$$a_t = \operatorname*{argmax}_a \mu_a + c\sigma_a / \sqrt{N(a)}$$

Probability Matching

There's the second way to make use of our probability distribution. If we computed all these posterior distributions over all action values, there's another way to make use of this information. And, this is also true for any Bayesian method.

- Probability matching selects action a according to probability that a is the optimal action

π(a) = P[Q(a) = max Q(a') | R_1, ... , R_{t-1}]

$$\pi(a \mid h_t) = \mathbb{P}\left[Q(a) > Q(a'), \forall a' \neq a \mid h_t\right]$$

This is heuristic idea that guides us to picking the action most which has the chance being the best.

automatically

- Probability matching is optimistic in the face of uncertainty
  - Uncertain actions have higher probability of being max
- Can be difficult to compute analytically from posterior

Thompson Sampling

oldest algorithm for bandit

- Thompson sampling implements probability matching

$$\begin{aligned}\pi(a \mid h_t) &= \mathbb{P}\left[Q(a) > Q(a'), \forall a' \neq a \mid h_t\right] \\ &= \mathbb{E}_{\mathcal{R}|h_t}\left[\mathbf{1}(a = \operatorname*{argmax}_{a \in \mathcal{A}} Q(a))\right]\end{aligned}$$

- Use Bayes law to compute posterior distribution $p\left[\mathcal{R} \mid h_t\right]$
- Sample a reward distribution $\mathcal{R}$ from posterior
- Compute action-value function $Q(a) = \mathbb{E}\left[\mathcal{R}_a\right]$
- Select action maximising value on sample, $a_t = \operatorname*{argmax}_{a \in \mathcal{A}} Q(a)$
- Thompson sampling achieves Lai and Robbins lower bound!

Value of Information

So far we've seen two of our three classes approach :
- Randomized Exploration Algorithm (randomly ε-greedy, random explore)
- Upper Confidence Algorithms, optimism in face of uncertainty
- Now the third is the Information State Space

Why is exploration useful?

If you just try some action, but then you weren't told how much reward you got from taking that action. That would be no point to explore. You wouldn't be learned from it.

- Exploration is useful because it gains information
- Can we quantify the value of information?
  - How much reward a decision-maker would be prepared to pay in order to have that information, prior to making a decision
  - Long-term reward after getting information - immediate reward

action에 따른 reward에 대한 정보량이 주어지면 우리는 정확하게 어떤 action을 취할 것인지 정할 수 있다.

- Information gain is higher in uncertain situations
- Therefore it makes sense to explore uncertain situations more
- If we know value of information, we can trade-off exploration and exploitation *optimally*

So, what's the real best way to trade-off exploration and exploitation.

Information State Space

Now, we transform our bandit problem back into an MDP.

- We have viewed bandits as *one-step* decision-making problems
- Can also view as *sequential* decision-making problems
- At each step there is an *information state* $\tilde{s}$

  This is what we know so far.
  - $\tilde{s}$ is a statistic of the history, $\tilde{s}_t = f(h_t)$
  - summarising all information accumulated so far

What we're going to do is that each time we've actually taken action, it's going to make a transition us.

- Each action a causes a transition to a new information state $\tilde{s}'$ (by adding information), with probability $\tilde{\mathcal{P}}^a_{\tilde{s},\tilde{s}'}$
- This defines MDP $\tilde{\mathcal{M}}$ in augmented information state space

We can define an MDP over information state.

normal reward space

normal action space

$$\tilde{\mathcal{M}} = \langle \tilde{\mathcal{S}}, \mathcal{A}, \tilde{\mathcal{P}}, \mathcal{R}, \gamma \rangle$$

large MDP

information state space

transition matrix

Example: Bernoulli Bandits

- Consider a Bernoulli bandit, such that $\mathcal{R}^a = \mathcal{B}(\mu_a)$

the reward is 1 or 0
just like coin flip

- e.g. Win or lose a game with probability μ_a
- Want to find which arm has the highest μ_a
- The information state is $\tilde{s} = \langle \alpha, \beta \rangle$
  - α_a counts the pulls of arm a where reward was 0
  - β_a counts the pulls of arm a where reward was 1

Solving Information State Space Bandits

- We now have an infinite MDP over information states
- This MDP can be solved by reinforcement learning
- Model-free reinforcement learning
  - e.g. Q-learning (Duff, 1994)
- Bayesian model-based reinforcement learning
  - e.g. Gittins indices (Gittins, 1979)
  - This approach is known as *Bayes-adaptive* RL
  - Finds Bayes-optimal exploration/exploitation trade-off with respect to prior distribution

This is why we spent rest of course on. So, we can apply our favorite method to this.

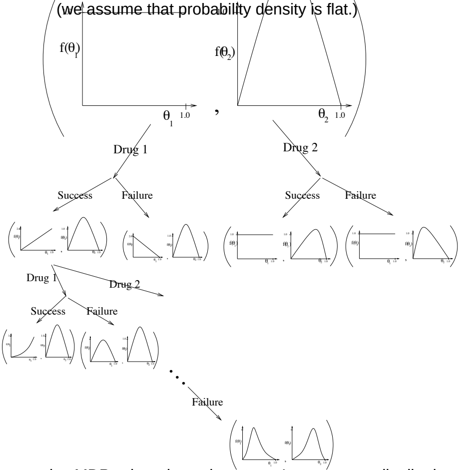
We can work in information states and there are many many different ways to work with information states.
If we characterize our information by a posterior distribution, then that's what's known as Bayes-adaptive RL.

Bayes-Adaptive Bernoulli Bandits

Beta Distribution
failure count

- Start with $Beta(\alpha_a, \beta_a)$ prior over reward function $\mathcal{R}^a$
success count
- Each time a is selected, update posterior for $\mathcal{R}^a$
  - $Beta(\alpha_a + 1, \beta_a)$ if $r = 0$
  - $Beta(\alpha_a, \beta_a + 1)$ if $r = 1$
- This defines transition function $\tilde{\mathcal{P}}$ for the Bayes-adaptive MDP
Bayes-adaptive approaches
- Information state $\langle \alpha, \beta \rangle$ corresponds to reward model $Beta(\alpha, \beta)$
- Each state transition corresponds to a Bayesian model update

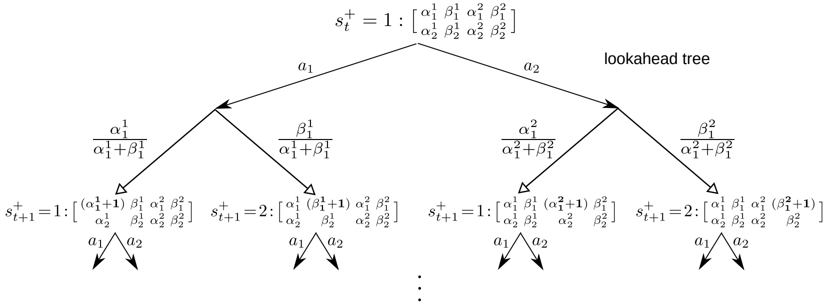
Drug example, two different drugs considering
First start off thinking that the probability is flat because we don't know what successful probability of this drug is.
(we assume that probability density is flat.)



We can solve MDP, where in each state we've got some distribution.

Bayes-Adaptive MDP for Bernoulli Bandits

This tells us how your distribution and success counts are changing as you move through tree.



Gittins Indices for Bernoulli Bandits

- Bayes-adaptive MDP can be solved by dynamic programming
- The solution is known as the *Gittins index*
- Exact solution to Bayes-adaptive MDP is typically intractable

Exactly solving the Bayes-adaptive MDP is typically intractable.

  - Information state space is too large
- Recent idea: apply simulation-based search (Guez et al. 2012)

Recent approaches have explored large-scale planning methods (e.g. Monte-Carlo tree search) to plan a lookahead tree and find the Bayes-optimal exploration-exploitation trade-off starting from the current information state.

  - Forward search in information state space
  - Using simulations from current information state

Summary of Multi-Armed Bandit Algorithms

* Random exploration
- ε-greedy
- Softmax
- Gaussian noise

* Optimism in the face of uncertainty
- Optimistic initialization
- UCB
- Thompson sampling (probability matching)

In infinite action space, we can be exploring over and over again (Problem of uncertainty)

Another problem is that 예기치 못한 탐험으로 큰 재앙이 일어날 수 있는 경우 (safety 조건이 중요하다.)

* Information state space
- Gittins indices
- Bayes-adaptive MDPs

Contextual Bandits

What David is going to do is to explain the Contextual Bandits problem, but he's not going to explain the solution method. He'll leave those further reading in the slides. (skip) He'll gonna move on to very briefly, just touch on ideas to extend to MDPs.

- A contextual bandit is a tuple $\langle \mathcal{A}, \mathcal{S}, \mathcal{R} \rangle$ from Multi-Arms bandit problem putting state back.
- $\mathcal{A}$ is a known set of actions (or "arms")

The KEY thing is that we've got the 'context'.

- $\mathcal{S} = \mathbb{P}[s]$ is an unknown distribution over states (or "contexts")
- $\mathcal{R}_s^a(r) = \mathbb{P}[r | s, a]$ is an unknown probability distribution over rewards
- At each step t
  - Environment generates state $s_t \sim \mathcal{S}$
  - Agent selects action $a_t \in \mathcal{A}$
  - Environment generates reward $r_t \sim \mathcal{R}_{s_t}^{a_t}$
- Goal is to maximise cumulative reward $\sum_{\tau=1}^{t} r_\tau$

Linear Regression

- Action-value function is expected reward for state s and action a

$$Q(s,a) = \mathbb{E}\left[r | s, a\right]$$

- Estimate value function with a linear function approximator

$$Q_\theta(s,a) = \phi(s,a)^\top \theta \approx Q(s,a)$$

- Estimate parameters by least squares regression

$$A_t = \sum_{\tau=1}^{t} \phi(s_\tau, a_\tau)\phi(s_\tau, a_\tau)^\top$$

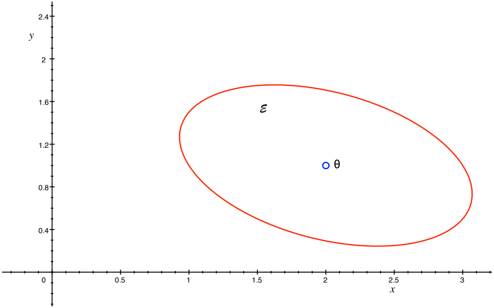
$$b_t = \sum_{\tau=1}^{t} \phi(s_\tau, a_\tau) r_\tau$$

$$\theta_t = A_t^{-1} b_t$$

Linear Upper Confidence Bounds

- Least squares regression estimates the mean action-value $Q_\theta(s, a)$
- But it can also estimate the variance of the action-value $\sigma^2_\theta(s, a)$
- i.e. the uncertainty due to parameter estimation error
- Add on a bonus for uncertainty, $U_\theta(s, a) = c\sigma$
- i.e. define UCB to be c standard deviations above the mean

Geometric Interpretation



- Define confidence ellipsoid $\mathcal{E}_t$ around parameters θ_t
- Such that $\mathcal{E}_t$ includes true parameters θ^* with high probability
- Use this ellipsoid to estimate the uncertainty of action values
- Pick parameters within ellipsoid that maximise action value

$$\underset{\theta \in \mathcal{E}}{\text{argmax}}\; Q_\theta(s, a)$$

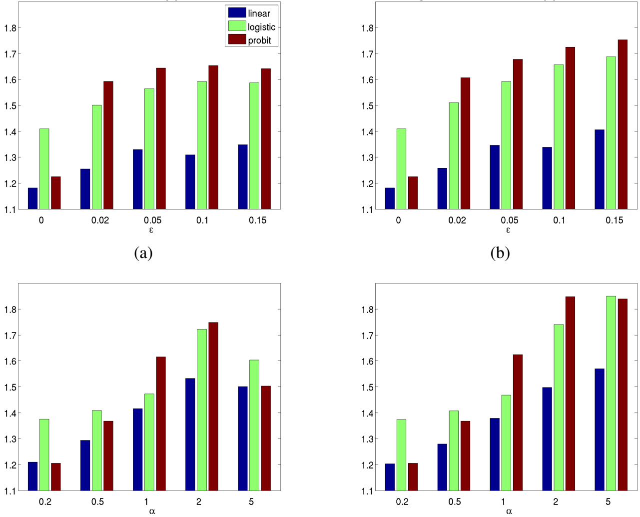
Calculating Linear Upper Confidence Bounds

- For least squares regression, parameter covariance is A^{-1}
- Action-value is linear in features, $Q_\theta(s, a) = \phi(s, a)^\top \theta$
- So action-value variance is quadratic, $\sigma^2_\theta(s, a) = \phi(s, a)^\top A^{-1} \phi(s, a)$
- Upper confidence bound is $Q_\theta(s, a) + c\sqrt{\phi(s, a)^\top A^{-1} \phi(s, a)}$
- Select action maximising upper confidence bound

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \; Q_\theta(s_t, a) + c\sqrt{\phi(s_t, a)^\top A_t^{-1} \phi(s_t, a)}$$

Example: Linear UCB for Selecting Front Page News

UCB approach is extended to context using linear function approximation



(a) (b)

(c) (d)

Exploration/Exploitation Principles to MDPs

How can we take the ideas we've seen so far and extend them, the full case, that we care about to really build agents. We want 'reinforcement learning' and understand how to trade-off exploration and exploitation, so to find the best solution whatever problem we addressing.

The same principles for exploration/exploitation apply to MDPs

- Naive Exploration
- Optimistic Initialisation
- Optimism in the Face of Uncertainty
- Probability Matching
- Information State Search

One thing David stresses about this is that this idea is not quite perfect in MDPs because this ignores very important fact, which is that we're just evaluating a current policy and that policy is likely to improve. If we're doing control the MDP and we're going to start to improve our policy, the Q values are actually better and better.

So, the uncertainty correctly should be taken into account.

Q values could be wrong in two ways because we evaluate using current state but there's lots of improvement we still make.

Optimistic Initialisation: Model-Free RL

- Initialise action-value function $Q(s, a)$ to $\frac{r_{max}}{1-\gamma}$
- Run favourite model-free RL algorithm
  - Monte-Carlo control
  - Sarsa
  - Q-learning
  - ...
- Encourages systematic exploration of states and actions

Optimistic Initialisation: Model-Based RL

One successful approach to exploration / exploitation in model-based RL
Construct a model of the MDP.
For unknown or poorly estimated states, replace reward function with r_max
i.e. Be (very) optimistic in the face of uncertainty.

- Construct an optimistic model of the MDP
- Initialise transitions to go to heaven
  - (i.e. transition to terminal state with r_{max} reward)
- Solve optimistic MDP by favourite planning algorithm
  - policy iteration
  - value iteration
  - tree search
  - ...
- Encourages systematic exploration of states and actions
- e.g. RMax algorithm (Brafman and Tennenholtz)

Upper Confidence Bounds: Model-Free RL

This is just say that all of the ideas we've seen in previous sections are extended to MDP case, everyone.

The UCB approach can be generalized to full MDPs. To give a model-free RL algorithm,

- Maximise UCB on action-value function $Q^{\pi}(s, a)$

One thing David stresses about this is that this idea is not quite perfect in MDPs because this ignores very important fact, which is that we're just evaluating a current policy and that policy is likely to improve. If we're doing control the MDP and we're going to start to improve our policy, the Q values are actually better and better. So, the uncertainty correctly should be taken into account. Q values could be wrong in two ways because we evaluate using current state but there's lots of improvement we still make.

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ Q(s_t, a) + U(s_t, a)$$

For example, Kaelbling's interval estimation (table lookup)
LSTD with confidence ellipsoid (linear function approximation)

  - Estimate uncertainty in policy evaluation (easy)
  - Ignores uncertainty from policy improvement

- Maximise UCB on optimal action-value function $Q^{*}(s, a)$

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ Q(s_t, a) + U_1(s_t, a) + U_2(s_t, a)$$

  - Estimate uncertainty in policy evaluation (easy)
  - plus uncertainty from policy improvement (hard)

Bayesian Model-Based RL

- Maintain posterior distribution over MDP models
- Estimate both transitions and rewards, $p\left[\mathcal{P}, \mathcal{R} \mid h_t\right]$
  - where $h_t = s_1, a_1, r_2, ..., s_t$ is the history
- Use posterior to guide exploration
  - Upper confidence bounds (Bayesian UCB)
  - Probability matching (Thompson sampling)

Thompson Sampling: Model-Based RL

- Thompson sampling implements probability matching

$$\begin{aligned}\pi(s, a \mid h_t) &= \mathbb{P}\left[Q^*(s, a) > Q^*(s, a'), \forall a' \neq a \mid h_t\right] \\ &= \mathbb{E}_{\mathcal{P},\mathcal{R} \mid h_t}\left[\mathbf{1}(a = \operatorname*{argmax}_{a \in \mathcal{A}} Q^*(s, a))\right]\end{aligned}$$

- Use Bayes law to compute posterior distribution $p\left[\mathcal{P}, \mathcal{R} \mid h_t\right]$
- Sample an MDP $\mathcal{P}, \mathcal{R}$ from posterior
- Solve MDP using favourite planning algorithm to get $Q^*(s, a)$
- Select optimal action for sample MDP, $a_t = \operatorname*{argmax}_{a \in \mathcal{A}} Q^*(s_t, a)$

Information State Search in MDPs

- MDPs can be augmented to include information state
- Now the augmented state is $\langle s, \tilde{s} \rangle$
  - where s is original state within MDP
  - and $\tilde{s}$ is a statistic of the history (accumulated information)
- Each action a causes a transition
  - to a new state s' with probability $\mathcal{P}^a_{s,s'}$
  - to a new information state $\tilde{s}'$
- Defines MDP $\tilde{\mathcal{M}}$ in augmented information state space

$$\tilde{\mathcal{M}} = \langle \tilde{\mathcal{S}}, \mathcal{A}, \tilde{\mathcal{P}}, \mathcal{R}, \gamma \rangle$$

Bayes-adaptive approach maintains a posterior model corresponding to each augmented state.
Solving the Bayes-adaptive MDP finds the optimal exploration/exploitation trade-off with respect to prior.
However, augmented MDP is typically enormous.
Monte-Carlo tree search has proven effective here. (Guez et al.)

Bayes Adaptive MDPs

- Posterior distribution over MDP model is an information state

$$\tilde{s}_t = \mathbb{P}\left[\mathcal{P}, \mathcal{R} | h_t\right]$$

- Augmented MDP over $\langle s, \tilde{s} \rangle$ is called Bayes-adaptive MDP
- Solve this MDP to find optimal exploration/exploitation trade-off (with respect to prior)
- However, Bayes-adaptive MDP is typically enormous
- Simulation-based search has proven effective (Guez et al.)

Conclusion

- Have covered several principles for exploration/exploitation
  - Naive methods such as ϵ-greedy
  - Optimistic initialisation
  - Upper confidence bounds
  - Probability matching
  - Information state search
- Each principle was developed in bandit setting
- But same principles also apply to MDP setting

Progressively more complicated approaches to exploration/exploitation :
- Random exploration
- Optimism in the face of uncertainty
- Information state space