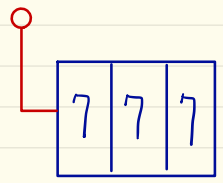
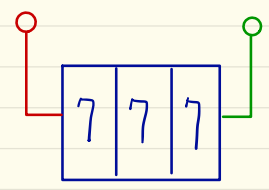
Part V Multi-Armed Bandit

∙ Slot machine: also known as one-armed bandit



It’s a bandit because it supposes to rob your money

∙ k-armed bandit



Total k arms, each arm has its own expected reward

∙ Expected rewards are unknown

∙ You are going to play the game a finite number T of rounds

How to maximize your expected reward in the end of T rounds?

This seemingly artificially problem has tons of real-life applications.

Moreover, it captures the fundamental essence of reinforcement learning.

Applications

∙ Gambling: Easy to see.

∙ Drug tests: William Thompson 1933.

Two experimental drugs, which one to give to the patients?

patient dies => reward 0

patient recovers => reward 1

Be careful, human lives are at stake!

∙ Recommender systems: Netflix recommendation

Each user’s front page is customized to improve user experience.

It is a very complex problem whose answer usually involves user’s race, age, gender, watch history, etc.

But its simplest version can be modeled as follows,

This is a multi-billion dollar business!

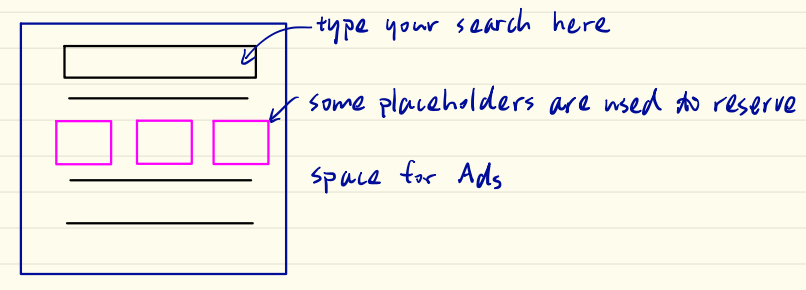
∙ Placing Ads on Google

Google is not a charity. How does Google make money?

Placing Ads on Google search. Google gets paid only when users click on Ads. You better place relevant Ads.

Again, it depends on a lot of things.

Nowadays, Google uses online auction to place Ads.



Your search text and other information are sent to potential bidders (bidding companies) and an online auction takes place.

This again can be modeled as a multi-armed bandit problem.

Unsupervised learning: Given a bunch of data, find hidden features in data.

No feedback.

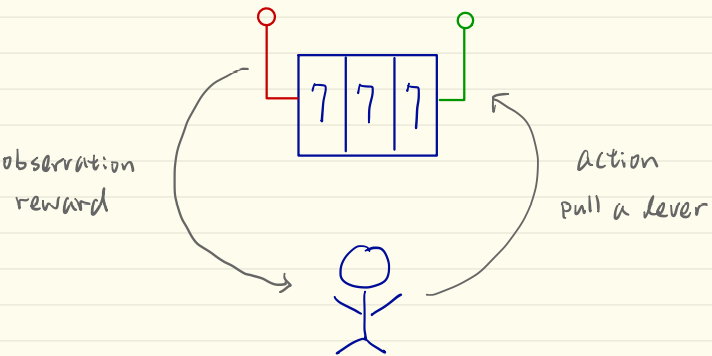
Supervised learning: Given a bunch of data with correct labels, find a function that best mimics underlying labeling.

Instructive feedback: Tells you correct action to take, independent of what action you took.

New learning framework: You get to interact with environment.

You pick an action (pull a lever), observation depends on your action (reward), which is used to evaluate the action.

Evaluative feedback: Indicates how good the action taken was.



Formal problem statement: k-armed bandit

k possible actions, each action is associated with a reward distribution, which we don’t know.

A­­t: the action selected at time t.

Rt: reward received at time t, drawn from the corresponding reward distribution

: Expected reward given a is selected

: estimate of at time t

Remark: If we know the reward distributions, can be easily computed and best action is . However, they are unknown and we resort to sample mean, which converges to ensemble average by weak law of large numbers.

Def (Convergence) Given a sequence of RVs X1, X2, …, we say that the sequence x1, x2, … converge to a RV x:

1) in probability if for every , there exists a large enough n,

s.t. . i.e.,

2) in mean square if

3) with probability 1(or almost surely) if

Here, we focus on convergence in probability.

Review Weak Law of Large Numbers (WLLN):

Let X1, X2, …, Xn be a sequence of i.i.d. RV with and

Define

Thm: (WLLN)

in probability.

That is, for any ,

Pf:

Note that .

Also,

By Chebyshev inequality, for any ,

=> =>

Sample average converges to ensemble average.

Two mentalities:

1. Exploration: Explore each arm a reasonable number of times so that the empirical distribution is fairly accurate.

Rounds spent for exploration may have small rewards.

2. Exploitation: Exploit the current best arm in order to maximize the current reward.

Some other not sufficiently explored arms may be the true best

Tradeoff: Exploration v.s. Exploitation

Greedy Algorithm

Always select the action with highest estimated reward

i.e., Always exploit the current knowledge to maximize immediate reward.

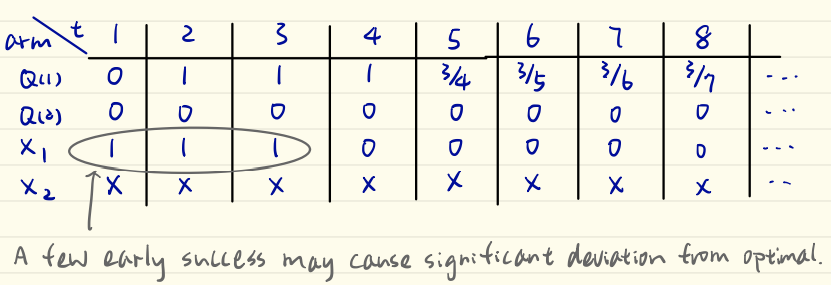
Not a good idea!

Ex: Two-armed Bernoulli bandit

Two coins

When arm is pulled,

Now, suppose , .



It happens with not so small probability.

Of course this can be alleviated by a few early explorations.

But still, there is a non-vanishing probability that you stuck in a bad arm and end up using it forever.

-Greedy Algorithm

Adopt greedy algorithm with probability

Perform random exploration with probability

This approach equips Greedy algorithm with random exploration.

But, we end up pulling the best arm fraction of times.

Also, when exploring, actions are treated indiscriminately, regardless our confidence about actions.

Upper Confidence Bound Algorithm

Select arms according to their potential for actually being optimal. For ,

(1) (2)

: number of times action has been pulled prior to .

∙ For s.t. , => every action must be explored once

∙ For those less explored, is large => encourage exploring

∙ As increases , the increase of gets slower => concentrates to best arm

∙ Actions with higher (1) rewards or significantly (2) less visits are preferred.

Question: How was chosen?

Thm. Confidence interval in law of large numbers

--- (\*)

Pf: Define

M.G.F.

Here, we assume sub-Guassian distributions. These include Guassian, Bernoulli, distributions with bounded support, etc.

=> =>

set =>

Now, setting completes the proof

Picks => (\*) ensures

We are quite certain (w.p. that our sample mean is not far away (within confidence interval) from .

So (2) in UCB measures uncertainty of our .

If (2) is large => we are uncertain and should explore more.

Remark: UCB performs better than -greedy.

But it is not suitable for nonstationary problems. i.e., problems whose parameters vary (slowly) over time.

Thompson Sampling (William Thompson 1933)

Consider Bernoulli bandit where

Instead of estimating , we obtain a distribution over possible values of

∙ stores the number of times

∙ stores the number of times

Suppose the prior distribution of follows Beta distribution.

where denotes the gamma function.

The arm is pulled and we observe

Similarly, if ,

We only have to keep tracking & for updating prior.

Thompson Sampling for Bernoulli Bandit

Initial for all

For

For

Sample

End

Observe reward

Update ,

End

Remarks

1) representing maximum uncertainty at beginning

2) As the distribution becomes more and more concentrated towards its mean

3) It has recently attracted a lot of attention

4) It can be easily generalized to general bandit problem by replacing Beta distribution to suitable prior distribution.

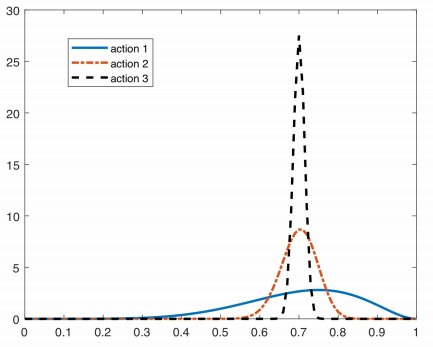
What do I mean by suitable?

Bernoulli reward & Beta prior are conjugate pair.

There are many other such pairs.

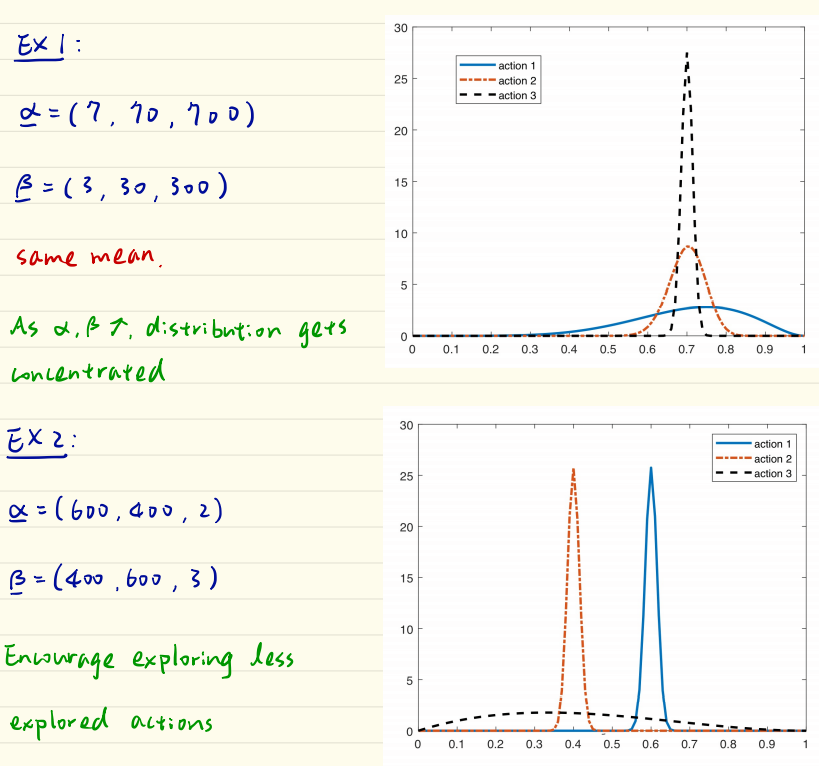
Ex1:

same mean.

As , distribution gets concentrated. 

Ex2:

Encourage exploring less explored actions.



Incremental Implementation

Many algorithms require . How to compute it efficiently?

For simplicity, let’s focus on single action.

: reward received after round

: estimate after this action being selected times

So,

While updating, we are moving toward the target can be understood as error in our old estimate.

It admits a recursive form and is easy to implement.

Tracking Non-Stationary Problem

Many applications are non-stationary => expected rewards may vary (slowly) over time. In such cases, it makes sense to give higher weights to more recent data.

Constant step-size

∙ Total weight

∙ The earlier observed rewards , the smaller weight

∙ In general, step-size can vary with n, i.e.,

For example, in sample average.

It all depends on applications.

Contextual Bandits

Many other names: bandits with side information, learning with partial feedback, associative search, etc.

Main idea: Contexts do matter!

In many applications, we are given many side information such as user’s age, gender, etc.

Multi-Armed Bandit ignores those information.

In contextual bandit, given side info, we are restricted to a state.

For example, states determined by age & gender

} => 4 states

Associated with each state is a multi-armed bandit problem.

We can use an algorithm to learn best action.

Overall, we are learning a “policy” that associates each state a good action.

Remarks:

1) Multi-Armed Bandit does not have the notation of states. The interaction between player & environment is built on action/reward.

MAB is reinforcement learning (RL) with single state.

2) Contextual Bandit is a step towards RL by incorporating states. But the states are given to you and your actions do not affect the next step.

Many applications belong to this categories.

3) RL in general has many states where each state can have its own action sets, reward functions, and most distinctly, the next step depends crucially on the action currently taken.

The interaction is through action/reward/state.