QBUS6850 Week 11

Recommendation Systems
Multi-Armed Bandits 2

Dr Stephen Tierney

The University of Sydney Business School

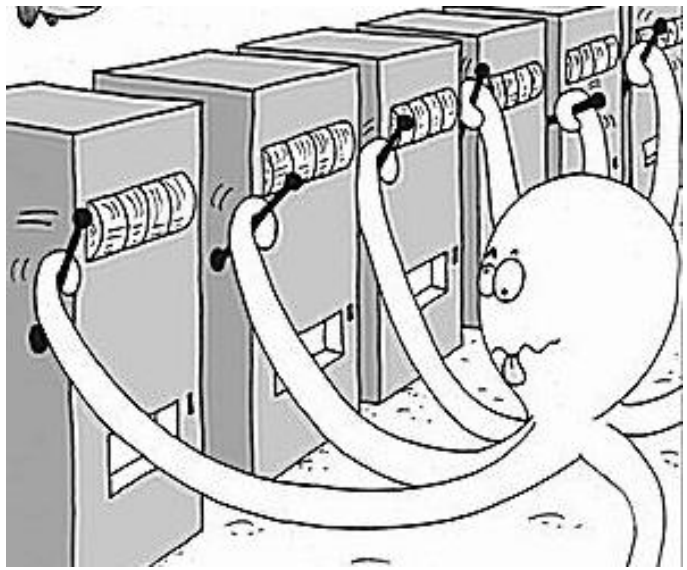
Reading

- Chapter 3 (skip the maths) Introduction to multi-armed bandits - A. Slivkins

Multi-Armed Bandits

Suppose you have a set of slot machines (one-armed bandits). Each machine has a **hidden** probability p_k of giving a positive return.

The goal is to maximise your returns when playing the machines.



Policies

Last Week

So far we've seen:

- a baseline policy that explores each bandit in sequence
- the epsilon greedy policy

Making Informed Plays

How can we improve on these simple algorithms?

Goal: sample as few times as possible.

Solution: only explore the bandits we are uncertain about, we do not need to explore the others.

Bayesian Inference

How do we model the uncertainty we have about a bandit?

We need a model for the probability of p_k.

Using **Bayes Rule**

$$P(p_k|x) = \frac{P(x|p_k)P(p_k)}{P(x)}$$

we can estimate a distribution for p_k.

Each time we make a play we can update the posterior with new information, using the old posterior as our prior.

Bayesian Inference

If we set that

- the likelihood $P(x|p_k)$ is Bernoulli distributed
- the prior $P(p_k)$ is Beta distributed

then the posterior $P(p_k|x)$ is also beta distributed.

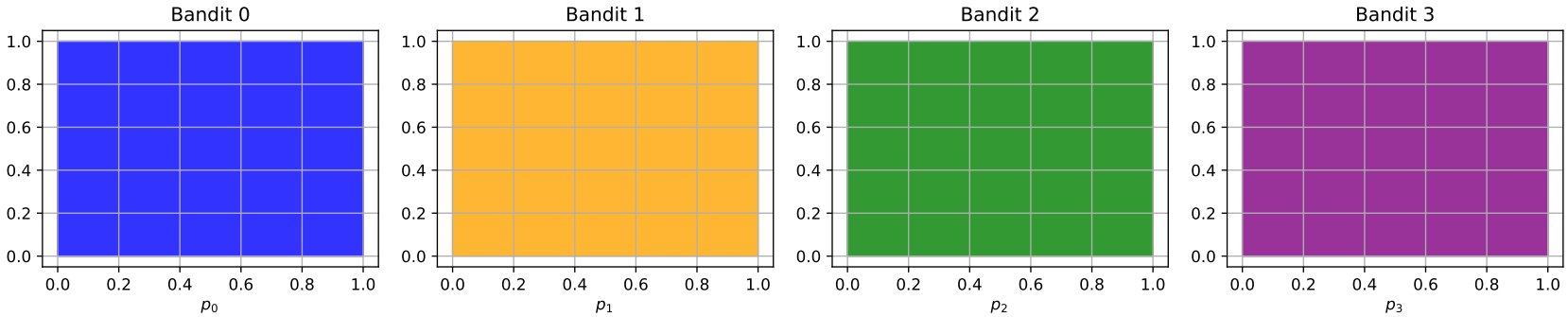
This is because the Beta distribution is a conjugate prior of the Bernoulli distribution.

Conveniently[1] the posterior is just $\sim Beta(1+\alpha, 1+\beta)$ where α and β are the number of observed wins and losses respectively.

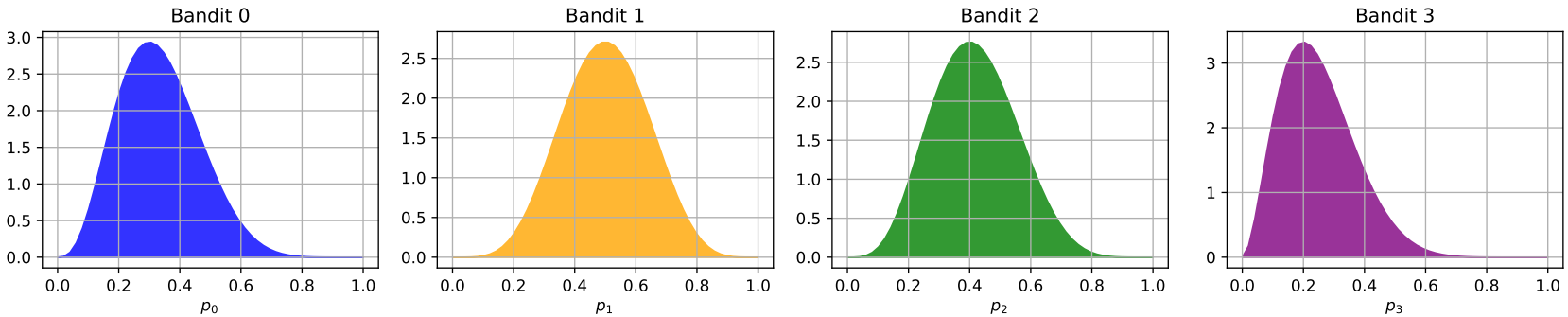
[1] skipping the maths

Bayesian Inference

After 0 draws

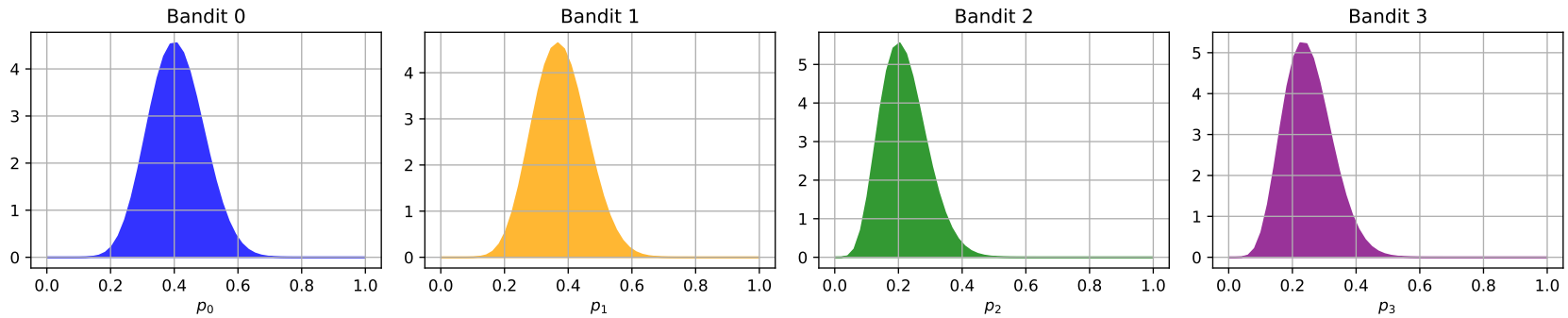


After 10 draws

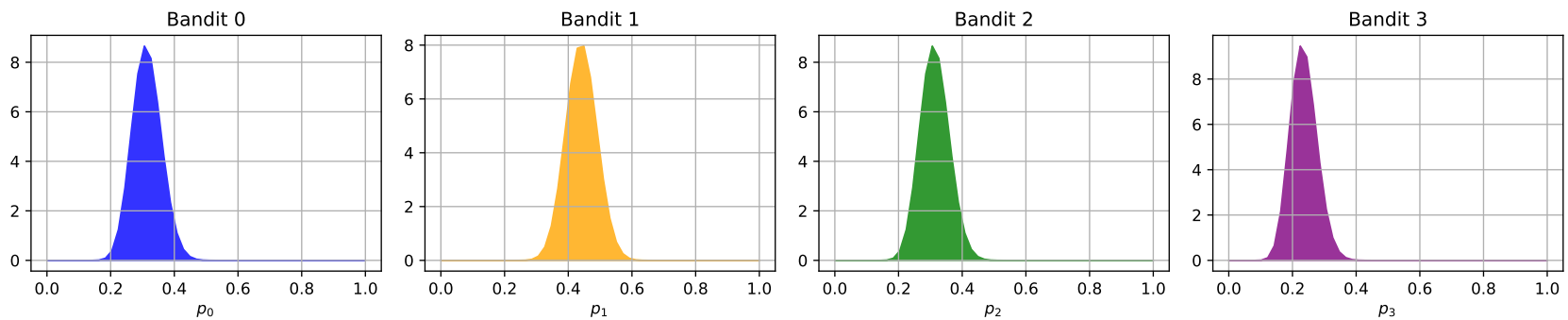


Bayesian Inference

After 30 draws



After 100 draws



Multi-Armed Bandits - Standard Algorithms

Equipped with our Bayesian toolkit we will now look at two algorithms:

- UCB
- Thompson Sampling

Multi-Armed Bandits - UCB

Concept: The bandits that we are least certain about will have the widest posterior distribution. At the same time, a high upper confidence bound indicates a potentially high probability.

The Upper Confidence Bound (UCB) strategy always picks the bandit that has the greatest UCB, which is given by

$$\hat{p}_k + \sqrt{\frac{2\ln n}{n_k}}$$

Therefore UCB **adaptively explores** the bandits. Eventually our estimated confidence bounds will shrink and the bandits with highest payoff will be identified for us to exploit.

Video Demonstration

Multi-Armed Bandits - UCB

The formula for the UCB comes from a paper which is discussed in in this set of slides

The formula applies to all distributions and is therefore generic enough for all types of reward outcomes, not just Bernoulli.

In our case we could calculate the sample confidence bound by using the standard deviation of the Beta distribution. But we will stick to the generic UCB estimation.

Multi-Armed Bandits - UCB

```
np.random.seed(0)

bandit_probs = [0.35, 0.40, 0.30, 0.25]

n_bandits = len(bandit_probs)
n_turns = 100

results = np.zeros((n_bandits, n_turns))
play_counts = np.zeros(n_bandits)

# Play each bandit once (initialise)
for k in range(n_bandits):
    results[k, 0] = np.random.binomial(1, bandit_probs[k], 1)
    play_counts[k] += 1

for i in range(1, n_turns):
    # Calculate UCB of all bandits
    est_probs = np.sum(results, axis=1) / play_counts
    ucbs = est_probs + np.sqrt( 2*np.log(i) / play_counts )

    # Pick biggest UCB
    chosen_idx = np.argmax(ucbs)

    results[chosen_idx, i] = np.random.binomial(1, bandit_probs[chosen_idx], 1)
    play_counts[chosen_idx] += 1

print(est_probs.round(2))
print(play_counts)
```

```
[0.32 0.35 0.12 0.3 ]
[28. 32. 16. 27.]
```

Multi-Armed Bandits - Thompson Sampling

The Thompson Sampling strategy relies on **probability matching**, which means that we pick a bandit which has highest probability of being optimal.

We can determine which bandit has the highest probability of being optimal by drawing a sample from the posterior of $\hat{p}_k$. In summary the strategy is:

1. update our posterior of $\hat{p}_k$
2. draw a single sample from each $\hat{p}_k$
3. pick the bandit k with the largest sample value

This shares similarities with the UCB method in that we adaptively explore based on uncertainty/confidence.

Video Demonstration

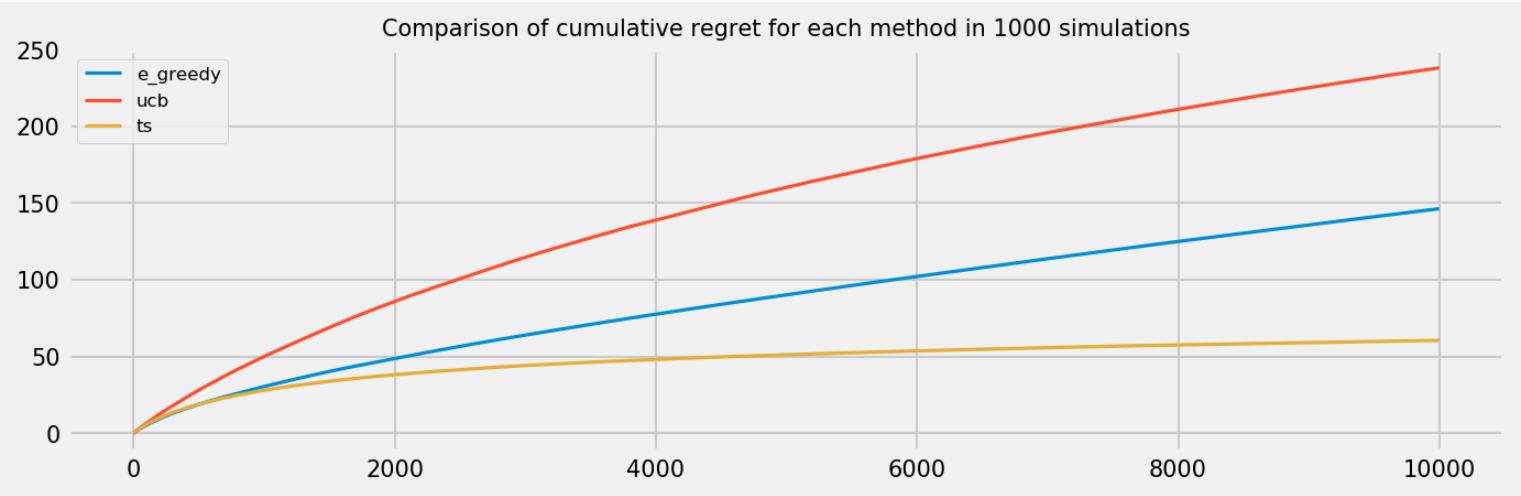
Multi-Armed Bandits - Thompson Sampling

```
np.random.seed(0)

bandit_probs = [0.35, 0.40, 0.30, 0.25]

n_bandits = len(bandit_probs)
n_turns = 10000

results = np.zeros((n_bandits, n_turns))
play_counts = np.zeros(n_bandits)

for i in range(n_turns):

    wins = np.sum(results, axis=1)
    losses = play_counts - wins

    posterior_samples = [beta.rvs(1 + wins[k], 1+ losses[k]) for k in range(n_bandits)]

    chosen_idx = np.argmax(posterior_samples)

    results[chosen_idx, i] = np.random.binomial(1, bandit_probs[chosen_idx], 1)
    play_counts[chosen_idx] += 1

est_probs = np.sum(results, axis=1) / play_counts
```

Policy Comparison

When we compare these algorithms we should do so using Regret, and if doing a simulation study, should do many runs and average the results.

The winner is Thompson Sampling in this example.



Contextual Bandits

Contextual Bandits

In some circumstances there might be factors that affect the probability of reward. These factors are known as **context**. This just means changing our model from $P(Y = 1)$ to $P(Y = 1|X = \mathrm{x})$.

To understand this, imagine that the probability of reward varies based on the state of the system.

Example: a slot machine pays out more or less at a particular times of day.

The process is much the same as before:

1. make an action based on policy
2. update (re-train) our estimate of model parameters $\boldsymbol{\beta}$ for selected bandit

Contextual Bandits - Thompson Sampling

The simplest example is to use Logistic Regression

$$P(Y = 1|X = \mathrm{x}) = p_k = \frac{1}{1 + \exp^{(-\mathrm{X}\boldsymbol{\beta})}}$$

Thompson Sampling requires that we draw a sample from the posterior. In this case the posterior is $P(\boldsymbol{\beta}|x)$.

We can assume that each $\beta_i \sim \mathcal{N}(m_i, q_i^{-1})$ i.e. it is normally distributed with some unknown mean and variance.

To make a play we generate a sample of $\boldsymbol{\beta}$ for each bandit then plug it into the formula above to estimate the probability of success. We choose the bandit with the highest probability.

Offline Evaluation

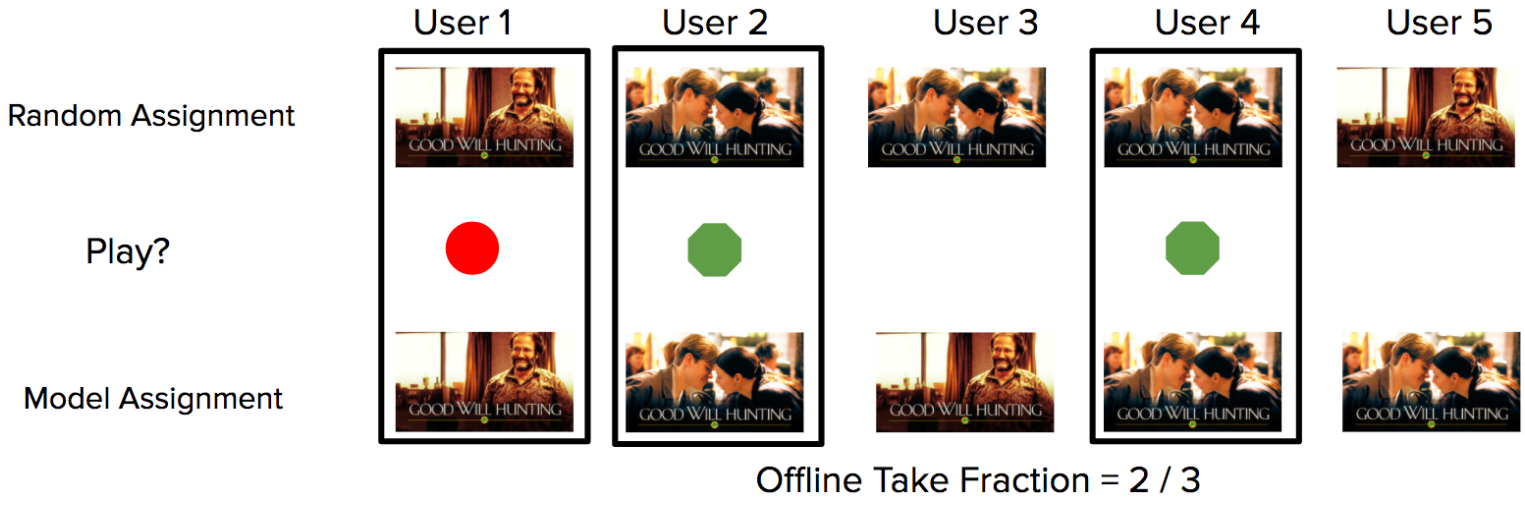
Offline Evaluation

To evaluate which MAB policy is best we need a way to evaluate them offline using historical data.

The most common method is known as **Replay**, which allows you to calculate rewards in hypothetical situations, rather than just the original recommendations that the user was presented with.

http://proceedings.mlr.press/v26/li12a/li12a.pdf

Offline Evaluation



Wrapping Up

Extensions

Multi-Armed Bandits don't stop here:

- **Non-binary rewards** - rewards are often numeric e.g. financial, which we should model. For example we might choose to model the rewards using a Normal distribution.
- **Non-stationary bandits** - the real world is dynamic, however we assume that the bandit parameters are fixed and tend to ignore some bandits after we determine that they have low payouts. Our model should consider the decay in our model estimate and refresh bandits accordingly.

Where to learn more?

Tutorials:

- Contextual Bandits - https://gdmarmerola.github.io/ts-for-contextual-bandits/
- Gaussian Rewards - https://gdmarmerola.github.io/approximate-bayes-bandits/
- Non-stationary Bandits - https://gdmarmerola.github.io/non-stationary-bandits/
- General Overview - https://lilianweng.github.io/lil-log/2018/01/23/the-multi-armed-bandit-problem-and-its-solutions.html