QBUS6850 Week 10

Recommendation Systems
Multi-Armed Bandits 1

Dr Stephen Tierney

The University of Sydney Business School

Reading

- Chapter 1 (skip the maths) Introduction to multi-armed bandits - A. Slivkins
- Artwork Personalization at Netflix

Recommendation Systems - Previously

The techniques discussed so far are far from perfect:

- user taste is **non-stationary** so the predictions we make will be stale
- hard to keep the models updated since they are not designed to be trained **online**
- often they suffer from **mode collapse** so customers are always recommended the same items or all customers get the same recommendations

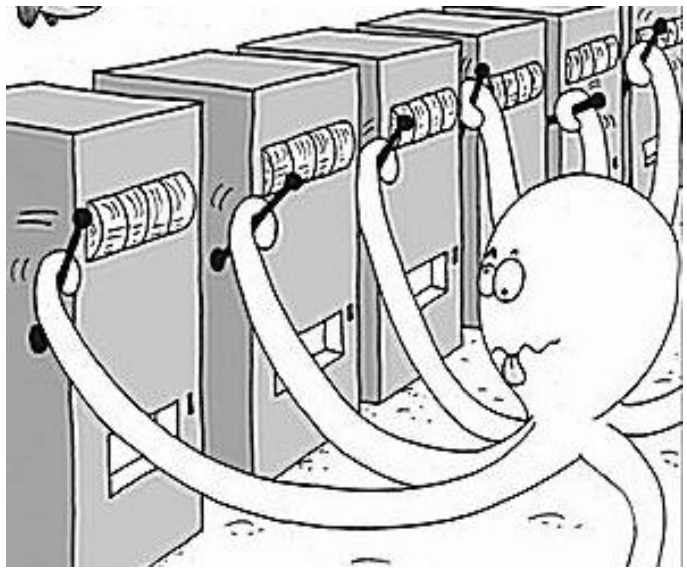
Making accurate predictions that are temporal and are novel will likely create more value to the user.

Multi-Armed Bandits

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.



Multi-Armed Bandits Model

We will assume binary outcomes (win or lose) for this lecture.

The full specification of the **model** is:

- there are k machines
- each machine has its own independent reward probability and is $\sim$ Bernoulli(p_k)
- at each time step t we can only take one action and receive an outcome r_t
- we stop after time T
- the list of actions (chosen machines) is $A = [a_1, a_2, \ldots, a_T]$

Multi-Armed Bandits Objective

Our goal can be expressed as

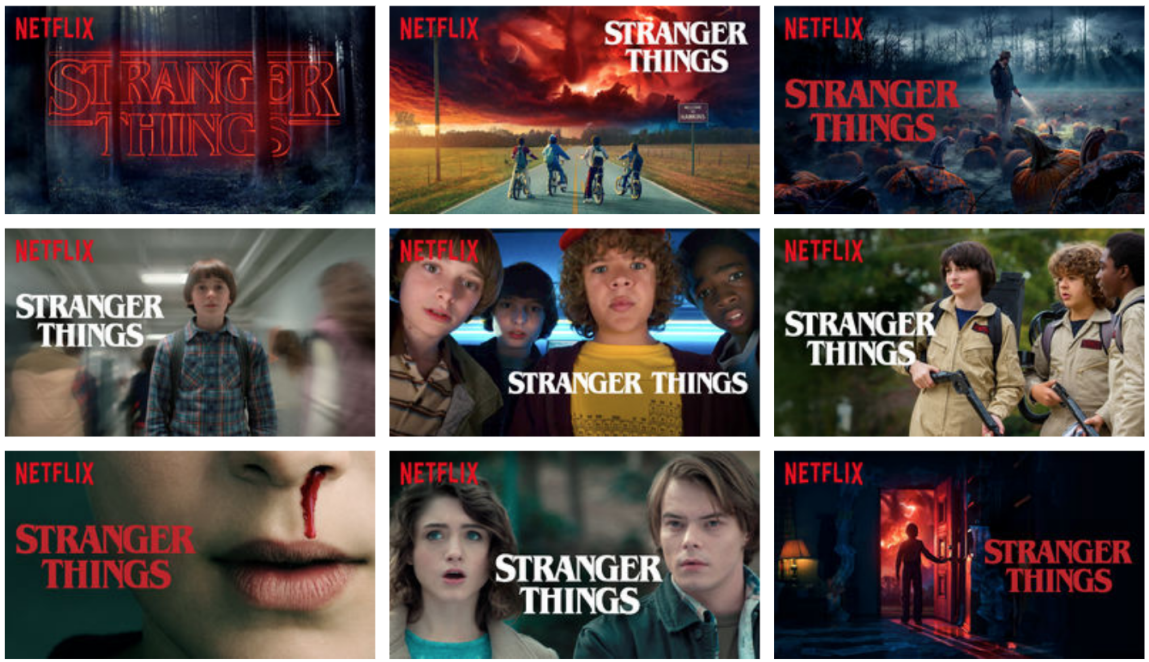
$$\max_{A} \sum_{t=1}^{T} r_t$$

which means

choose a sequence of actions that maximises the total reward over the T steps

Multi-Armed Bandits - Problem Framing

Netflix has publicly revealed that they use MAB for customising the artwork shown to users. The goal is to maximise the probability that they will view the title.



My suspicion is that Netflix now uses this approach for everything recommendation/personalisation related.

https://medium.com/netflix-techblog/artwork-personalization-c589f074ad76

Multi-Armed Bandits - Problem Framing

How can we pose artwork personalisation as a Multi-Armed Bandits Problem?

Multi-Armed Bandits - Solution

Multi-Armed Bandits is only a modelling paradigm or assumption. It says nothing about how we should learn the hidden distribution of each slot machine.

There's no training data and therefore no possibility of an analytic/closed-form solution.

The only way to learn about the slot machines is to play them.

The way we play is called either the **strategy** or **policy**.

The Whole Picture

We can pose recommendation as a MAB problem:

- each product is a bandit
- interacting with a product (click, watch or purchase) is a success, while no interaction is a failure
- the recommended items are a ranked list of the bandits with highest probability

Alternative presentations might e.g categorising each slot in the recommended list, in which case we would have multiple MAB models, one for each category.

Simple Policies

Exploitation-Exploration Tradeoff

The goal is to learn the underlying distributions as quickly as possible.

To learn the distributions we have to **explore** the bandits, however this might lead to wasted plays.

On the other hand if we decide to **exploit** or focus on a bandit too early then we might be following a sub-optimal strategy.

Baseline Policy

One of the simplest policies is to explore first and then exploit.

The policy is:

1. Try each bandit N times (explore stage)
2. Estimate reward probability of each bandit
3. Play the bandit with with the estimated highest reward probability (exploit stage)

Baseline Policy

Calculate Reward Probability

$$\hat{p}_k = \sum_{i=1}^{N} \frac{r_{ik}}{N}$$

where r_{ik} is the outcome of play $i \leq N$ on bandit k.

Baseline Policy Example

Suppose there are four bandits with success probabilities

$$\mathrm{p} = [0.35, 0.40, 0.30, 0.25]$$

We sample from each bandit 30 times and the estimated reward probabilities are

$$\hat{\mathrm{p}} = [0.4, 0.37, 0.2, 0.23]$$

We would then exploit the first bandit since it has highest probability based on this run. This is sub-optimal, why? What can we do to fix it?

	Bandit 1	Bandit 2	Bandit 3	Bandit 4
Play 1	0	0	0	0
Play 2	1	1	0	0
Play 3	0	0	0	0
Play 4	0	0	0	0
Play 5	0	0	0	0
...	...	...	...	...
Play 26	0	0	0	0
Play 27	0	1	0	1
Play 28	1	0	0	0
Play 29	0	0	0	1
Play 30	0	0	1	0

Epsilon Greedy Policy

What if we could interleave our exploration with our exploitation?

The ϵ-greedy strategy is defined by randomly picking an action type with probability ϵ:

- exploit: choose the bandit with current highest probability of return
- explore: uniformly randomly select a bandit

Video Demonstration

ϵ-greedy risks getting stuck in a sub-optimal strategy if ϵ is not high enough, so we must carefully select this hyper parameter.

Epsilon Greedy Policy

```
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)
est_probs = np.zeros(n_bandits)

# Try adjusting this!
eps = 0.1

for i in range(n_turns):
    # Flip uneven coin to decide action
    if np.random.random() < eps:
        # Explore
        chosen_idx = np.random.choice(np.arange(n_bandits))
    else:
        # Expoit
        chosen_idx = np.argmax(est_probs)

    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

print(est_probs)
print(play_counts)
```

```
[0.44 0.5  0.   0.5 ]
[34. 60.  2.  4.]
```

Regret

Regret

The **regret** measures the quality of a strategy by comparing its expected return against the optimal strategy (if we know the parameters)

$$\text{regret} = T\theta^* - \sum_{t=1}^{T} r_t$$

where T is the number of plays, r_t is the reward for play t, and θ^* is the best reward mean i.e. $\theta^* = \max_k\{\theta_k\}$.

In our case of binary rewards (Bernoulli trials) the regret is

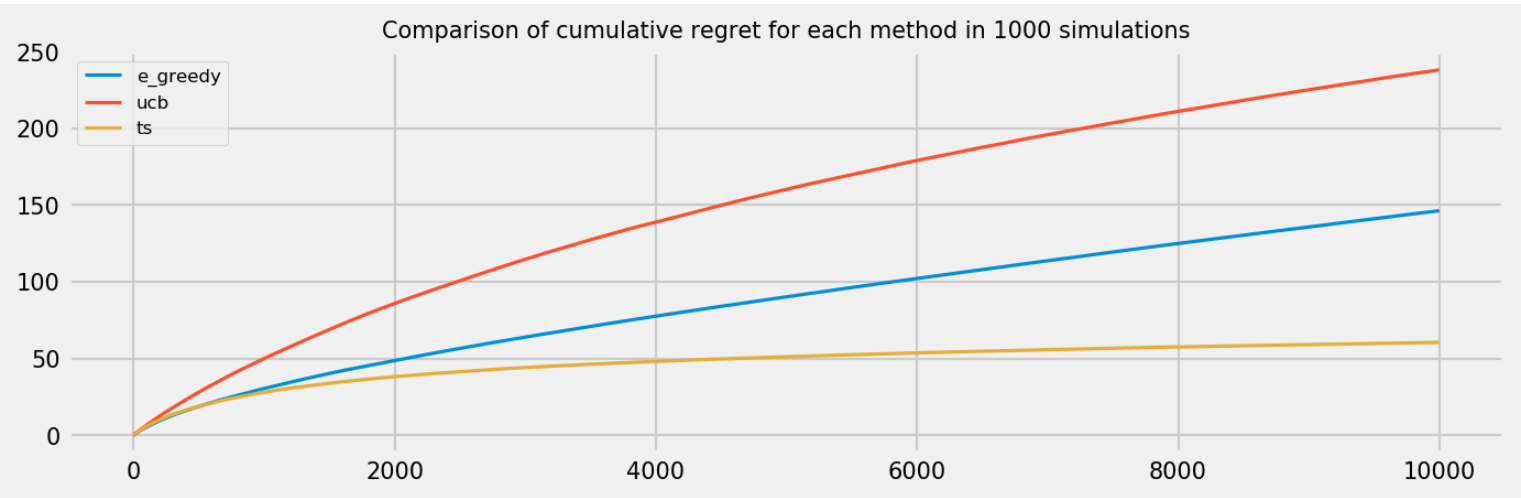
$$\text{regret} = Tp^* - \sum_{t=1}^{T} r_t$$

since the mean of a Bernoulli random variable is just p.

Regret

Regret is a function of time. When we compare strategies we should then plot the regret over time, rather than at a fixed point.

We are mostly interested in the long run performance, since it is common for strategies to have high regret early because they are doing more exploration.



Regret - Example

In our earlier example we estimated (incorrectly) that the first bandit had the highest success probability.

Suppose we then play 250 more times on this first bandit only and we get 86 wins. The second bandit has the highest probability of success i.e. $p_2 = p^* = 0.4$.

The regret is

$$\begin{aligned} \text{regret} &= Tp^* - \sum_{t=1}^{T} r_t \\ &= 250 \times 0.4 - 86 \\ &= 14 \end{aligned}$$

In other words, we would expect to have 14 more wins if we played the second bandit.

Wrapping Up

Next Week

- dynamic policies
- using context (i.e. user information)
- offline evaluation of policies