Bandits

David S. Rosenberg

NYU: CDS

October 6, 2021

Contents

Motivation

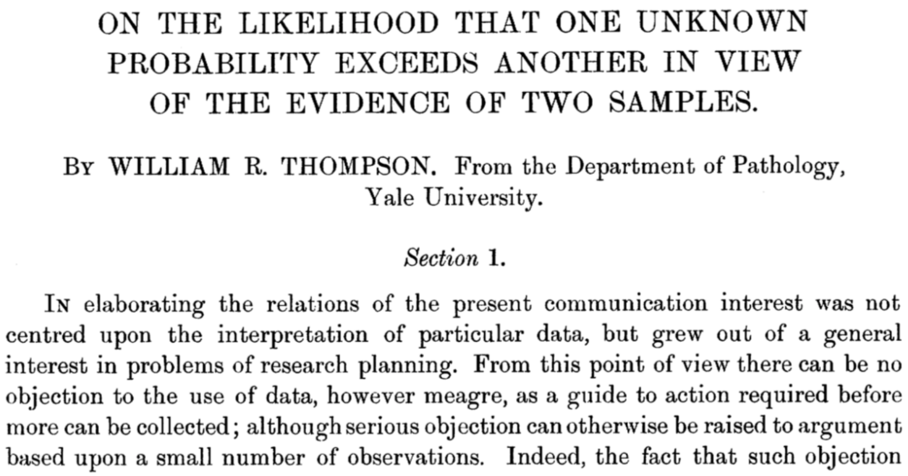
William Thompson (Yale, Dept. of Pathology, 1933)

- Thompson was thinking about comparing two medical treatments.
- Standard practice:
    - Run a randomized controlled trial (RCT).
    - If one treatment is significantly better than the other,
        - use the better one, going forward.
    - Otherwise,
        - back to the lab.

William Thompson (Yale, Dept. of Pathology, 1933)

- The more data we have, the more certainty we have in RCT conclusions.
- But what if we only have a modest amount of data?
- Suppose data is suggestive that one treatment is better,
  - but not yet "conclusive" (as defined by $p < 0.05$, or whatever).
- Should we adjust our actions in accordance with this information?

Thompon's 1933 paper



ON THE LIKELIHOOD THAT ONE UNKNOWN PROBABILITY EXCEEDS ANOTHER IN VIEW OF THE EVIDENCE OF TWO SAMPLES.

By WILLIAM R. THOMPSON. From the Department of Pathology, Yale University.

Section 1.

In elaborating the relations of the present communication interest was not centred upon the interpretation of particular data, but grew out of a general interest in problems of research planning. From this point of view there can be no objection to the use of data, however meagre, as a guide to action required before more can be collected; although serious objection can otherwise be raised to argument based upon a small number of observations. Indeed, the fact that such objection

From [Tho33]: Basically he's saying: use whatever data you have as a guide to action, and adjust appropriately as you get new data.

William Thompson

- Thompson says, that if

  > *"P is the probability estimate that one treatment . . . is better than a second, as judged by data at present available, then we might take some monotone increasing function of P, say f(P), to fix the fraction of such individual to be treated in the first manner; until more evidence may be utilised . . . the remaining fraction . . . to be treated in the second manner."*

- For example, we could take $f(P) = P$.
- Then if we're 90% sure that 1 is better than 2, then
  - 90% of people get treatment 1 and
  - 10% of people get treatment 2.
- As we get more data, these percentages will adjust accordingly.
- This is the essence of "**Thompson sampling**" as we know it today.

Note that Thompson is taking a Bayesian approach from the first sentence: having a probability P that one treatment is better than a second is something that we have in a Bayesian approach, but not in the classical frequentist approach. With modern terminology, we would say that P is the posterior probability that treatment 1 is better than treatment 2 (or the prior probability, if no data has yet been observed).

Exploration / exploitation tradeoff

- We can choose the action that seems best
    - **according to the data we already have.**
- We can choose the action that is optimal for
    - **improving our knowledge about the value of different actions**.
- The **explore/exploit** tradeoff refers to the tradeoff between these extremes.
- The bandit setting is a natural setting for exploring this problem.

Online vs. Offline learning

Offline supervised learning

- Given training data $\mathcal{D}$: $(X_1, Y_1), \ldots, (X_n, Y_n)$
- Find $f \in \{f_\theta(x) : \theta \in \Theta\}$ with small average loss on $\mathcal{D}$.
- Then deploy f.
- Once f is deployed,
    - we observe new X's — the inputs that f makes predictions for.
- **Sometimes**, we'll get corresponding Y's soon after prediction $f(X)$ is made
    - e.g. the rating a user gives to a movie
    - e.g. whether or not a user clicks a link
    - e.g. what price a stock went to
- These new labeled data are stored for future training.
- But can we act on them more immediately?

Online supervised learning

Online supervised learning

Choose an initial $\theta = \theta_0 \in \Theta$.

Each round of online learning comprises the following steps:

1. Observe X.
2. Take action or make prediction $f_\theta(X)$.
3. Observe Y.
4. Update θ to account for new observation (X, Y)

Repeat indefinitely...

- In **online** learning, f_θ is updated after every round.
- In **offline** learning, f_θ isn't updated once deployed.

Online vs. offline

- Online not much different from offline.
- Functionally, we could view online learning as offline learning where
  - we deploy a new model after every round;
  - new model trained on all data observed up to that point.
- In practice, we probably can't wait for a full retraining after every round.
- In practice, an **online learning algorithm** is usually one that can update a model with a new round (or minibatch) of training data (X, Y), without looking at the previous training data.

SGD is online learning!

- SGD is an online learning algorithm.
- From this perspective, we no longer have epochs.
- Whenever we get a new data point or mini-batch, take an update step.
- In practice, periodically reinitialize your model with a full batch retraining.

- You might be tempted to think that online learning is more appropriate when the world is changing over time, and we want our model to adapt.
- This isn't really the case. Online algorithms generally assume the world is stationary. For example, SGD usually has a decaying step size because we want to "remember" what we've learned before. If we want to forget what we learned from old training data, we shouldn't decay the step size after a certain point.
- If you think the world is changing over time, it's better to explicitly bring your assumptions into the problem. For example, if you think the last 2 weeks of data are much more relevant than anything preceding it, you might try training on a rolling window of 2 weeks, or exponentially weighting training data so that most of the weight is given to data from the last two weeks.

The bandit setting (informal)

Observing the label y is very helpful!

Key feature of supervised learning

Once we observe y, it's usually straightforward to figure out what the optimal action would have been:

$$a^* = \arg\min_a \ell(a, y).$$

Example: Multiclass Classification and Regression

Once we observe the correct class label (multiclass) or the true response (regression), we know the optimal action would have been to produce the correct label or the true response as an action.

What if the outcome y is never revealed?

- Consider k-class classification with 0/1 loss.
- We get an x and we predict class label 3.
- We receive a loss of 1 $\implies$ we were wrong.
- The correct class label y is **not** revealed.
- What to do next time we receive the same or a similar x?
- Try a different label and hope for the best...
- This setting forces us to do trial & error to figure out the optimal action.
    - Makes it much harder to learn!

Example: recommendations with full rating feedback

- There are 5 hit movies and we want to recommend one to a user
- We get a feature vector x that describes the user.
- We choose a movie and recommend it (that's our action).
- In online learning, we would then get feedback on
    - which of the 5 movies the individual would actually have liked best
    - that would be the "label"
- This is known as **full feedback** since it's enough information to determine
    - what would have been the best movie to suggest (i.e. best action).
- If another individual with the same or similar x shows up, we would know what to do.

Example: recommendations with partial rating feedback

- A [slightly] more realistic scenario:
- User watches the movie we recommend and gives a rating
  - Suppose 4 out of 5 stars
- 4 out of 5 is decent... (or maybe that depends?)
- No feedback on the other 4 movies.
- If we get a similar user $x' \approx x$, should we recommend the same movie?
- Or should we try a different movie to see if we can get to 5 out of 5?
- This is another exploration / exploitation problem.

The bandit problem

A bandit problem proceeds in rounds of the following steps:

1. [Optional] Observe input/**context** x.
2. Take action a.
3. Receive loss $\ell \in \mathbb{R}$.

- Note that there is no mention of a label y.

Types of bandit problems

- **Multiarmed bandit**: when the set of possible actions is finite
- **Contextual bandit**: when a bandit problem has a context x in each round
    - Context x can help determine the best action to take
    - The context is analogous to input features supervised learning
- **Losses vs. Rewards:** In bandits (and reinforcement learning), we often speak in terms of receiving rewards $r \in \mathbb{R}$ rather than losses $\ell \in \mathbb{R}$.
- Depending on whether we have rewards or losses, we either try to
    - maximize the total rewards received, or
    - minimize the total losses received.

Bandit setting (formalized)

Stochastic k-armed bandit – reward vectors

- Environment randomly draws a rewards vector for each round.
- Here we stack the reward vectors into a matrix (each row is a reward vector) :

Round	$R(1)$	$R(2)$	$R(3)$	$\cdots$	$R(k)$
1	1	6	3	$\cdots$	2
2	2	4	1	$\cdots$	1
3	0	2	3	$\cdots$	8
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
T	2	4	1	$\cdots$	2

- Column $R(a)$ in row t gives the reward for action a in round t.
- Player never observes this matrix in its entirety.

Stochastic k-armed bandit – play

- In each round, player selects action $a \in \{1, \ldots, k\}$.
- After 2 rounds, player will have observed two entries of this matrix, e.g.:

Round	$R(1)$	$R(2)$	$R(3)$	$\cdots$	$R(k)$
1		6		$\cdots$	
2				$\cdots$	1
3				$\cdots$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
T				$\cdots$	

- Next up: choose action for round 3.
- Objective: choose actions to maximize the total rewards received

Stochastic k-armed bandit (formal)

Stochastic k-armed bandit

1. Environment samples **reward vectors** for all rounds:

$$R_1, \ldots, R_T \in \mathbb{R}^k \text{ i.i.d. } P,$$

   where $R_t = (R_t(1), \ldots, R_t(k)) \in \mathbb{R}^k$.

2. For $t = 1, \ldots, T$,
   1. Our algorithm **selects action**/arm $A_t \in \{1, \ldots, k\}$ based on **history**

$$\mathcal{D}_t = \Big((A_1, R_1(A_1)), \ldots, (A_{t-1}, R_{t-1}(A_{t-1})) \Big).$$

   2. Our algorithm **receives reward** $R_t(A_t)$.

- We **never observe** $R_t(a)$ for $a \neq A_t$.

- It would be equivalent to say that at the beginning of every round, the environment generates a reward vector $R_t \in \mathbb{R}^k$ from P. However, we want to emphasize that the reward vectors $R_1, \ldots, R_T$ are i.i.d. and independent of the actions selected for all time.
- In fact, the assumption is generally that the reward vector distribution is a product distribution: $P = P_1 \times P_2 \times \cdots \times P_k$. In other words, the rewards are independent, both across actions within a particular round and across rounds. But we won't need this.
- We use a capital letter A_t for action because we're thinking of A_t as random variables, since 1) we allow algorithms to use randomness in action selection, but more fundamentally, 2) they should depend on the history of rewards received, which are assumed to be random.
- However, the "stochastic" in stochastic k-armed bandit refers to the randomness of the reward vectors.

A bandit algorithm

- The **history** at round t is the sequence of observations up to the start of round t:

$$\mathcal{D}_t = ((A_1, R_1(A_1)), \ldots, (A_{t-1}, R_{t-1}(A_{t-1})))$$

- In each round t, a **bandit algorithm** chooses an action A_t
  - based **only on** the history $\mathcal{D}_t$ at round t and
  - a random number (for randomized algorithms)

The connection with missing data and RCTs

- If $k = 2$ we can write the observation sequence at round 6 as

Round t	A	$R(1)$	$R(2)$
1	1	6.1	?
2	2	?	1.2
3	1	0.9	?
4	2	?	3.0
5	2	?	1.9

- The "full-data" for a bandit at the beginning of round t would be

$$(A_1, R_1(1), \ldots, R_1(k)), \ldots, (A_{t-1}, R_{t-1}(1), \ldots R_{t-1}(k)).$$

The connection with missing data and RCTs

Round t	A	$R(1)$	$R(2)$
1	1	6.1	?
2	2	?	1.2
3	1	0.9	?
4	2	?	3.0
5	2	?	1.9

- In causal inference setting, we want to estimate $\mathbb{E}[R(1) - R(2)]$ from this data.
- With bandits, we want to **choose actions** A_t that maximize rewards: $\sum_t R_t(A_t)$.
- The problems are clearly related... but how are they different?

Bandits vs. treatment effect estimation

- In treatment effect estimation, we want good estimates of $\mathbb{E}R(1)$ and $\mathbb{E}R(2)$.
    - (Or $\mathbb{E}Y(1)$ and $\mathbb{E}Y(0)$ in the notation of that module.)
- Suppose $\mathbb{E}R(1) \ll \mathbb{E}R(2)$.
- In bandits, we don't really care about getting a precise estimate of $\mathbb{E}R(1)$.
- Once we're quite sure that $\mathbb{E}R(2) > \mathbb{E}R(1)$, we don't need
    - a precise estimate of $\mathbb{E}R(1)$, or
    - a precise estimate of $\mathbb{E}R(2)$ for that matter.
- Knowing $\mathbb{E}R(2) > \mathbb{E}R(1)$ is enough to determine optimal action choice.
- For bandits: we should only choose actions that might give the best expected rewards.

Bandit strategy: ε-greedy

Strategies for stochastic bandits

- Let's try to optimize the expected cumulative reward:

$$\mathbb{E}\left[\sum_{i=1}^{T} R_i(A_i)\right].$$

- Need a strategy for selecting action A_t at round t,
    - based on the history $(A_1, R_1(A_1)), \ldots, (A_{t-1}, R_{t-1}(A_{t-1}))$.
- In any round t, the expected reward for playing action a is $\mathbb{E}R_t(a) = \mathbb{E}R(a)$.
- So the optimal action is the action that has the largest expected reward.

Expected rewards and estimate

- Let's write $q_*(1) = \mathbb{E}R(1), \ldots, q_*(k) = \mathbb{E}R(k)$.
- Let $\hat{q}_t(1), \ldots, \hat{q}_t(k)$ be the natural estimators of these parameters at the **beginning of round** t:

$$\hat{q}_t(a) = \frac{\sum_{i=1}^{t-1} \mathbb{1}\left[A_i = a\right] R_i(a)}{\sum_{i=1}^{t-1} \mathbb{1}\left[A_i = a\right]} \qquad \text{for } a = 1, \ldots, k.$$

- If the denominator is 0, we'll set $\hat{q}_t(a)$ to 0, or to whatever our best guess is for $q_*(a)$.

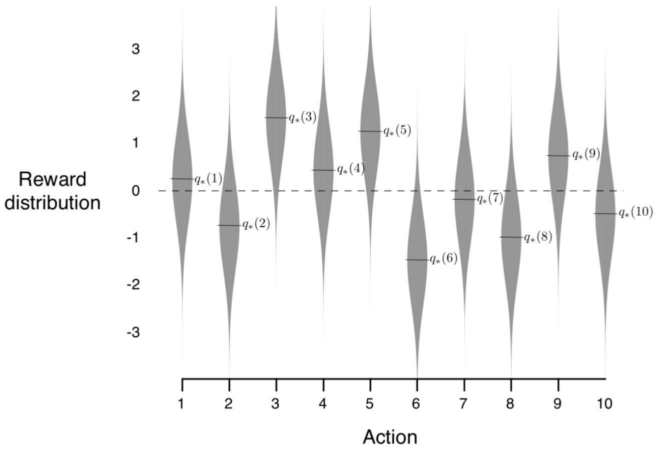
The ε-greedy algorithm

- Define the **greedy** action at round t to be the action with the largest mean estimate:

$$a_t^{\text{greedy}} = \arg\max_a \hat{q}_t(a).$$

- We'll assume that we break ties in the argmax at random.
- In the ε-greedy algorithm, in round t
    - With probability ε we **explore**: take $A_t \sim \text{Uniform}(1, \dots, k)$
    - With probability $1 - \varepsilon$ we **exploit**: take $A_t = a_t^{\text{greedy}}$.
- We're "exploiting" our knowledge with probability $1 - \varepsilon$.
- We're "exploring" to gain better estimates of the arm rewards with probability ε.

The 10-armed bandit from Sutton and Barto[1]



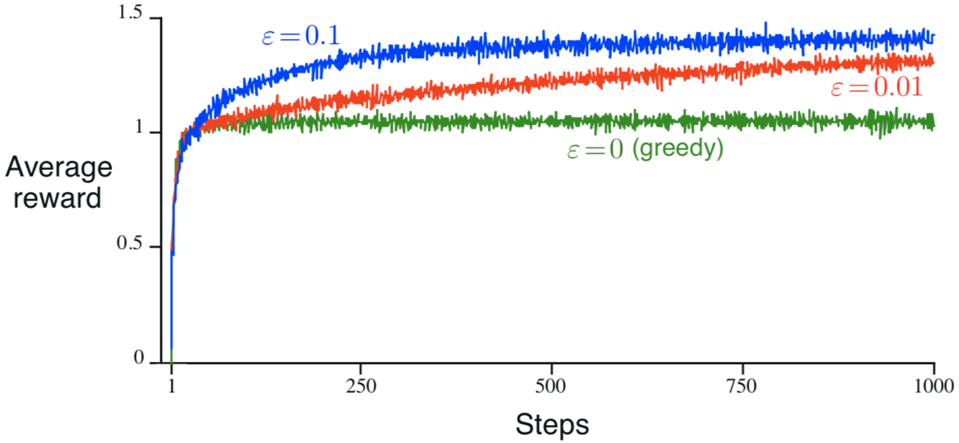
[1]Fig 2.1 in [SB18, Sec 2.3].

This is a single instance of the "10-armed testbed" from [SB18, Sec 2.3]. Each instance is a 10-armed bandit with reward distributions chosen randomly as follows:

- Let $q_*(a) = \mathbb{E}[R_t \mid A_t = a]$, for $a = 1, \ldots, 10$, denote the mean of the reward distribution for action a, for each round t.
- We choose each $q_*(a)$ i.i.d. from $\mathcal{N}(0, 1)$, for each a.
- Then the reward distribution for action a is given by $\mathcal{N}(q_*(a), 1)$.

We repeat this process 2000 times to generate 2000 random 10-armed bandits.

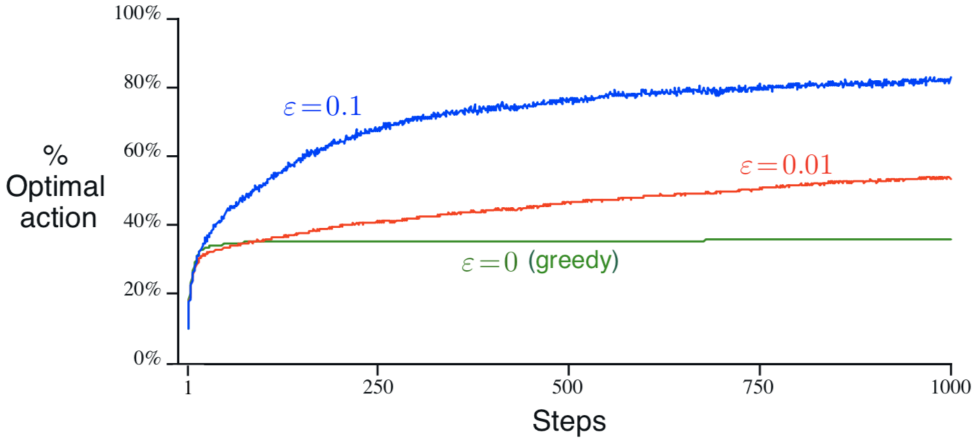
Performance of ε- greedy on 10-armed testbed[2]



Average reward at round t is $\frac{1}{t}\sum_{i=1}^{t} R_t(A_t)$.

[2]Fig 2.2 in [SB18, Sec 2.3].

Performance of ε- greedy on 10-armed testbed[3]



[3]Fig 2.2 in [SB18, Sec 2.3].

- Sutton and Barto ran the ε-greedy algorithms with 3 settings of epsilon on the 2000 bandit problems, each for 1000 steps. The average reward achieved at step t across the 2000 bandit problems is plotted over time. Note that ε-greedy with $\varepsilon = 0.1$ will never choose the optimal action more than about 90% of the time. Or $.9 + \varepsilon/k = .91$ of the time, to be more precise. The pure greedy strategy ($\varepsilon = 0$) gets stuck for lack of exploration.
- As the number of steps goes to infinity, we will eventually have perfect estimates of the expected reward for each action, and so we will choose the optimal action $1 - \varepsilon + \varepsilon/k$ fraction of the time.

References

Resources

- There's a chapter on bandits in Sutton and Barto's book on reinforcement learning [SB18, Sec 2.3], which is worth reading.
- *Bandit Algorithms* by Lattimore and Szepesvári is a relatively new book that is much more theoretical, but a great book if you're into that sort of thing [LS20].

References I

[LS20] Tor Lattimore and Csaba Szepesvári, *Bandit algorithms*, Cambridge University Press, 2020.

[SB18] Richard S. Sutton and Andrew G. Barto, *Reinforcement learning: An introduction*, A Bradford Book, Cambridge, MA, USA, 2018.

[Tho33] William R. Thompson, *On the likelihood that one unknown probability exceeds another in view of the evidence of two samples*, Biometrika **25** (1933), no. 3/4, 285.