Learning through Experimentation

CS246: Mining Massive Datasets
Jure Leskovec, Stanford University
http://cs246.stanford.edu
Web advertising
- We discussed how to match advertisers to queries in real-time
- But we did not discuss how to estimate the CTR (Click-Through Rate)

Recommendation engines
- We discussed how to build recommender systems
- But we did not discuss the cold-start problem
Learning through Experimentation

- What do **CTR** and cold-start have in common?
- With every ad we show/product we recommend we gather more data about the ad/product
- Theme: Learning through experimentation
Example: Web Advertising

- **Google’s goal:** Maximize revenue
- **The old way:** Pay by impression (CPM)
  - Best strategy: Go with the highest bidder
    - But this ignores the “effectiveness” of an ad
- **The new way:** Pay per click! (CPC)
  - Best strategy: Go with expected revenue
  - What’s the expected revenue of ad $a$ for query $q$?
  - $E[\text{revenue}_{a,q}] = P(\text{click}_a \mid q) \times \text{amount}_{a,q}$

Prob. user will click on ad $a$ given that she issues query $q$
(Uunknown! Need to gather information)

Bid amount for ad $a$ on query $q$
(Known)
Clinical trials:
- Investigate effects of different treatments while minimizing adverse effects on patients

Adaptive routing:
- Minimize delay in the network by investigating different routes

Asset pricing:
- Figure out product prices while trying to make most money
Approach: Bandits
Approach: Multiarmed Bandits
Each arm $\alpha$

- **Wins** (reward=1) with fixed (unknown) prob. $\mu_\alpha$
- **Loses** (reward=0) with fixed (unknown) prob. $1-\mu_\alpha$
- All draws are independent given $\mu_1 \ldots \mu_k$
- **How to pull arms to maximize total reward?**
k-Armed Bandit

- How does this map to our setting?
- Each query is a bandit
- Each ad is an arm
- We want to estimate $\mu_a$, the arm’s probability of winning (i.e., ad’s CTR $\mu_a$)
- Every time we pull an arm we do an ‘experiment’
Stochastic k-Armed Bandit

The setting:
- Set of $k$ choices (arms)
- Each choice $a$ is associated with unknown probability distribution $P_a$ supported in $[0, 1]$
- We play the game for $T$ rounds
- In each round $t$:
  1. We pick some arm $a$
  2. We obtain random sample $X_t$ from $P_a$
     - Note reward is independent of previous draws
- Our goal is to maximize $\sum_{t=1}^{T} X_t$
- Problem: we don’t know $\mu_a$! But every time we pull some arm $a$ we get to learn a bit about $\mu_a$
Online Optimization

- Online optimization with limited feedback

<table>
<thead>
<tr>
<th>Choices</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>$X_5$</th>
<th>$X_6$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$a_2$</td>
<td>0</td>
<td></td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a_k$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Like in online algorithms:
  - Have to make a choice each time
  - But we only receive information about the chosen action
Policy: a strategy/rule that tells me which arm to pull in each iteration
- Hopefully policy depends on the history of rewards

How to quantify performance of the algorithm? Regret!
Performance Metric: Regret

- Let $\mu_a$ be the mean reward of $P_a$
- Payoff/reward of best arm: $\mu^* = \max_a \mu_a$
- Let $i_1, i_2 ... i_T$ be the sequence of arms pulled
- Instantaneous regret at time $t$: $r_t = \mu^* - \mu_{i_t}$
- Total regret:

\[
R_T = \sum_{t=1}^{T} r_t
\]

- Typical goal: Want a policy (arm allocation strategy) that guarantees: $\frac{R_T}{T} \to 0$ as $T \to \infty$
  - Note: Ensuring $R_T/T \to 0$ is stronger than maximizing payoffs (minimizing regret), as it means that in the limit we discover the true best hand.
Allocation Strategies

- If we knew the payoffs, which arm would we pull?

\[
\text{Pick } \text{arg max}_a \mu_a
\]

- What if we only care about estimating payoffs \( \mu_a \)?
  - Pick each of \( k \) arms equally often: \( \frac{T}{k} \)
  - Estimate: \( \hat{\mu}_a = \frac{k}{T} \sum_{j=1}^{T/k} X_{a,j} \)
  - Regret: \( R_T = \frac{T}{k} \sum_{a=1}^{k} \left( \mu^* - \hat{\mu}_a \right) \)

\( X_{a,j} \ldots \) payoff received when pulling arm \( a \) for \( j \)-th time
Bandit Algorithm: First try

- Regret is defined in terms of average reward
- So, if we can estimate avg. reward we can minimize regret
- Consider algorithm: *Greedy*
  Take the action with the highest avg. reward
  - **Example:** Consider 2 actions
    - **A1** reward 1 with prob. 0.3
    - **A2** has reward 1 with prob. 0.7
  - Play **A1**, get reward 1
  - Play **A2**, get reward 0
  - Now avg. reward of **A1** will never drop to 0, and we will never play action **A2**
Exploration vs. Exploitation

- The example illustrates a classic problem in decision making:
  - We need to trade off between **exploration** (gathering data about arm payoffs) and **exploitation** (making decisions based on data already gathered)

- The Greedy algo does not explore sufficiently
  - **Exploration**: Pull an arm we never pulled before
  - **Exploitation**: Pull an arm $a$ for which we currently have the highest estimate of $\mu_a$
The problem with our Greedy algorithm is that it is too certain in the estimate of $\mu_a$

- When we have seen a single reward of 0 we shouldn’t conclude the average reward is 0

Greedy can converge to a suboptimal solution!
Algorithm: Epsilon-Greedy

For $t=1:T$
- Set $\varepsilon_t = O\left(\frac{1}{t}\right)$ (that is, $\varepsilon_t$ decays over time $t$ as $1/t$)
- With prob. $\varepsilon_t$: Explore by picking an arm chosen uniformly at random
- With prob. $1 - \varepsilon_t$: Exploit by picking an arm with highest empirical mean payoff

Theorem [Auer et al. ‘02]
For suitable choice of $\varepsilon_t$ it holds that

$$R_T = O(k \log T) \Rightarrow \frac{R_T}{T} = O\left(\frac{k \log T}{T}\right) \to 0$$

$k$...number of arms
What are some issues with Epsilon-Greedy?

- “Not elegant”: Algorithm explicitly distinguishes between exploration and exploitation

- More importantly: Exploration makes suboptimal choices (since it picks any arm equally likely)

Idea: When exploring/exploiting we need to compare arms
Suppose we have done experiments:

- Arm 1: 1 0 0 1 1 0 0 1 0 1
- Arm 2: 1
- Arm 3: 1 1 0 1 1 1 0 1 1 1

Mean arm values:

- Arm 1: 5/10, Arm 2: 1, Arm 3: 8/10

Which arm would you pick next?

Idea: Don’t just look at the mean (that is, expected payoff) but also the confidence!
A confidence interval is a range of values within which we are sure the mean lies with a certain probability.

- We could believe $\mu_a$ is within [0.2, 0.5] with probability 0.95.
- If we would have tried an action less often, our estimated reward is less accurate so the confidence interval is larger.
- Interval shrinks as we get more information (try the action more often).
Confidence Intervals (2)

- Assuming we know the confidence intervals
- Then, instead of trying the action with the highest mean we can try the action with the highest upper bound on its confidence interval
- This is called an optimistic policy
  - We believe an action is as good as possible given the available evidence
Confidence Based Selection

After more exploration
Suppose we fix arm $a$:

- Let $X_{a,1} \ldots X_{a,m}$ be the payoffs of arm $a$ in the first $m$ trials
  - So, $X_{a,1} \ldots X_{a,m}$ are i.i.d. rnd. vars. taking values in $[0,1]$
- Mean payoff of arm $a$: $\mu_a = E[X_{a,\cdot}]$
- **Our estimate:** $\hat{\mu}_{a,m} = \frac{1}{m} \sum_{\ell=1}^{m} X_{a,\ell}$
- Want to find $b$ such that with high probability $|\mu_a - \hat{\mu}_{a,m}| \leq b$
  - Want $b$ to be as small as possible (so our estimate is close)

**Goal:** Want to bound $P(|\mu_a - \hat{\mu}_{a,m}| \leq b)$
Hoeffding’s Inequality (1)

**Hoeffding’s inequality** provides an upper bound on the probability that the average deviates from its expected value by more than a certain amount:

- Let $X_1 \ldots X_m$ be i.i.d. rnd. vars. taking values in $[0,1]$
- Let $\mu = E[X]$ and $\hat{\mu}_m = \frac{1}{m} \sum_{\ell=1}^{m} X_\ell$
- Then: $P(|\mu - \hat{\mu}_m| \geq b) \leq 2 \exp(-2b^2m) = \delta$
  - $\delta$... is the confidence level

- To find out the confidence interval $b$ (for a given confidence level $\delta$) we solve:
  - $2e^{-2b^2m} \leq \delta$ then $-2b^2m \leq \ln(\delta/2)$
  - So: $b \geq \sqrt{\frac{\ln\left(\frac{2}{\delta}\right)}{2m}}$
Hoeffding’s Inequality (2)

- \( P(|\mu - \hat{\mu}_m| \geq b) \leq 2 \exp(-2b^2m) \)
  where \( b \) is our upper bound, \( m \) is number of times we played the action

- Let’s set \( b = b(a, T) = \sqrt{2\log(T)/m_a} \)

- Then: \( P(|\mu - \hat{\mu}_m| \geq b) \leq 2T^{-4} \) which converges to zero very quickly:

  Notice:
  - If we don’t play action \( a \), its upper bound \( b \) increases
    - This means we never permanently rule out an action no matter how poorly it performs
  - Prob. our upper bound is wrong decreases with time \( T \)
**UCB1 Algorithm**

- **UCB1 (Upper confidence sampling) algorithm**
  - Set: $\hat{\mu}_1 = \cdots = \hat{\mu}_k = 0$ and $m_1 = \cdots = m_k = 0$
    - $\hat{\mu}_a$ is our estimate of payoff of arm $a$
    - $m_a$ is the number of pulls of arm $a$ so far
  - For $t = 1:T$
    - For each arm $a$ calculate: $UCB(a) = \hat{\mu}_a + \alpha \sqrt{\frac{2 \ln t}{m_a}}$
    - Pick arm $j = \arg \max_a UCB(a)$
    - Pull arm $j$ and observe $y_t$
    - Set: $m_j \leftarrow m_j + 1$ and $\hat{\mu}_j \leftarrow \frac{1}{m_j} (y_t + (m_j - 1) \hat{\mu}_j)$

$\alpha$...is a free parameter trading off exploration vs. exploitation

[Auer et al. ’02](http://www.jmlr.org/papers/volume3/auer02a/auer02a.pdf)
UCB1: Discussion

- \( \text{UCB}(\alpha) = \hat{\mu}_a + \alpha \sqrt{\frac{2 \ln t}{m_a}} \)

  - Confidence interval **grows** with the total number of actions \( t \) we have taken
  - But **shrinks** with the number of times \( m_a \) we have tried arm \( a \)
  - This ensures each arm is tried infinitely often but still balances exploration and exploitation
  - \( \alpha \) plays the role of \( \delta \): \( \alpha = f \left( \frac{2}{\delta} \right) \)

\[ b \geq \frac{\ln \left( \frac{2}{\delta} \right)}{\sqrt{2m}} \]

"Optimism in face of uncertainty": The algorithm believes that it can obtain extra rewards by reaching the unexplored parts of the state space
Theorem [Auer et al. 2002]

Suppose optimal mean payoff is $\mu^* = \max_a \mu_a$

And for each arm let $\Delta_a = \mu^* - \mu_a$

Then it holds that

$$E[R_T] \leq \left[ 8 \sum_{a: \mu_a < \mu^*} \frac{\ln T}{\Delta_a} \right] + \left( 1 + \frac{\pi^2}{3} \right) \left( \sum_{i=a}^k \Delta_a \right)$$

$O(k \ln T)$

$O(k)$

So: $O \left( \frac{R_T}{T} \right) \leq k \frac{\ln T}{T}$

(note this is worst case regret)
Summary so far

- \( k \)-armed bandit problem as a formalization of the exploration-exploitation tradeoff

- Analog of online optimization (e.g., SGD, BALANCE), but with **limited feedback**

- Simple algorithms are able to achieve no regret (in the limit)
  - Epsilon-greedy
  - UCB (Upper Confidence Sampling)
Example

- 10 actions, 1M rounds, uniform [0,1] rewards

Theoretical worse-case cumulative regret

Real cumulative regret
Use-case: Pinterest

- **Problem:** For new pins/ads we do not have enough signal on how good they are
  - How likely are people to interact with them?
- **Idea:**
  - Try to maximize the rewards from several unknown slot machines by deciding which machines and the order to play
  - Each pin is regarded as an arm, user engagement are considered as rewards
  - Making tradeoff between exploration and exploitation, avoid keep showing the best known pins and trapping the system into local optima
Solution: Bandit algorithm in round $t$

1. Algorithm observes user is seeing a set $A$ of pins/ads
2. Based on payoffs from previous trials, algorithm chooses arm $a \in A$ and receives payoff $r_{t,a}$
   - Note only feedback for the chosen $a$ is observed
3. Algorithm improves arm selection strategy with each observation $(a, r_{t,a})$

If the score for a pin is low, filter it out
Use Case: A/B testing

- A/B testing is a controlled experiment with two variants, A and B
- Part of the traffic sees variant A, part variant B

Click rate: 52%  72%
Use Case: A/B testing

- Part of the traffic sees variant A, part variant B
- Hypothesis test: does variant A outperform variant B? What test to perform?

<table>
<thead>
<tr>
<th>Assumed Distribution</th>
<th>Example</th>
<th>Standard Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>Average Revenue Per Paying User</td>
<td>Welch's t-test (Unpaired t-test)</td>
</tr>
<tr>
<td>Binomial</td>
<td>Click Through Rate</td>
<td>Fisher's exact test</td>
</tr>
<tr>
<td>Poisson</td>
<td>Transactions Per Paying User</td>
<td>E-test</td>
</tr>
<tr>
<td>Multinomial</td>
<td>Number of each product purchased</td>
<td>Chi-squared test</td>
</tr>
</tbody>
</table>

- If A outperforms B, we want to stop the experiment as soon as possible
Imagine you have two versions of the website and you’d like to test which one is better

- Version A has engagement rate of 5%
- Version B has engagement rate of 4%

You want to establish with 95% confidence that version A is better

- Using t-test, you’d need 22,330 observations (11,165 in each arm) to establish that

Can bandits do better?
Example: Bandits vs. A/B testing

- **How long does it take to discover A > B?**
  - **A/B test:** We need 22,330 observations. Assuming 100 observations/day, we need 223 days

- **The goal is to find the best action (A vs. B)**
  - The randomization distribution (traffic to A vs. B) can be updated as the experiment progresses

- **Idea:**
  - Twice per day, examine how each of the variations/arms has performed
  - Adjust the fraction of traffic that each arm will receive going forward
  - An arm that appears to be doing well gets more traffic, and an arm that is clearly underperforming gets less
Thompson Sampling

- **Thompson sampling** assigns sessions to arms in proportion to the probability that each arm is optimal.

- Assume outcome distribution in the set \{0,1\}
  - The arm either converts or not

- Then we flip a coin with probability \(\theta\) \(\rightarrow\) Bernoulli distribution!

- To estimate \(\theta\), we count up numbers of ones and zeros.
Given observed ones and zeroes, how do we calculate the distribution of possible values of $\theta$?

Let:

- $\theta = (\theta_1, \theta_2, ..., \theta_k)$ ... the vector of conversion rates for arms 1, ..., $k$.
  - $\theta_i = \text{#successes} / (\text{#successes} + \text{#failures})$
Beta-Bernoulli Case

- Beta(\(\alpha, \beta\)) → Given a 0’s and b 1’s, what is the distribution over means?
- Prior → pseudocounts
- Likelihood → observed counts
- Posterior → psuedocounts + observed counts
Arm probabilities $\theta$ can be computed using sampling:

- Each element of $\theta$ is an independent random variable from a Beta distribution ($\alpha + \text{successes}, \beta + \text{failures}$)

---

Algorithm 2 Thompson sampling for the Bernoulli bandit

**Require:** $\alpha, \beta$ prior parameters of a Beta distribution

$S_i = 0, F_i = 0$, $\forall i$. {Success and failure counters}

for $t = 1, \ldots, T$ do

for $i = 1, \ldots, K$ do

   Draw $\theta_i$ according to Beta($S_i + \alpha, F_i + \beta$).

end for

Draw arm $\hat{i} = \arg \max_i \theta_i$ and observe reward $r$

if $r = 1$ then

   $S_{\hat{i}} = S_{\hat{i}} + 1$

else

   $F_{\hat{i}} = F_{\hat{i}} + 1$

end if

end for
Thompson Sampling in General

Thompson Sampling:
- 1. Specify prior (in Beta case often Beta(1,1))
- 2. Sample from each posterior distribution to get estimated mean for each arm
- 3. Pull arm with highest mean
- 4. Repeat step 2 & 3 forever
But, in our case we have to set the amount of traffic. Set it to be proportional to \( P(I_a) \):

- **(1)** Simulate many draws from \( Beta(\alpha + S_a, \beta + F_a) \):

<table>
<thead>
<tr>
<th>Time</th>
<th>Arm 1</th>
<th>Arm 2</th>
<th>Arm 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.54</td>
<td>0.73</td>
<td>0.74</td>
</tr>
<tr>
<td>2</td>
<td>0.55</td>
<td>0.66</td>
<td>0.73</td>
</tr>
<tr>
<td>3</td>
<td>0.53</td>
<td>0.81</td>
<td>0.80</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **(2)** The probability that arm \( a \) is optimal is the empirical fraction of rows for which arm \( a \) had the largest simulated value

- **(3)** Set traffic to arm \( a \) to be equal to \% of wins of arm \( a \)
Imagine you have two versions of the website and you’d like to test which one is better

- Version A has engagement rate of 5%
- Version B has engagement rate of 4%

You want to establish with 95% confidence that version A is better

- You’d need 22,330 observations (11,165 in each arm) to establish that
  - Use t-test to establish the sample size

Can bandits do better?
**Example**

**A/B test:** We need 22,330 observations. Assuming 100 observations/day, we need 223 days

- On 1st day about 50 sessions are assigned to each arm
- Suppose A got really lucky on the first day, and it appears to have a 70% chance of being superior

- Then we assign it 70% of the traffic on the second day, and the variant B gets 30%

- At the end of the 2nd day we accumulate all the traffic we’ve seen so far (over both days), and recompute the probability that each arm is best
The experiment finished in 66 days, so it saved you 157 days of testing (66 vs 223)
Generalization to multiple arms

- Easy to generalize to multiple arms:

![Graph showing optimal arm probabilities and true success rate over time and across arms. The graph includes multiple lines for different arm probabilities and success rates, with time period on the x-axis and arm on the y-axis.]