Learning through Experimentation

CS246: Mining Massive Datasets
Jure Leskovec, Stanford University
http://cs246.stanford.edu
Learning through Experimentation

- **Web advertising**
  - We discussed how to match advertisers to queries in real-time
  - But we did not discuss how to estimate **CTR**

- **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

3/7/2013

Example: Web Advertising

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

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

  Bid amount for ad $i$ on query $q$
  (Known)
Other Applications

- **Clinical trials:**
  - Investigate effects of different treatments while minimizing patient losses

- **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 $i$

- **Wins** (reward=1) with fixed (unknown) prob. $\mu_i$
- **Loses** (reward=0) with fixed (unknown) prob. $1-\mu_i$
- All draws are independent given $\mu_1 \ldots \mu_k$
- **How to pull arms to maximize total reward?**
How does this map to our setting?

- Each query is a bandit
- Each ad is an arm
- We want to estimate the arm’s probability of winning $\mu_i$ (i.e., ad’s the CTR $\mu_i$)
- Every time we pull an arm we do an ‘experiment’
The setting:
- Set of \( k \) choices (arms)
- Each choice \( i \) is associated with unknown probability distribution \( P_i \), supported in \([0,1]\)
- We play the game for \( T \) rounds
- In each round \( t \):
  - \((1)\) We pick some arm \( j \)
  - \((2)\) We obtain random sample \( X_t \) from \( P_j \)
    - Note reward is independent of previous draws
- Our goal is to maximize \( \sum_{t=1}^{T} X_t \)
- But we don’t know \( \mu_i \)! But every time we pull some arm \( i \) we get to learn a bit about \( \mu_i \)
Online Optimization

- Online optimization with limited feedback

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

- **Policy**: a strategy/rule that in each iteration tells me which arm to pull
  - Hopefully policy depends on the history of rewards

- How to quantify performance of the algorithm? Regret!
Let be $\mu_i$ the mean of $P_i$
Payoff/reward of best arm: $\mu^* = \max_i \mu_i$
Let $i_1, i_2 \ldots i_T$ be the sequence of arms pulled
Instantaneous regret at time $t$: $r_t = \mu^* - \mu_i$
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 $T \to \infty$
Allocation Strategies

- If we knew the payoffs, which arm would we pull?
  \[\text{Pick } \arg \max_i \mu_i\]

- What if we only care about estimating payoffs \(\mu_i\)?
  - Pick each arm equally often: \(\frac{T}{k}\)
  - Estimate: \(\hat{\mu}_i = \frac{k}{T} \sum_{j=1}^{T} X_{i,j}\)
  - Regret: \(R_T = \frac{T}{k} \sum_i (\mu^* - \mu_i)\)
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**.
The example illustrates a classic problem in decision making:

- We need to trade off **exploration** (gathering data about arm payoffs) and **exploitation** (making decisions based on data already gathered)

- **The Greedy does not explore sufficiently**
  - **Exploration**: Pull an arm we never pulled before
  - **Exploitation**: Pull an arm for which we currently have the highest estimate of $\mu_i$
The problem with our Greedy algorithm is that it is too certain in the estimate of $\mu_i$
- When we have seen a single reward of 0 we shouldn’t conclude the average reward is 0

Greedy does not explore sufficiently!
# New Algorithm: Epsilon-Greedy

## Algorithm: Epsilon-Greedy

- **For** $t=1:T$
  - Set $\epsilon_t = O(1/t)$
  - **With prob. $\epsilon_t$: Explore** by picking an arm chosen uniformly at random
  - **With prob. $1 - \epsilon_t$: Exploit** by picking an arm with highest empirical mean payoff
- **Theorem [Auer et al. ‘02]**
  
  For suitable choice of $\epsilon_t$ it holds that

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

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
Comparing 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 (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 $\mu_i$ 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).

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

\[ \mu_i \]

99.99% confidence interval

After more exploration
Suppose we fix arm $i$

- Let $Y_1 \ldots Y_m$ be the payoffs of arm $i$ in the first $m$ trials
- Mean payoff of arm $i$: $\mu = E[Y]$
- Our estimate: $\hat{\mu}_m = \frac{1}{m} \sum_{l=1}^{m} Y_l$
- Want to find $b$ such that with high probability $|\mu - \hat{\mu}_m| \leq b$
  - Also want $b$ to be as small as possible (why?)

**Goal:** Want to bound $P(|\mu - \hat{\mu}_m| \leq b)$
Hoeffding’s Inequality

- **Hoeffding’s inequality:**
  - Let $X_1 \ldots X_m$ be i.i.d. rnd. vars. taking values in $[0,1]$
  - Let $\mu = E[X]$ and $\hat{\mu}_m = \frac{1}{m} \sum_{l=1}^{m} X_l$
  - Then: $P(|\mu - \hat{\mu}_m| \leq b) \leq 2 \exp(-2b^2m) = \delta$

- To find out $b$ we solve
  - $2e^{-2b^2m} \leq \delta$ then $-2b^2m \leq \ln(\delta/2)$
  - So: $b \geq \sqrt{\frac{\ln\left(\frac{2}{\delta}\right)}{2m}}$
The UCB1 Algorithm

- **UCB1 (Upper confidence sampling) algorithm**
  - Set: $\widehat{\mu}_1 = \cdots = \widehat{\mu}_k = 0$ and $n_1 = \cdots = n_k = 0$
  - For $t = 1:T$
    - For each arm $i$ calculate: $UCB(i) = \widehat{\mu}_i + \sqrt{\frac{2 \ln t}{n_i}}$
    - Pick arm $j = \text{arg max}_i UCB(i)$
    - Pull arm $j$ and observe $y_t$
    - Set: $n_j \leftarrow n_j + 1$ and $\widehat{\mu}_j \leftarrow \frac{1}{n_j} (y_t - \widehat{\mu}_j)$

- **Optimism in face of uncertainty**
  - The algorithm believes that it can obtain extra rewards by reaching the unexplored parts of the state space

[Auer et al. ‘02]

Optimism in face of uncertainty
The UCB₁ Algorithm

\[ UCB(i) = \hat{\mu}_i + \sqrt{\frac{2 \ln t}{n_i}} \]

- Confidence bound **grows** with the total number of actions we have taken
- But **shrinks** with the number of times we have tried this particular action
- This ensures each action is tried infinitely often but still balances exploration and exploitation
Theorem [Auer et al. 2002]

Suppose optimal mean payoff is $\mu^* = \max_i \mu_i$

And for each arm let $\Delta_i = \mu^* - \mu_i$

Then it holds that

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

So: $O\left(\frac{R_T}{T}\right) = k \frac{\ln T}{T}$
**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)
News Recommendation

- Every round receive context [Li et al., WWW ‘10]
  - Context: User features, articles view before
- Model for each article’s click through rate
News Recommendation

- Feature-based exploration:
  - Select articles to serve users based on contextual information about the user and the articles
  - Simultaneously adapt article selection strategy based on user-click feedback to maximize total number of user clicks
Contextual Bandits

- **Contextual bandit algorithm in round** $t$
  - **(1)** Algorithm observes user $u_t$ and a set $A_t$ of arms together with their features $x_{t,a}$
    - Vector $x_{t,a}$ summarizes both the user $u_t$ and arm $a$
    - We call vector $x_{t,a}$ the **context**
  - **(2)** Based on payoffs from previous trials, algorithm chooses arm $a \in A_t$ and receives payoff $r_{t,a}$
    - Note only feedback for the chosen $a$ is observed
  - **(3)** Algorithm improves arm selection strategy with observation $(x_{t,a}, a, r_{t,a})$
LinUCB Algorithm (1)

- Payoff of arm $\alpha$: $E[r_{t,a} | x_{t,a}] = x_{t,a}^T \cdot \theta^*_\alpha$
  - $x_{t,a}$ ... $d$-dimensional feature vector
  - $\theta^*_\alpha$... unknown coefficient vector we aim to learn
    - Note that $\theta^*_\alpha$ are not shared between different arms!
- How to estimate $\theta^*_\alpha$?
  - $D_\alpha$... $m \times d$ matrix of $m$ training inputs $[x_{a,t}]$
  - $c_\alpha$... $m$-dim. vector of responses to $\alpha$ (click/no-click)
  - Linear regression solution to $\theta^*_\alpha$ is then
    $$\hat{\theta}_\alpha = (D_\alpha^T D_\alpha + I_d)^{-1} D_\alpha^T c_\alpha$$

And $I_d$ is $d \times d$ identity matrix
LinUCB Algorithm (2)

- One can then show (using similar techniques as we used for UCB) that

\[ \left| x_{t,a}^\top \hat{\theta}_a - \mathbb{E}[r_{t,a} \mid x_{t,a}] \right| \leq \alpha \sqrt{x_{t,a}^\top (D_a^\top D_a + I_d)^{-1} x_{t,a}} \]

\[ \alpha = 1 + \sqrt{\ln(2/\delta)/2} \]

- So LinUCB arm selection rule is:

\[ a_t \overset{\text{def}}{=} \arg \max_{a \in A_t} \left( x_{t,a}^\top \hat{\theta}_a + \alpha \sqrt{x_{t,a}^\top A_a^{-1} x_{t,a}} \right) \]

\[ A_a \overset{\text{def}}{=} D_a^\top D_a + I_d \]
Algorithm 1 LinUCB with disjoint linear models.

0: Inputs: $\alpha \in \mathbb{R}_+$
1: for $t = 1, 2, 3, \ldots, T$ do
2:   Observe features of all arms $a \in \mathcal{A}_t$: $x_{t,a} \in \mathbb{R}^d$
3:   for all $a \in \mathcal{A}_t$ do
4:     if $a$ is new then
5:       $A_a \leftarrow I_d$ (d-dimensional identity matrix)
6:       $b_a \leftarrow 0_{d \times 1}$ (d-dimensional zero vector)
7:     end if
8:     $\hat{\theta}_a \leftarrow A_a^{-1} b_a$
9:     $p_{t,a} \leftarrow \hat{\theta}_a x_{t,a} + \alpha \sqrt{x_{t,a}^T A_a^{-1} x_{t,a}}$
10:   end for
11: Choose arm $a_t = \arg \max_{a \in \mathcal{A}_t} p_{t,a}$ with ties broken arbitrarily, and observe a real-valued payoff $r_t$
12: $A_{a_t} \leftarrow A_{a_t} + x_{t,a_t} x_{t,a_t}^T$
13: $b_{a_t} \leftarrow b_{a_t} + r_t x_{t,a_t}$
14: end for
What to put in slots F1, F2, F3, F4 to make the user click?
Results

The diagram shows the performance of different algorithms as the data size decreases. The algorithms include:
- ε-greedy
- ucb
- ε-greedy (seg)
- ucb (seg)
- ε-greedy (disjoint)
- linucb (disjoint)
- ε-greedy (hybrid)
- linucb (hybrid)
- omniscient

The y-axis represents the ctr (conversion rate), and the x-axis represents the data size in percentage: 100%, 30%, 20%, 10%, 5%, and 1%.

As the data size decreases, the performance of all algorithms is affected, with the ε-greedy and ucb algorithms showing a more significant drop in ctr compared to the hybrid and omniscient approaches.
Relevance vs. Diversity

- Want to choose a set that caters to as many users as possible
- Users may have different interests, queries may be ambiguous
- Want to optimize both the relevance and diversity
3 Announcements
(1) Last Class

- Last class meeting (Thu, 3/14) is canceled (sorry!)
- I will prerecord the last lecture and it will be available via SCPD on Thu 3/14
  - Last lecture will give an overview of the course and discuss some future directions
(2) Final Exam Logistics
Final: At Stanford

- Alternate final:
  Tue 3/19 6:00-9:00pm in 320-105
  - We have 100 slots. First come first serve!

- Final:
  Fri 3/22 12:15-3:15pm in CEMEX Auditorium
  - See http://campus-map.stanford.edu
  - Practice finals are posted on Piazza

- SCPD students can take the exam at Stanford!
Exam protocol for SCPD students:

- On Monday 3/18 your exam proctor will receive the PDF of the final exam from SCPD
- If you will take the exam at Stanford:
  - Ask the exam monitor to delete the SCP email
- If you won’t take the exam at Stanford:
  - Arrange 3h slot with your exam monitor
  - Take the exam
- Email exam PDF to cs246.mmds@gmail.com by Thursday 3/21 5:00pm Pacific time
(3) CS341: Project in Mining Massive Datasets
Data mining research project on real data

- Groups of 3 students
- We provide interesting data, computing resources (Amazon EC2) and mentoring
- You provide project ideas
- There are (practically) no lectures, only individual group mentoring

Information session:
Thursday 3/14 6pm in Gates 415
(there will be pizza!)
CS341: Schedule

- Thu 3/14: Info session
  - We will introduce datasets, problems, ideas
- Students form groups and project proposals
- Mon 3/25: Project proposals are due
- We evaluate the proposals
- Mon 4/1: Admission results
  - 10 to 15 groups/projects will be admitted
- Tue 3/30, Thu 5/2: Midterm presentations
- Tue 6/4, Thu 6/6: Presentations, poster session