Final Project Reports

• Suggested Lengths
  – 1 person ~ 6 to 8 pages
  – 2 persons ~ 7 to 10 pages
  – 3 persons ~ 8 to 12 pages

• If your results did not turn out as well as you had hoped
  – Don’t panic!
  – Clearly describe what you did, your results, and provide as much insight as you can into why the results turned out the way they did

• Key things to keep in mind
  – Structure: Introduction, Goals, Related Work, Methods/Approach, Results, Discussion
  – Figures can be very useful
  – Write clearly – check your writing – explain your methods clearly
  – It’s good to have details…but insight is important

• Overall: your reports should be along the lines of a “mini” research paper
Internet Advertising, Bids, and Auctions
“Computational Advertising”

• Revenue of many internet companies is driven by advertising

• Key problem:
  – Given user data:
    • Pages browsed
    • Keywords used in search
    • Demographics
  – Determine the most relevant ads (in real-time)
  – About 50% of keyword searches can not be matched effectively to any ads
  – Other aspects include bidding/pricing of ads

• New research area of “computational advertising”
  – See link to Stanford class by Andrei Broder on class Web site
Why is Advertising Important for Internet Companies?

Internet adspend by type 2012-2015 (US$bn)

- **2012**: Total display 33.7, Classified 11.0, Paid search 40.7
- **2013**: Total display 40.2, Classified 11.5, Paid search 46.9
- **2014**: Total display 48.2, Classified 12.0, Paid search 53.6
- **2015**: Total display 57.6, Classified 12.6, Paid search 61.4

*Source: ZenithOptimedia*

From Techcrunch.com, Sept 30, 2013
Types of Online Ads

- **Display or Banner**
  - Fixed content, usually visual
  - Or (more recently) video ads

- **Sponsored search (Text Ad)**
  - Triggered by search results
  - Ad selection based on search query terms, user features, click-through rates, ....

- **Context-based/Text (Text Ad)**
  - Can be based on content of Web page during browsing
  - Ad selection based on matching ad content with page content
Participants in Online Advertising

• Publishers
  – Provide the space on Web pages for the ads
  – e.g., Search engines, Yahoo front page, CNN, New York Times, WSJ

• Advertisers
  – Provide the ads
  – e.g., Walmart, Ford, Target, Toyota...

• Ad Exchanges
  – Match the advertisers and publishers in real-time
  – e.g., Doubleclick, Google, etc
  – Contract with advertisers to run advertising campaigns, e.g., deliver up to 100k clicks using up to 10 million impressions in 30 days
  – Ad-server runs complex prediction/optimization software (in real-time) to optimize revenue (from ad-server’s viewpoint)
Concepts in Online Advertising

- Impression: showing an ad to an online user
  - CTR = clickthrough rate (typically around 0.1%)

- Revenue mechanisms (to ad-exchange or publisher, from advertiser)
  - CPM: cost per 1000 impressions
  - CPC: cost per click
  - CPA: cost per action (e.g., customer signs up, makes a purchase..)

- Ad-exchanges and auctions
  - Impressions can be bid on in real-time in ad-exchanges
  - Typically a 2\textsuperscript{nd}-price (Vickery) auction
  - Key to success = accurate prediction of CTR for each impression
Each ? represents an “ad slot”

In real-time the ad-exchange will compute which ads to show a particular user.
These ads are “impressions”
Simplified View of Advertising (Publisher View)

Users visiting a Web site (a publisher) and being served ads

Ad Exchange sells “slots” via the Publisher’s Web page via real-time auctions

Advertisers bid on ad slots in real-time

User 1
User 2
User 3
User 4
User 5
User 6
User 7
User 8
User 9
User 10

Publisher

Ad Exchange

Advertiser A
Advertiser B
Advertiser C
Advertiser D
Advertiser E
Publisher Share of Display Ad Impressions

Source: comScore Ad Metrix, U.S., Q3 2011

- 28% Facebook
- 12% Yahoo! Sites
- 4% Microsoft Sites
- 4% Google Sites
- 3% AOL, Inc.
- 49% Others
Top Ten U.S. Online Display Ad Publishers by Number of Impressions in Millions

Source: comScore Ad Metrix, Jan-2011 to Dec-2011, U.S.

- FACEBOOK.COM: 1,343,170
- YAHOO! SITES: 528,993
- MICROSOFT SITES: 215,650
- GOOGLE SITES: 179,929
- AOL, INC.: 131,373
- TURNER DIGITAL: 73,588
- GLAM MEDIA: 54,810
- ESPN: 47,096
- VIACOM DIGITAL: 38,532
- EBAY: 34,464
Simplified View of Advertising (Advertiser View)

Users visiting Web sites (publishers) and being served ads

Publishers selling “inventory” (ad slots) on an Ad Exchange

An Advertiser making an ad available to be shown to some set of users
Top Ten U.S. Online Display Advertisers by Number of Impressions in Millions

Source: comScore Ad Metrix, Jan-2011 to Dec-2011, U.S.

- AT&T INC. 105,792
- EXPERIAN INTERACTIVE 67,565
- VERIZON COMMUNICATIONS 49,481
- SCOTTRADE, INC. 44,031
- GOOGLE INC. 40,454
- MICROSOFT CORPORATION 38,662
- NETFLIX, INC. 36,991
- EBAY, INC. 32,302
- PROGRESSIVE CORPORATION 30,042
- IAC - INTERACTIVECORP 29,403
Behind the Scenes...

- The previous slides are a very simplified picture of how these systems work.......... in practice there are many other factors

- Multiple 3rd party “advertising companies”
  - In practice rather than just a single “ad exchange” there is a whole “ecosystem” of different systems and companies that sit between the publisher and the advertisers, optimizing different parts of the ad matching process

- Auction mechanisms
  - Use of “2nd price auctions”
Auctions and Bidding for Queries

• Say we have a query (like “flower delivery”)

• Different advertisers can bid to have their ad shown whenever this search query is entered by a user

• Say there are $K$ different positions on the search results page, each with different likelihood of being seen by user
  – For simplicity imagine that they are in a vertical column with $K$ positions, top to bottom

• Advertisers submit bids (in real-time) in terms of how much they are willing to pay the search engine for a click on their ad (CPC model)
  – Tradeoff between the getting a good position and paying too much

• So there is an auction (often in real-time) among the advertisers
Auction Mechanisms

- Initial Internet advertisers paid flat fees to search engines (per impression)

- Overture (later purchased by Yahoo!) in 1997 introduced the notion of bidding and auctions
  - Advertisers submitted bids indicating what they would pay (CPC) for a keyword
  - Improvement over flat fees.....but found to be inefficient/volatile, with rapid price swings, which discouraged advertisers from participating

- 2002: Google introduced the idea of 2nd price Auctions for keyword bidding
  - Advertisers make bids on K positions, bids are ranked in positions 1 through K
  - Advertiser in position k is charged the bid of advertiser in position k+1 plus some minimum (e.g., 1 cent)
  - Advertiser in Kth position is charged a fixed minimum amount
  - Google (and others) quickly noticed that this made the auction market much more stable an “user-friendly”, much less susceptible to gaming
    (Yahoo!/Overture also switched to this method)
  - Google’s AdWords uses a modified ranking:
    - Instead of ranking by Bid it ranks by Bid * Estimated CTR
Example of 2\textsuperscript{nd} Price Auction Bidding Work?

- 2 slots and 3 advertisers
  - So the advertisers want to (a) get a slot, and (b) get the best slot

- Advertisers place a true value on a click of $10, $4, $2 respectively
  - This notion of “true value” is important
  - It is what an advertiser truly believes a click on their ad is worth
  - Or in other words, it is the maximum they should be willing to pay

- 2\textsuperscript{nd} price auction: each advertiser bids their true value
  - Advertiser 1 is ranked 1st, gets slot 1, and pays $4 + 1 cent
  - Advertiser 2 is ranked 2nd, gets slot 2, and pays $2 + 1 cent
  - Advertiser 3 is ranked 3\textsuperscript{rd} and gets no slot
2nd Price Auctions

• Various economic arguments as to why this is much more efficient than 1st price auctions
  – Advertisers have no incentive to bid anything other than their true value
  – This discourages advertisers from dynamically changing bids, which was a cause of major instability in earlier first-price auctions

• Methods seems to work particularly well for internet advertising

• References:
Google’s second price auction

<table>
<thead>
<tr>
<th>advertiser</th>
<th>bid</th>
<th>CTR</th>
<th>ad rank</th>
<th>rank</th>
<th>paid</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$4.00</td>
<td>0.01</td>
<td>0.04</td>
<td>4</td>
<td>(minimum)</td>
</tr>
<tr>
<td>B</td>
<td>$3.00</td>
<td>0.03</td>
<td>0.09</td>
<td>2</td>
<td>$2.68</td>
</tr>
<tr>
<td>C</td>
<td>$2.00</td>
<td>0.06</td>
<td>0.12</td>
<td>1</td>
<td>$1.51</td>
</tr>
<tr>
<td>D</td>
<td>$1.00</td>
<td>0.08</td>
<td>0.08</td>
<td>3</td>
<td>$0.51</td>
</tr>
</tbody>
</table>

- **bid**: maximum bid for a click by advertiser
- **CTR**: click-through rate: when an ad is displayed, what percentage of time do users click on it? **CTR is a measure of relevance.**
- **ad rank**: bid × CTR: this trades off (i) how much money the advertiser is willing to pay against (ii) how relevant the ad is
- **rank**: rank in auction
- **paid**: second price auction price paid by advertiser

Second price auction: The advertiser pays the minimum amount necessary to maintain their position in the auction (plus 1 cent).
Keywords with high bids

According to http://www.cwire.org/highest-paying-search-terms/

$69.1  mesothelioma treatment options
$65.9  personal injury lawyer michigan
$62.6  student loans consolidation
$61.4  car accident attorney los angeles
$59.4  online car insurance quotes
$59.4  arizona dui lawyer
$46.4  asbestos cancer
$40.1  home equity line of credit
$39.8  life insurance quotes
$39.2  refinancing
$38.7  equity line of credit
$38.0  lasik eye surgery new york city
$37.0  2nd mortgage
$35.9  free car insurance quote
Top 20 most expensive keywords in Google AdWords Advertising

Source: http://www.wordstream.com/download/docs/most-expensive-keywords.pdf
### Examples of Costs per Click

<table>
<thead>
<tr>
<th>Metric</th>
<th>2010</th>
<th>2011</th>
<th>2012</th>
<th>2013</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost per click (CPC)</td>
<td>$1.24</td>
<td>$1.04</td>
<td>$0.84</td>
<td>$0.92</td>
</tr>
<tr>
<td>Click through rate (CTR)</td>
<td>0.7%</td>
<td>0.4%</td>
<td>0.5%</td>
<td>0.5%</td>
</tr>
<tr>
<td>Average Ad Position</td>
<td>3.7</td>
<td>3.0</td>
<td>2.6</td>
<td>2.1</td>
</tr>
<tr>
<td>Conversion rate</td>
<td>6.8%</td>
<td>5.3%</td>
<td>3.4%</td>
<td>8.8%</td>
</tr>
<tr>
<td>Cost per conversion</td>
<td>$13.14</td>
<td>$19.74</td>
<td>$24.40</td>
<td>$10.44</td>
</tr>
<tr>
<td>Invalid click rate</td>
<td>6.7%</td>
<td>10.9%</td>
<td>8.0%</td>
<td>8.3%</td>
</tr>
</tbody>
</table>

Predicting Click-Through Rates for Online Advertisements
Optimally Matching Advertisements to Users

• Advertising is a very large component of revenue for search engines
  – Displaying the “best” set of ads to users is a key issue

• Problem Statement (from search engine’s perspective)
  – Inventory = a set of possible ads that could be shown
  – Query = query string typed in by a user
  – Problem: what is the best set of ads to show the user, and in what positions

• This is a complicated optimization problem
  – Objectives:
    • Search engine: maximize revenue (usually by attracting clicks)
    • Advertiser: maximize click rate
    • User: only wants to see relevant ads (overall user quality)
  – Other aspects
    • Each advertiser may only want to show a fixed maximum number of ads
    • User saturation if they see the same ad multiple times
    • Click fraud, etc
Cost-Per-Click (CPC) Model

- Cost-Per-Click, or CPC:
  - Search engine is paid every time an ad is clicked by a user

- Simple Expected Revenue Model
  \[ E[\text{revenue}] = p(\text{click} | \text{ad}) \cdot \text{CPC}_{\text{ad}} \]

- Simple heuristic
  - Order the ads in terms of expected revenue
### Examples of Costs per Click

<table>
<thead>
<tr>
<th>Metric</th>
<th>2010</th>
<th>2011</th>
<th>2012</th>
<th>2013</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost per click (CPC)</td>
<td>$1.24</td>
<td>$1.04</td>
<td>$0.84</td>
<td>$0.92</td>
</tr>
<tr>
<td>Click through rate (CTR)</td>
<td>0.7%</td>
<td>0.4%</td>
<td>0.5%</td>
<td>0.5%</td>
</tr>
<tr>
<td>Average Ad Position</td>
<td>3.7</td>
<td>3.0</td>
<td>2.6</td>
<td>2.1</td>
</tr>
<tr>
<td>Conversion rate</td>
<td>6.8%</td>
<td>5.3%</td>
<td>3.4%</td>
<td>8.8%</td>
</tr>
<tr>
<td>Cost per conversion</td>
<td>$13.14</td>
<td>$19.74</td>
<td>$24.40</td>
<td>$10.44</td>
</tr>
<tr>
<td>Invalid click rate</td>
<td>6.7%</td>
<td>10.9%</td>
<td>8.0%</td>
<td>8.3%</td>
</tr>
</tbody>
</table>

Expected Revenue Model

- Simple Expected Revenue Model
  \[ E[\text{revenue}] = CTR_{ad} \times CPC_{ad} = p(\text{click} | \text{ad}) \cdot CPC_{ad} \]

- \( CPC_{ad} \) is known ahead of time: the key problem is estimating CTR

- Typically we also condition on additional factors beyond the ad itself, e.g.,
  - We really want to estimate \( p(\text{click} | \text{ad}, \text{query}, \text{user}, \text{ad\_position}) \)
  - For simplicity we will ignore everything except “ad” here

- If we have some click data we can just estimate
  \[ P(\text{click} | \text{ad}) = \frac{\text{(number of clicks)}}{\text{(number of times ad was shown)}} \]

- Typical click through rates are small, e.g., 1 in 1000 or 1 in 10000
  - So we are typically trying to estimate the probability of a rare event
Computing the CTR from Click Data

- Estimate of CTR = \( \frac{\text{number of clicks}}{\text{number of views}} \)

- Number of clicks = number of times ad was clicked

- Number of views?
  - Use a “discount” model based on eye-tracking to estimate how many times the ad was seen by users
  - So number of views is total number of times ad was shown, “discounted” by position model
Eye-Tracking: The Golden Triangle for Search
from Hotchkiss, Alston, Edwards, 2005; EnquiroResearch
Simple Example of CTR Estimation

- Assume that the true $P(\text{click} \mid \text{ad}) = 10^{-4}$
  - Say we have seen $r$ clicks, from $N$ showings of the ad
  - Our estimate of $P(\text{click} \mid \text{ad}) = P' = \frac{r}{N}$

- What is our uncertainty about $P'$?
  Simple binomial model, assume $Np > 5$, i.e., $N > 5 \times 10^4$ in our problem
  -> 95% confidence interval is
  \[w = 1.96 \sqrt{p(1-p)/\sqrt{N}} \approx 0.02/\sqrt{N}\]

  Say we want $w < 10^{-5}$ (10% of the true value)

  Rearranging terms above this means we need
  \[\sqrt{N} > 0.02 \ 10^5 \ \text{or} \ \ N > 4 \times 10^6\]

  This means we need a very large $N$ to be confident in our estimation of small probabilities
Difficulty of CTR Prediction Problem

• Clickthrough rates are small -> need large number of impressions to get reliable estimates

• Every day there will be a large number of new ads that the ad placement algorithm has not seen before, i.e., with unknown CTR

• Making mistakes is expensive
  – Say we show ad A 10 million times, and the CPC is $1 with a true CTR of $10^{-4}$
  – And we don’t show ad B, which has a CPC of $1 with a true CTR of $10^{-2}$
  – Then the “cost of learning” about ad A (versus not showing B) is $10^{-2} \times 10 \text{ million}$, or $100,000$ (!)
More Sophisticated Methods

• Ad = terms in the ad + keywords bid on by the advertiser

• Use machine learning to predict CTR based on
  – Features of ads, terms, and advertisers

• Conduct active online experiments among different ads
  – “explore/exploit” problem
  – Can be modeled as a stochastic multi-arm bandit problem
Learning to Predict CTR

- Assume a historical database consisting of many ads with
  1. Features that can be computed a priori before the ad is shown to users
     - Text in the ad and in the title of the ad
     - Bid terms or keywords (which query terms it will be matched to)
  2. Numbers of clicks and views (and CTR) for the ad after it was shown to users

- The goal is to predict item 2 (future CTR) given item 1 (ad features)
  - This will help in the “cold-start” problem of ranking new ads so that the highest expected revenue ads are at the top

In the next few slides we describe the approach of Richardson, Dominowska, Ragno, WWW 2007, who used logistic regression for this problem
Recall: The Logistic Regression Classification Model

Notation:

- A $d$-dimensional feature vector $\mathbf{x}$ (e.g., word counts for a document)
- We assume one of the components of $\mathbf{x}$ is set to all 1’s (to give us an intercept term in the model)
- $c \in \{0, 1\}$ is a binary class label

Logistic regression model with parameter weights $\beta_1, \ldots, \beta_d$:

$$P(c_i = 1|\mathbf{x}) = \frac{1}{1 + e^{-\sum_{j=1}^{d} \beta_j x_j}}$$

We can interpret this model as a linear weighted sum of the inputs

$$z(\mathbf{x}) = \sum_{j=1}^{d} \beta_j x_j$$

where $z(\mathbf{x})$ is then put through a "squashing" logistic function, $\frac{1}{1 + e^{z(-\mathbf{x})}}$, to ensure that its values stay between 0 and 1.
Loss Functions

From Richardson, Dominowska, Ragno, WWW 2007

• Let $q_i$ be the CTE predicted by the model for an ad and $p_i$ be the measured future CTR (in the training data) for that ad

• MSE loss function

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (q_i - p_i)^2$$

• Cross-entropy (or Kullback-Leibler loss function)

$$E(w) = \frac{1}{N} \sum_{i=1}^{N} \left( q_i \log \frac{q_i}{p_i} + (1-q_i) \log \frac{1-q_i}{1-p_i} \right)$$

$w$ = weights of the logistic regression model

$E(w)$ is 0 if and only if $q_i = p_i$, for all $i$, otherwise is $> 0$
Training of Logistic Regression Model

From Richardson, Dominowska, Ragno, WWW 2007

• Training algorithm: used a variant of gradient descent
  – Limited memory L-BFGS method
  – Uses a memory efficient approximation to the full 2\textsuperscript{nd} order Hessian matrix

• Used cross-entropy loss function, \(E(w)\)

• Used squared error regularization, i.e., minimized \(E(w) + \lambda \sum w^2\)
  – Searched over \(\lambda = 0.01, 0.03, 0.1, 0.3, 1, 2, 10, 30, 100\) on a validation set
  – Found that \(\lambda = 0.01\) worked best

• Normalized all features to have mean 0 and standard deviation 1

• Any feature value more than 5\(\sigma\) away from the mean was moved to 5\(\sigma\)
  – Helps in reducing the effect of features with outlier values

• For each feature \(f_j\) also used \(\log(f_j + 1)\) as a feature
Features used in Learning to Predict CTR

From Richardson, Dominowska, Ragno, WWW 2007

- Term CTR = average CTR of ads in the training data with the same “bid terms”
  - Smooth towards the overall mean CTR for ads with new bid terms
  - Also used the number of other ads that have the same bid terms as a feature

- Related Term CTR
  - Average CTR of “related ads” - ads matched based on text similarity
  - Also used number of “related ads”
Features used in Learning to Predict CTR

From Richardson, Dominowska, Ragno, WWW 2007

- Term CTR = average CTR of ads in the training data with the same “bid terms”
  - Smooth towards the overall mean CTR for ads with new bid terms
  - Also used the number of other ads that have the same bid terms as a feature

- Related Term CTR
  - Average CTR of “related ads” - ads matched based on text similarity
  - Also used number of “related ads”

<table>
<thead>
<tr>
<th>Features</th>
<th>MSE (x 1e-3)</th>
<th>KL Divrg. (x 1e-2)</th>
<th>% Improv.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline ($\overline{CTR}$)</td>
<td>4.79</td>
<td>4.03</td>
<td>-</td>
</tr>
<tr>
<td>Term CTR</td>
<td>4.37</td>
<td>3.50</td>
<td>13.28%</td>
</tr>
<tr>
<td>Related term CTRs</td>
<td>4.12</td>
<td>3.24</td>
<td>19.67%</td>
</tr>
</tbody>
</table>

(Baseline = predict the average CTR for all ads)
Features based on “Order Specificity”

From Richardson, Dominowska, Ragno, WWW 2007

Orders placed by advertisers can contain more or less specific terms
Entropy of categories of order bid terms can be used as a feature
  - Produced an extra 5% improvement

<table>
<thead>
<tr>
<th>Title:</th>
<th>Buy shoes now,</th>
<th>Less Specific Terms -&gt; lower CTR?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text:</td>
<td>Shop at our discount shoe warehouse!</td>
<td></td>
</tr>
<tr>
<td>Url:</td>
<td>shoes.com</td>
<td></td>
</tr>
<tr>
<td>Terms:</td>
<td>{buy shoes, shoes, cheap shoes}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Title:</td>
<td>Buy [term] now,</td>
<td></td>
</tr>
<tr>
<td>Text:</td>
<td>Shop at our discount warehouse!</td>
<td></td>
</tr>
<tr>
<td>Url:</td>
<td>store.com</td>
<td></td>
</tr>
<tr>
<td>Terms:</td>
<td>{shoes, TVs, grass, paint}</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Order Specificity results

<table>
<thead>
<tr>
<th>Features</th>
<th>MSE (x 1e-3)</th>
<th>KL Divrg. (x 1e-2)</th>
<th>% Improv.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (CTR)</td>
<td>4.79</td>
<td>4.03</td>
<td>-</td>
</tr>
<tr>
<td>CTRs &amp; Ad Quality +Order</td>
<td>4.00</td>
<td>3.09</td>
<td>23.45%</td>
</tr>
<tr>
<td></td>
<td>3.75</td>
<td>2.86</td>
<td>28.97%</td>
</tr>
</tbody>
</table>
Variation in CTR across Type of Ad

From Richardson, Dominowska, Ragno, WWW 2007

Figure 3. CTR variance across all ads for several keywords. Horizontal bars show average CTR; the bottom of the vertical bar is the minimum CTR, and the top is the maximum CTR.
Ad Quality and Unigram Features

From Richardson, Dominowska, Ragno, WWW 2007

• 81 manually-defined features based on “Ad Quality”
  – Appearance: number of words in title? In body? Etc
  – Reputation: number of segments in display URL. Etc
  – Relevance: do bid (query) terms appear in the ad? Etc
  – And more....

• Unigram Features (Words in the Ad)
Unigram Features from Words in the Ad

- 10k most common words in ads (10k binary features)

"Shipping" appears much more often in high CTR ads. So certain terms in the ad may boost its CTR.

Figure 4. Frequency of advertisement word unigrams, sorted by overall frequency. The light and dark gray lines give the relative frequency of unigrams in low and high CTR ads.
## Improvement with “Ad Quality” Features

From Richardson, Dominowska, Ragno, WWW 2007

<table>
<thead>
<tr>
<th>Features</th>
<th>MSE ($x \ 1e^{-3}$)</th>
<th>KL Divrg. ($x \ 1e^{-2}$)</th>
<th>% Improv.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (CTR)</td>
<td>4.79</td>
<td>4.03</td>
<td>-</td>
</tr>
<tr>
<td>Related term CTRs</td>
<td>4.12</td>
<td>3.24</td>
<td>19.67%</td>
</tr>
<tr>
<td>+Ad Quality</td>
<td>4.00</td>
<td>3.09</td>
<td>23.45%</td>
</tr>
<tr>
<td>+Ad Quality without unigrams</td>
<td>4.10</td>
<td>3.20</td>
<td>20.72%</td>
</tr>
</tbody>
</table>
Features based on Search Engine Data

- For each term in an ad
  - Estimated number of pages found for this term in a search engine query
  - Frequency of queries for the ad term, based on a 3 month period of search query logs

Figure 5. Relative average CTR for ads displayed for each query frequency decile (in decreasing order), aggregated across all ranks.
Features based on Search Engine Data

For each term in an ad

- Estimated number of pages found for this term in a search engine query
- Frequency of queries for the ad term, based on a 3 month period of search query logs

Table 4: Search Engine Data results. AQ means the Ad Quality feature set, and OB means the Order Specificity.

<table>
<thead>
<tr>
<th>Features</th>
<th>MSE (x 1e-3)</th>
<th>KL Divrg. (x 1e-2)</th>
<th>% Improv.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (CTR)</td>
<td>4.79</td>
<td>4.03</td>
<td>-</td>
</tr>
<tr>
<td>+Search Data</td>
<td>4.68</td>
<td>3.91</td>
<td>3.11%</td>
</tr>
<tr>
<td>CTRs &amp; AQ &amp; OS</td>
<td>3.75</td>
<td>2.86</td>
<td>28.97%</td>
</tr>
<tr>
<td>+Search Data</td>
<td>3.73</td>
<td>2.84</td>
<td>29.47%</td>
</tr>
</tbody>
</table>
High and Low Weight Features

From Richardson, Dominowska, Ragno, WWW 2007

<table>
<thead>
<tr>
<th><strong>Top ten features</strong></th>
<th><strong>Bottom ten features</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>log(#chars in term)</td>
<td>log(# terms in order)</td>
</tr>
<tr>
<td>v_{12}</td>
<td>log(v_{0*})</td>
</tr>
<tr>
<td>v_{22}</td>
<td>sqr(p_{00})</td>
</tr>
<tr>
<td>log(order category entropy)</td>
<td>sqr(order category entropy)</td>
</tr>
<tr>
<td>log(#most common word)</td>
<td>log(#chars in landing page)</td>
</tr>
<tr>
<td>sqr(#segments in displayurl)</td>
<td>log(a_{01})</td>
</tr>
<tr>
<td>sqr(#action words in body)</td>
<td>a_{13}</td>
</tr>
<tr>
<td>p_{10}</td>
<td>sqr(p_{0*})</td>
</tr>
<tr>
<td>p_{**}</td>
<td>log(#chars in body)</td>
</tr>
<tr>
<td>log(v_{00})</td>
<td>sqr(#chars in term)</td>
</tr>
</tbody>
</table>
High and Low Weight Unigrams

From Richardson, Dominowska, Ragno, WWW 2007

Table 6: Unigrams with highest (and lowest) weight.

<table>
<thead>
<tr>
<th>Top ten unigrams</th>
<th>Bottom ten unigrams</th>
</tr>
</thead>
<tbody>
<tr>
<td>official</td>
<td>quotes</td>
</tr>
<tr>
<td>download</td>
<td>hotels</td>
</tr>
<tr>
<td>photos</td>
<td>trial</td>
</tr>
<tr>
<td>maps</td>
<td>deals</td>
</tr>
<tr>
<td>official</td>
<td>gift</td>
</tr>
<tr>
<td>direct</td>
<td>have</td>
</tr>
<tr>
<td>costumes</td>
<td>software</td>
</tr>
<tr>
<td>latest</td>
<td>engine</td>
</tr>
<tr>
<td>version</td>
<td>compare</td>
</tr>
<tr>
<td>complete</td>
<td>secure</td>
</tr>
</tbody>
</table>
Error Rate Evolution

From Richardson, Dominowska, Ragno, WWW 2007

Figure 6: Expected mean absolute error in CTR as a function of the number of times an ad is viewed.
Performance on Ads that get more Views

From Richardson, Dominowska, Ragno, WWW 2007

Table 7: Comparison of results for a model trained and tested on ads with over 100 views vs. over 1000 views.

<table>
<thead>
<tr>
<th>Features</th>
<th>%Improv &gt;100 views</th>
<th>%Improv &gt;1000 views</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (CTR)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>+Term CTR</td>
<td>13.28</td>
<td>25.22</td>
</tr>
<tr>
<td>+Related CTR</td>
<td>19.67</td>
<td>32.92</td>
</tr>
<tr>
<td>+Ad Quality</td>
<td>23.45</td>
<td>33.90</td>
</tr>
<tr>
<td>+Order Specificity</td>
<td>28.97</td>
<td>40.51</td>
</tr>
<tr>
<td>+Search Data</td>
<td>29.47</td>
<td>41.88</td>
</tr>
</tbody>
</table>
Predicting CTR given an Ad and a Query

See paper “Ad Click Prediction: a View from the Trenches”, McMahan et al (Google), SIGKDD 2013 (on the class Web page)

Estimate P(click | ad, query)
- Potentially millions of text features
- Extremely sparse (only a tiny fraction of non-zero values per row)
- Billions of predictions per day
- Model needs to be updated quickly as clicks and non-clicks are observed

Aspects of the Google approach
- Regularized logistic regression
- Trained with a variant of stochastic gradient ("FTRL-Proximal" algorithm)
- Importance of confidence estimates on CTR, and calibration
- Many engineering tricks to make this work at scale
- Significant emphasis on visualizing/diagnosing model performance
  (important to be able to track/detect the effect of any changes to the model)
Behavioral/Audience Targeting

• Statistical models used to predict best ads for each individual user
  – \( P(\text{click} \mid \text{ad features, query features, user features}) \)

• User features
  – Categories of sites visited
  – Categories of search query terms issued
  – Demographics (if available – may be inferred)
  – Typically exponentially-decayed over time

• Logistic regression widely used
  – to estimate \( P(\text{click} \mid \text{ad and user attributes}) \)

• Significant online updating and tuning
  – Highly non-stationary environment
Clustering URLs into Categories of URLs

Would like to use URLs as features – but we might have 100 million URLs

Can cluster URLs based on user access patterns e.g., using hierarchical clustering reduce to a few thousand meaningful clusters

From Media 6 Degrees presentation, Troy Raeder, Columbia Data Science Class
Online Learning of ClickThrough Rates
Online Learning of CTRs

• Once we begin to show ads, we would like to learn the CTRs

• Consider K different ads, with CTRs of $p_1, \ldots, p_K$

• We would like to learn these CTRs so that we can maximize expected revenue... but we don’t want to lose too much potential revenue in doing so

• This is an example of the “explore/exploit” problem
  – Explore: for each ad show it enough times so that we can learn its CTR
  – Exploit: once we find a good ad, or the best ad, we want to show it often so that we maximize expected revenue

• Problem: what is the optimal strategy for showing the K ads?
  – Strategy = sequence of (ad, click/no-click) pairs
The Multi-Armed Bandit Problem

- Model the explore/exploit problem as a “multi-armed bandit”, i.e., as a slot machine for gambling with $K$ arms

- Each “arm” corresponds to an ad, with “payoff” probability $p_k$, $k = 1, \ldots, K$
  - Assume for simplicity that if we pull an arm and “win” we get rewarded 1 unit

- Objective: construct $N$ successive pulls of the slot machine to maximize the expected total reward

- This is a well-studied problem in sequential optimization
  - Even earlier work dating back to the 1950’s
  - Other instances of this problem occur in applications where you have to make choices “along the way” from a finite set of options based only on partial information
Theoretical Framework

- **K** bandits, with payoff probabilities \( p_k \), \( k = 1, \ldots, K \), and unit rewards = 1
  - Assume for simplicity that \( p_k \) probabilities and rewards don’t change over time
  - Also assume that bandits are memoryless (as in coin-tossing)

- Let \( X_k \) be the reward on any trial for bandit \( k \). Assume for simplicity that
  \[ X_k = 1 \text{ with probability } p_k, \text{ and } = 0 \text{ with probability } 1 - p_k \]
  Expected reward from bandit \( k \) is \( E[X_k] = 1 \cdot p_k + 0 \cdot (1 - p_k) = p_k \)

- Optimal strategy to maximize the expected reward?
  - Always select the \( k \) value that maximizes \( E[X_k] \), i.e., the largest probability \( p_k \)
  - This optimal strategy exists only in theory, if we know the \( p_k \)’s (which we don’t)

- Various theoretical analyses look at what happens on average by using certain types of strategies.
  Expected Regret(S) = \( E [\text{reward} \mid \text{optimal strategy}] - E [\text{reward} \mid \text{strategy S}] \)
Naïve Strategies

• Deterministic Greedy Strategy:
  – at iteration N, pick the bandit that has performed best up to this time
  – Weakness?
    • Will under-explore bandits and may easily select a sub-optimal bandit forever

• Play-the-Winner Strategy
  – At iteration N
    • play the bandit from iteration N-1 if it was successful, otherwise
    • select another arm uniformly at random or cycle through them deterministically
  – This is the optimal thing to do if the bandit was successful at time N-1
  – But not necessarily optimal to switch away from this bandit if it failed
  – Thus, this strategy tends to switch too much and over-explorers
    (see Berry and Fristedt, Bandit Problems: Sequential Allocation of Experiments, Chapman & Hall, 1985)

Note that both strategies above perform even more poorly if the learning is happening in batch mode rather than at each iteration.
Simple Example of Multi-Armed Bandit Strategy

• Epsilon-Greedy Strategy
  – At iteration t in the algorithm
  – Select the best bandit (up to this point) with probability, $1 - \varepsilon$, e.g., $\varepsilon = 0.1$
  – Select one of the other K-1 bandits with probability $\varepsilon$
    • uniformly at random
    • or in proportion to their estimated $p_k$ at this point

• Key aspects of the strategy
  – How to select $\varepsilon$
    • If its too small, we won’t explore enough
    • If its too large, we won’t exploit enough
  – How do we define “best”?
    • E.g., raw frequency $p_k = r_k / N_k$, or a smoothed estimate?

• Weakness?
  • $\varepsilon$ is fixed: so it continues to explore with probability $\varepsilon$, long after the best bandit has been identified – and hence is suboptimal
**Other Examples of Strategies**

- **Epsilon-greedy where we decrease \( \epsilon \) as the experiment progress**
  - Makes intuitive sense: explore a lot at first, then start to exploit more
  - Adds an additional “tuning” parameter of how to decrease \( \epsilon \)

- **Epsilon-first Strategy**
  - Pure exploration followed by pure exploitation
  - First explore for \( \epsilon N \) trials, selecting bandits uniformly at random
  - Then exploit for \((1-\epsilon)N\) trials, selecting the best bandit from the explore phase

- **Theoretical analyses provide results like bounds on the rates at which arms should be played, as a function of the true (unknown) \( p_k \) values**
  - These results provide very useful insights and general guidance
  - But don’t provide specific strategies
Randomized Probability Matching Strategy

• Idea: number of pulls from bandit $k$ should be proportional to the probability that bandit $k$ is optimal
  – Also known as Thompson sampling or “Bayesian bandits”

• Let $P( p_k \mid r_k, N_k)$ be a Bayesian density on the value $p_k$
  – where $r_k, N_k =$ number of trials and successes with the $k$th bandit so far
  – $P( p_k \mid r_k, N_k)$ is our posterior belief about $p_k$, given the data $r_k, N_k$
  – e.g., using a Beta prior and a Beta posterior density

• At each iteration we do the following:
  – Sample $M$ values of $p_k$ for each bandit $k$ from its density $P( p_k \mid r_k, N_k)$
  – For each bandit compute $w_k =$ proportion of $M$ samples that bandit $k$ has the largest $p_k$ value
  – Select a bandit $k$ by sampling from the distribution $w = [w_1, \ldots, w_K]$ 
  – Update the $r_k, N_k$ values and update the density $P( p_k \mid r_k, N_k)$
Simulation example showing 1000 draws from posterior distributions on bandit probabilities

Note that the probability of selecting one of the 2 bandits is the proportion of samples above or below the x=y line

Figure 1. One thousand draws from the joint distribution of two independent beta distributions. In both cases, the horizontal axis represents a beta (20,30) distribution. The vertical axis is (a) beta(2,1) and (b) beta(20,10).

Figure from S. L. Scott, A modern Bayesian look at the multi-armed bandit, *Applied Stochastic Models in Business and Industry*, 26:639-658, 2010
Randomized Probability Matching Strategy

• Strengths
  – Works well on a wide-range of problems
  – Relatively simple to implement
  – Relatively free of tuning parameters
  – Flexible enough to accommodate more complicated versions of the problem
  – Balances exploration and exploitation in an intuitive way

• Weaknesses
  – Requires more computation to select an arm at each iteration
  – Theoretical results/guarantees, relative to other methods, not generally known (yet)

Click Fraud

• Click fraud = generation of artificial (non-human) clicks for ads

• Why?
  – Artificially increases the costs for the advertiser (for CPC)
  – Artificially increases the revenue of the site hosting the ad (for CPC)

• Click Quality Teams
  – All major search engines have full-time teams monitoring/managing click fraud
  – Use a combination of human analysis and machine learning algorithms

• Controversial topic
  – Advertisers say search engines are not doing enough, claim fraud clicks are > 20%
  – Search engines reluctant to publish too much data on frauds, claim fraud click percentage is much lower