Marketing and CS

Philip Chan

Enticing you to buy a product

1. What is the content of the ad?
2. Where to advertise?
   - TV, radio, newspaper, magazine, internet, …
3. Who is the target audience/customers?

- Which question is the most important?

Target customers

- The more you know about the customers
  - The more effective to find the “right” customers
- Advertising kids’ toys
  - Where to advertise?
  - How to advertise?

Traditional vs Modern Media

- Traditional media (TV, Newspaper, …)
  - non-interactive
  - mostly broadcast
- Modern media (via internet)
  - interactive
  - more individualize
  - more information on individuals

Problems to Study

- Problem 1
  - Ranking Ad’s on Search Engines
- Problem 2
  - Product Recommendation

Ranking Ad’s on Search Engines

Problem 1

Advertising on Search Engines

- User
  - Query
- Advertiser
  - Ad
  - Keyword
    - for triggering the ad to be considered
  - Bid on a keyword
    - How much it's willing to pay
    - https://adwords.google.com/select/KeywordToolExternal?defaultView=3
- Search Engine
  - Score and rank ad's to display
  - Advertiser pays only when its ad is clicked

Factors affecting the score

- Advertiser's bid
  - Highest bidder wins (auction)
  - Is that sufficient?

Factors affecting the score

- Advertiser's bid
  - Highest bidder wins (auction)
  - Is that sufficient?

  - Bigger companies have deeper pocket…
  - What if the ad is not relevant?
    - Bid on keywords that are very popular
      - e.g. "ipod" but selling furniture
  - What if the ad/company/product is not "well received"?

Importance of audience/customer

- If the ad is not relevant
  - The users don't click
  - Doesn't matter how high the advertiser bids

Importance of audience/customer

- If the ad is not relevant
  - The users don't click
  - Doesn't matter how high the advertiser bids

- Displaying ad's relevant to users is important
  - Advertisers get more visits/revenue
  - Search engines get more revenue
  - User experience is better

Problem Formulation

- Given (Input)
  - Ad
  - Keyword
  - Bid
  - Query
  - (part of the algorithm is to decide other factors)
- Find (Output)
  - Score of Ad

Ad Rank score [Google AdWords]

- Cost Per Click (CPC) bid
- Quality Score

- https://support.google.com/adwords/answer/1722122

Quality Score [Google AdWords]

- Ad's relevance
- Keyword relevance
- Landing page experience

Quality Score [Google AdWords]

- Clickthrough Rate (CTR) of ad via that keyword [clicks / displays]
- CTR of display URL (URL in the ad)
- CTR of other ad's of the advertiser

Quality Score [Google AdWords]

- Clickthrough Rate (CTR) of ad via that keyword [clicks / displays]
- CTR of display URL (URL in the ad)
- CTR of other ad's of the advertiser
- Relevance of keyword to ad
- Relevance of keyword to query

Quality Score [Google AdWords]

- Clickthrough Rate (CTR) of ad via that keyword [clicks / displays]
- CTR of display URL (URL in the ad)
- CTR of other ad's of the advertiser
- Relevance of keyword to ad
- Relevance of keyword to query
- Usefulness and clarity of landing page
- Relevance of landing page

Quality Score [Google AdWords]

- Clickthrough Rate (CTR) of ad via that keyword [clicks / displays]
- CTR of display URL (URL in the ad)
- CTR of other ad's of the advertiser
- Relevance of keyword to ad
- Relevance of keyword to query
- Usefulness and clarity of landing page
- Relevance of landing page

- Advertiser's performance in geographical location
- Ad's performance on a site
- Ad's performance on devices
- Others
- https://support.google.com/adwords/answer/2454010
- https://support.google.com/adwords/answer/1659694

Weighted Linear Sum

- *Score* = $w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_nx_n$

Product Recommendation

Problem 2

Product Recommendation

- Shopping sites: amazon, netflix, …
- To sell more products
  - Recommend products the customers might buy

Can you read minds?

- "Can you read minds?" (amazon.com recruitment T-shirt)
- Why does amazon.com want employees who can read minds?

Recommendation Systems

- amazon.com
  - based on what you have looked at, bought, on your wish list, what similar customers bought, …
  - recommends products
- netflix.com
  - based on your ratings of movies, what similar customers rate, …
  - recommends movies

Netflix Prize (2006)

- Task
  - Given customer ratings on some movies
  - Predict customer ratings on other movies
- If John rates
  - "Mission Impossible" a 5
  - "Over the Hedge" a 3, and
  - "Back to the Future" a 4,
  - how would he rate "Harry Potter", … ?
- Performance
  - Error rate (accuracy)
- www.netflixprize.com

Performance of Algorithms

- Root Mean Square Error (RMSE)

$$\sqrt{\frac{\sum_{i}^{n}(real_i - prediction_i)^2}{n}}$$

Cash Award

- Grand Prize
  - $1M
  - 10% improvement
  - by 2011 (in 5 years)

Leader Board

- Announced on Oct 2, 2006
- Progress
  - www.netflixprize.com/community/viewtopic.php?id=386
- Improvement by the top algorithm
  - after 1 week: ~ 0.9%
  - after 2 weeks: ~ 4.5%
  - after 1 month: ~ 5%
  - after 1 year: 8.43%
  - after 2 years: 9.44%
  - after ~3 years: 10.06% [July 26, 2009]

Problem Formulation

- Given (input)
  - Movie
    - MovieID, title, year
  - Customer:
    - CustID, MovieID, rating, date
- Find (output)
  - Rating of a movie by a user
- Simplification: no actors/actresses, genre, …

Netflix Data (1998-2005)

- Customers
  - 480,189 (ID: 1 – 2,649,429)
- Movies
  - 17,770 (ID: 1 – 17,770)
  - ID, title, year
- Ratings given in Training Set
  - 100,480,507
  - min=1; max=17,653; avg=209 ratings per customer
  - Rating scale: 1 – 5
  - Date
- Ratings to predict in Qualifying Set
  - 2,817,131
- About 1 GB (700 MB compressed)

Naïve Algorithm 1

- Calculate the average rating for each movie
- Always predict the movie average
  - with no regard to the customer
- RMSE =1.0515
- “improvement” = -11%

Naïve Algorithm 2

- For each movie
  - Instead of simple average
  - Weighted average
    - customers who have rated more movies are weighted higher
- RMSE = 1.0745
- "Improvement" = -13%

Naïve Algorithm 3

- Calculate the average rating for a customer
- Always predict the customer average
  - with no regard to the movies
- RMSE = 1.0422
- "Improvement" = -10%

Naïve Algorithm 4

- Weight the two average ratings by their standard deviation
- sm = stdev of movie ratings
- sc = stdev of customer ratings

$$rating(custID, movID) = \frac{sc \times avgRating(movID) + sm \times avgRating(custID)}{sc + sm}$$

- RMSE = 0.9989
- "Improvement" = - 5%

Getting more serious…

- Find customers who:
  - Rated the same movies
  - Gave the same ratings

Getting more serious…

- Find customers who:
  - Rated the same movies and
  - Gave the same ratings
  - How likely you'll find such customer?

Getting more serious…

- Find customers who:
  - Rated the same movies?
  - Gave the same ratings?
  - Rated the same movies and more?
    - Ratings might not be the same

Superset customers

- For each customer *X*
  1. Find “superset” customer *Y*
  2. Use the “superset” customers to predict *X*’s rating

Superset Example

	m1	m2	m3	m4	m5	m6	m7	m8	m9
c1	?		1		3		4	?	
c2	2		3		1		4	5	
c3	4	5	3	3	3		4	4	1
c4			3		2		4		
c5	3					4	1	3	3

- ? = unknown rating to be predicted
  - (for simplicity, only for c1)
- c2 and c3 are supersets of c1
- How to predict “?”

Algorithm for Rating Prediction

- Average the movie ratings of the “superset” users
- Can we improve this algorithm?

Algorithm for Rating Prediction

- Average the movie ratings of the “superset” users
- Weighted average based on how “close” the “superset” users are
  - distance(*X*, *Y*) = ?

Algorithm for Rating Prediction

- Average the movie ratings of the “superset” users
- Weighted average based on how “close” the “superset” users are
  - distance(*X*, *Y*) = “RMSE(*X* , *Y*)”

Algorithm for Rating Prediction

- Average the movie ratings of the “superset” users
- Weighted average based on how “close” the “superset” users are
  - distance(*X*, *Y*) = “RMSE(*X* , *Y*)”
  - But smaller distance, higher weight, so we want “similarity(*X*, *Y*) ” not “distance(*X*, *Y*) ”
  - similarity(*X*, *Y*) = ?

Algorithm for Rating Prediction

- Average the movie ratings of the “superset” users
- Weighted average based on how “close” the “superset” users are
  - distance(X, Y) = “RMSE(X , Y)”
  - But smaller distance, higher weight, so we want “similarity(X, Y) ” not “distance(X, Y) ”
  - similarity(X, Y) = maxDist - distance(X, Y)
  - maxDist = ?

Algorithm for Rating Prediction

- Average the movie ratings of the “superset” users
- Weighted average based on how “close” the “superset” users are
  - distance(X, Y) = “RMSE(X , Y)”
  - But smaller distance, higher weight, so we want “similarity(X, Y) ” not “distance(X, Y) ”
  - similarity(X, Y) = maxDist - distance(X, Y)
  - maxDist = 4

Euclidean Distance

- 2-dimensional
  - A: (x_1 , y_1)
  - B: (x_2 , y_2)
  - sqrt ($(x_1 - x_2)^2 + (y_1 - y_2)^2$)
- n-dimensional
  - A: (a_1 , a_2 , … , a_n)
  - B: (b_1 , b_2 , … , b_n)
  - sqrt(Σ_i (a_i - b_i)^2)
- Similarity
  - 1 / EuclideanDistance

Prediction Range

- Netflix allows rating prediction in fractional values, e.g. 3.4, but users can only rate in integers
  - Why?
- Do we want to predict smaller than 1 or larger than 5?
  - Why?

What if a customer doesn’t have a superset

- What to predict?

Key operation

- For each customer X
  1. Find “superset” customer Y
  2. Use the “superset” customers to predict X’s rating
- Which step is more time consuming?

Superset Problem

- $O(C^2)$ problem
  - C = number of customers
  - $C(C - 1)$ pairs of customers
    - check if A is a superset of B
    - check if B is a superset of A
    - could be neither, why?
- To find the supersets
  - ignore ratings

B is potentially a superset of A

B A	Y	N	?
Y			
N			
?			

B is potentially a superset of A

B A	Y	N	?
Y	T	F	F
N	T	T	T
?	T	F	F

Implementation for Superset

- Key operation:
  - Find supersets of customer *X*
- How to store which customer rated which movie?

2D Boolean Array

- *C* = number of customers
- *M* = number of movies
- *C* x *M* Boolean array
  - 480189 * 17770 * 1 = ~8 GB

Bit Vector

- Movie ID list for each customer
  - c1: 1,4,7,8
  - c2: 1,7
  - c1 is a superset of c2
- Bit (Boolean) Vectors

Bit Vector

- Movie ID list for each customer
  - c1: 1,4,7,8
  - c2: 1,7
  - c1 is a superset of c2
- Bit (Boolean) Vectors
  - c1: 10010011
  - c2: 10000010
  - c1 is a superset of c2

Bit Vectors

- 1 bit per Boolean value
- $\lceil 17770 \div 32 \rceil$ = 556 words per customer
- 480189 * 556 * 4 = ~1 GB
- If you have 1GB physical memory, is this a good idea?

Array of Linked Lists

- 100,480,507 ratings
- Each movie ID needs 2 bytes
  - 100,480,507 * 2 = ~0.2 GB
- Each pointer needs 4 bytes
  - ~0.4 GB
- Array of ~500K pointers, 4 bytes each
  - ~0.002 GB
- Total: ~0.6 GB
- If you have 1GB physical memory, is this a good idea?

Just storing the data

Data Structure	Size in GB
2D Boolean array	~8
Array of bit vectors	~1
Array of linked lists	~0.6

What is in the memory?

- Running on my office Linux machine:
  - Operating system
  - Web browser
  - Email reader
  - emacs
  - xterm
  - Viewer for pdf, ps
  - …

Superset Implementation 1

- Instead of storing the movie IDs in memory
  - Read from the text files when needed
  - Each customer has two text files
    - Training set
    - Qualifying set
- Use pointer arithmetic, inlining, …
- Run the program for a while and extrapolate its completion time
- How long was the extrapolated completion time?

Some Hints

- ~230 billion pairs of customers to compare
  - average 209 movies per customer
  - ~48.2 trillion movie comparisons
- 1 day = 86,400 seconds (~100K)
- walk clock time
- on the background with medium priority
- How many days?
- Probably making your head hurt

Estimated Completion 1

- ~109 days
- ~41 microseconds per customer pair

Superset Implementation 2

- Text files were preprocessed into binary files
- Read from the binary files when needed
- Run the program for a while and extrapolate its completion time

Estimated Completion 2

- ~9 days (> 10x faster)
- ~3.4 microseconds per customer pair

Superset Implementation 3

- One binary file:
  - all movie IDs in customer order (~.2 GB)
  - index to the offset for each customer (~2 MB)
- Read from the file and store in memory
- Basically most of the data are memory resident
- Run the program for a while and extrapolate its completion time

Estimated Completion 3

- ~4 days (~2x faster)
- ~1.5 microseconds per customer pair

Just storing the data

Data Structure	Size in GB
2D Boolean array	~8
Array of bit vectors	~1
Array of linked lists	~0.6
Array + Offset	~0.2

Revisit customers without supersets

- Find customers *Y* that intersect (overlap) *X*
- If intersection(*X*, *Y*) = *X*
  - *Y* is a superset of *X*
- If no supersets
  - Find *Y* such that intersection(*X*, *Y*) is a subset of *X*
  - Overlap is less than 100% of *X*

Intersection Algorithm: 3 cases

1. intersection(*X*, *Y*) = *X* [superset]
2. intersection(*X*, *Y*) is a subset of *X* [subset]
3. intersection(*X*, *Y*) is empty

Intersection example

	m1	m2	m3	m4	m5	m6	m7	m8	m9
c1	?		3		3		4		
c2	4		3		3		4		1
c3	4		3		3			3	
c4	3		3						4
c5			3		3	4		3	3

- *X* is c1
- How to compute/compare similarity(*X*, *Y*) if the intersections are of different sizes?
- c2, c3, c4 all have the same RMSE=0

Intersection example

	m1	m2	m3	m4	m5	m6	m7	m8	m9
c1	?		3		3		4		
c2	4		3		3		4		1
c3	4		3		3			3	
c4	3		3						4
c5			3		3	4		3	3

- *X* is c1
- How to compute/compare similarity(*X*, *Y*) if the intersections are of different sizes?
- c2, c3, c4 all have the same RMSE=0
  - are they all equally similar to c1?

Distance

- Two factors
  - RMSE
  - %NotRated
- Distance = RMSE + %NotRated
  - RMSE is 4 times more important because max is 4.
- Distance = RMSE/4 + %NotRated
  - If we want them to be equally important/weighted

Distance

- Error is automatically 4 (MaxDist)
  - When there is no rating

Missing Rating

- Replace it with:
  - 0 [MaxDist is 5]
    - Good:
    - Bad:

Missing Rating

- Replace it with:
  - 0
    - Good: more zeros, more error
    - Bad: no rating means *Y* "hates" the movie?
  - 3
    - Neutral value
    - No rating means *Y* is neutral on the movie

Missing Rating

- Replace it with:
  - 0
    - Good: more zeros, more error
    - Bad: no rating means *Y* "hates" the movie?
  - 3
    - Neutral value
    - No rating means *Y* is neutral on the movie
  - a predicted value
    - global averages weighted by standard deviation

Intersection Algorithm: 3 cases

1. intersection(*X*, *Y*) = *X* [superset]
   - weighted average of supersets
2. intersection(*X*, *Y*) is a subset of *X* [subset]
   - weighted average of subsets
3. intersection(*X*, *Y*) is empty
   - global averages weighted with standard deviation

Summary of Intersection Algorithm

- If *X* has supersets, use supersets only
- If *X* does not have supersets, but has subsets, use subsets
- If *X* does not have supersets nor subsets, use naïve algorithm.

k-Nearest Neighbor Algorithm

- Distance/Similarity for any pair of customers
- Find the top *k* most similar customers (nearest neighbors)
  - Weighted by similarity
- Customers with no supersets or subsets do not have neighbors—use naïve alg
- One issue is how to determine *k*

Summary

- Problem 1
  - Ranking Ad’s on Search Engines
- Problem 2
  - Product Recommendation