Collaborative Filtering, Off-Policy Evaluation, and Marketing Mix Models

MKTG 6279

Week 6

Today

- Collaborative Filtering
- Off Policy Evaluation
- Marketing Mix Modeling

Collaborative Filtering

Collaborative Filtering

What do one person's choices say about another?

- Amazon: "people who buy this book also bought. . . "

These types of tasks are referred to as **collaborative filtering**

- *collaborative* means two or more people working together. Thus, *filter* your choices based on others.

It's a big field with many tools

- logistic regression of each product onto all other choices
- principal components analysis to find underlying taste factors

Goals of collaborative filtering

Improve conversion rates by helping customers find products faster

- Amazon has thousands of products so getting customers oriented quickly is important

Promote **cross-selling**

- Give you ideas on what products to buy in the same transaction or in the near future

Improve **loyalty** by creating a valued-added relationship

- Netflix *knows* me

How it works

Input: data from many users for many items

Goal: fill in ratings in missing user/item combinations

.	Movie 1	Movie 2	Movie 3	Movie 4
John	5	4	?	1
Ringo	5	5	2	2
Paul	1	?	3	4
George	3	2	?	3

The matrix can contain be either the **rating** or simply an indicator for whether it was **consumed/purchased**

Two types of collaborative filtering algorithms

1) Memory-Based

- Use the whole database (or at least a large sample) to create recommendations
- The most prominent algorithm is “user-based” CF
- Similar customers will rate items similarly
- Doesn’t scale well

2) Model-Based

- Use the database to learn a more compact model (e.g., cluster users into segments) that is then used to create recommendations
- A prominent algorithm here is “item-based” CF
- E[preference|user] is modeled explicitly

Measuring similarity between users

Compare users i and j on item n

Pearson correlation coefficient:

$$s_{i,j} = \frac{\sum_n (r_{in} - \bar{r}_i)(r_{jn} - \bar{r}_j)}{\sqrt{\sum_n (r_{in} - \bar{r}_i)^2}\sqrt{(r_{jn} - \bar{r}_j)^2}}$$

Cosine similarity:

$$s_{i,j} = \frac{\sum_n r_{it} r_{jn}}{\sqrt{\sum_n r_{in}^2}\sqrt{\sum_n r_{jn}^2}}$$

Note: only use the items that were rated by both users to find their similarity.

User-based CF: find similar users, predict the missing rating

Use the average rating from the “most similar” customers as a best guess of what the target user would rate that item.

Stylized example: 5 users (rows) and 4 movies (columns).

```
rating
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    5    4   NA    1
## [2,]    4    5    1    2
## [3,]    1    1    3    4
## [4,]    3    2    4    3
## [5,]    3    5    1    2
```

For the first user, what is the expected rating on the third movie?

Find the similarity

```r
#the target user cannot be compared on item 3
#consider only items 1, 2, and 4
simMat  = rating[,-3]
simVec1 = rep(NA,4) #sim b/t user 1 and other 4
for(i in 2:5){
 simVec1[i-1] =
  sum((simMat[1,] - mean(simMat[1,]))*
        (simMat[i,]-mean(simMat[i,])))/
  (sqrt(sum((simMat[1,] - mean(simMat[1,]))^2))*
  sqrt(sum((simMat[i,] - mean(simMat[i,]))^2)))
}
simVec1
```

```
## [1]  0.8386279 -0.9707253 -0.2773501  0.5765567
```

Looks like the second and fifth users are most similar, so average their rating as a best guess.

User-based vs. item-based CF

User-based

- We just saw this: users prefer items that other similar users prefer
- Memory-based algorithm
- Users with similar preferences will rate similarly
- All user-user similarities need to be stored

Item-based

- Model-based
- Takes similarities between item's consumption history
- Compute the similarity between items, and select the most similar items
- More scalable, since only the k most similar items have to be stored in an $n \times k$ matrix

Item-based CF Example

Users prefer items that are similar to other items they like.

For the first user, is the third movie similar to the other ones they like?

If yes, recommend it.

```r
rating
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    5    4   NA    1
## [2,]    4    5    1    2
## [3,]    1    1    3    4
## [4,]    3    2    4    3
## [5,]    3    5    1    2
```

Item-Item correlation matrix

Should we recommend the third movie to the first user?

```
##       [,1]  [,2]  [,3]  [,4]
## [1,]  1.00  0.80 -0.49 -0.90
## [2,]  0.80  1.00 -0.89 -0.97
## [3,] -0.49 -0.89  1.00  0.75
## [4,] -0.90 -0.97  0.75  1.00
```

The ratings from the third movie are highly correlated with ratings from the fourth movie from other users (.75) and negatively correlated with ratings from first and second movie (-.49 and -.89).

The first user hated the fourth movie and loved the first and second movie. So we probably should not recommend the third movie.

Item-based CF is more scalable

Rather than calculating the similarity between the target user all other users, item-based CF compares the similarity between items alone (which is usually a lot smaller).

For Amazon (with thousands of products), they can reduce this even more: rather than store an $n \times n$ item matrix, just record the k most similar items against each target item.

- For example: no need to see how similar a specific toaster is to, say, a dog leash. Ratings for the specific toaster you are looking at tend to be positively correlated with a few other toasters (that we can recommend) and negatively correlated with other toasters (that we won't recommend).

Evaluating recommender systems

Typically, randomly partition the rating matrix into a training and test set.

In the test set, calculate the mean average error (average error between actual rating and predicted rating) or root mean square error.

See `recommenderlab` vignette for more details.

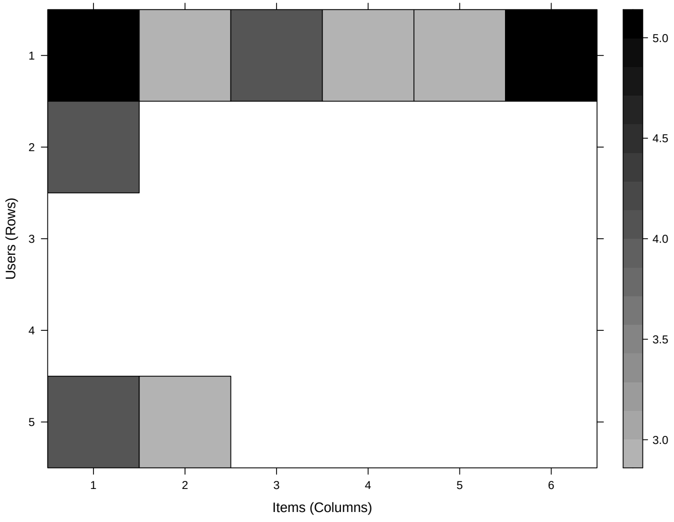
Example: MovieLense dataset

100,000 ratings (1-5) from 943 users on 1664 movies

```r
library(recommenderlab)
data("MovieLense")
```

Ratings from first 5 users on first 6 movies

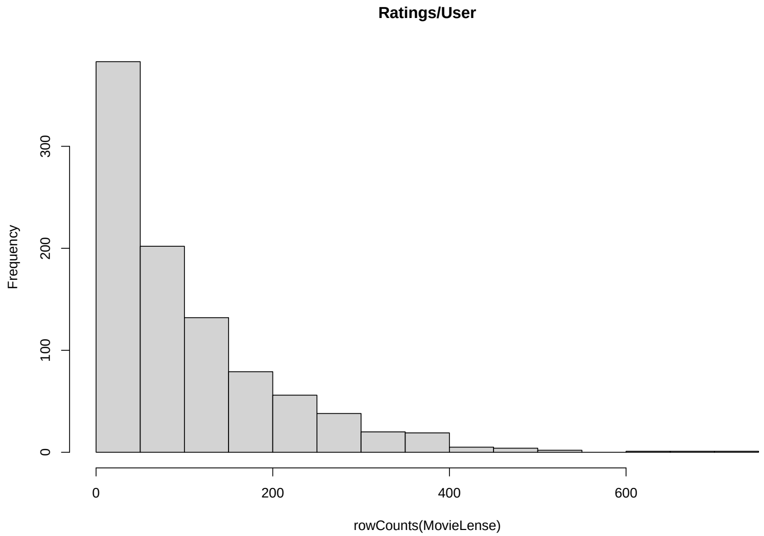
```r
image(MovieLense[1:5,1:6])
```



Dimensions: 5 x 6

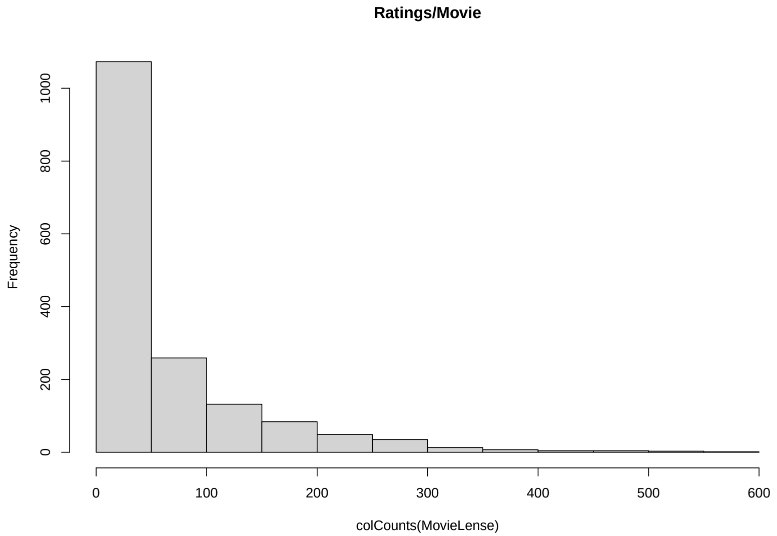
Number of ratings per **user**

```r
hist(rowCounts(MovieLense),main = "Ratings/User")
```



Number of ratings per **movie**

```
hist(colCounts(MovieLense),main = "Ratings/Movie")
```



Estimate the recommender

Each row contains ratings for a single user.

To remove rating bias the algorithm subtracts the mean rating from each row.

```r
rec  = Recommender(MovieLense,method = "UBCF")
names(getModel(rec))
```

```
##  [1] "description"          "data"                 "meth
##  [4] "nn"                   "sample"               "weig
##  [7] "normalize"            "min_matching_items"   "min_
## [10] "verbose"
```

Use the model

```r
#top 5 movies for first user
i1top5 = predict(rec,MovieLense[1,],n=5)
as(i1top5,"list")[[1]]
```

```
## [1] "Boot, Das (1981)"                 "Matilda (1996)"
## [3] "Winter Guest, The (1997)"         "She’s the One (1
## [5] "Manchurian Candidate, The (1962)"
```

```r
#This user has not rated any of these movies
MovieLense[1,match(as(i1top5,"list")[[1]],
                   colnames(MovieLense))]
```

```
## 1 x 5 rating matrix of class ’realRatingMatrix’ with 0 r
```

Model-Based Collaborative Filtering

These models don't rely on similarity alone, they learn patterns from user-item interactions

Matrix Factorization

- Creates latent factors from the user-item matrix
- These latent factors can be labeled based on the grouped items: "romantic comedy lover" or "thriller junkie"
- Spotify uncovers hidden listener preferences (e.g., preferred genres, moods, tempos), enhancing personalized playlists

Neural Collaborative Filtering

- Uses deep learning to find non-linear patterns in behavior
- Learns subtle things like binge-watching habits, genre fatigue, or context shifts (e.g., weekday vs. weekend viewing)
- Instagram and TikTok dynamically personalize user feeds, significantly increasing user engagement

Netflix

Netflix leverages deep learning to implement advanced collaborative filtering

- Detect nonlinear viewing patterns (like genre fatigue or late-night binge sessions)
- Learn from subtle signals – pause/play behavior, time of day, skip rates
- Dynamically adapt recommendations as you interact with content

Off-Policy Evaluation for Recommendation Systems

Context

Recommendation systems aim for continuous improvement

How do we validate a new strategy (e.g., personalized ranking) *before* deploying it to all users?

Direct A/B testing can be expensive, time-consuming, and may harm user experience if the new strategy is poor

Enter Off-Policy Evaluation

- Let's say you have historical data from a recommendation system that showed some products (or movies, or ads, or anything) to users
- But you've developed a new algorithm
- Off-policy evaluation (OPE) lets you estimate how good that new policy would have been using only past data (i.e., without actually running it live)

How it works

You have logged data from an **old** policy

- What was shown to users

You want to evaluate a **new** policy

- A better algorithm

But this data is biased toward actions the old policy took

Solution: re-weight observations based on the relative likelihood of actions between policies

Use importance sampling

Re-weight the logged data to simulate *what would have happened* under a new system

- What is the probability of that recommendation under the new policy?

If the new policy had a relatively greater probability of targeting this observation, give it more weight

$$W_i = \frac{\pi_{\text{new}}(a_i|X_i)}{\pi_{\text{old}}(a_i|X_i)}$$

Where $\pi(a_i|X_i)$ is the probability of taking action a given state X

Note: we've already seen this for matching treatment and control groups

Intuition

If the old policy was likely to have *the same* recommendation as the new policy, the observation gets low weight

- If the recommendations are exactly the same between policies, it is hard to estimate the added value
- "This probably would have happened anyway, no new information"

If the old policy was likely to have *a different* recommendation as the new policy, the observation gets more weight

- "This gives us a rare look at what might happen under the new system, so it's valuable information."

These weights "amplify" the signal from rare but relevant data points

Um, still confused

But wait, why overweight more “likely” data points?

Relative to the old policy, these recommendations are *more* likely, so given them more weight

Since it was unlikely from the old policy, it “deviated” from the biased old policy

These are rare, but relevant, occurrences in the data that should help us evaluate the new policy

It is similar to how we add more weight to observations to try and match treatment and control group data (from week 2)

Example output

User	Product	Pr (Rec Old)	Pr(Rec New)	W
1	Headphones	0.6	0.8	1.3
2	Smartwatch	0.5	0.5	1.0
3	Running Shoes	0.7	0.2	0.3

Using the "importance" weights we can get a good idea of how often people would click on recommendations if we were using the new policy.

This helps the online store decide if the new system is worth the risk of fully launching.

OPE at Amazon I of II

Problem 1 Off-policy evaluation (OPE) methods struggle with a large number of actions or when certain actions are rarely taken. Existing methods, like those based on inverse-propensity scoring (IPS), can have high variance in these situations.

Problem 2 Prior work has proposed *marginalized* IPS (MIPS) that uses *action embeddings* to reduce the dimensionality. But setting these embeddings is a manual process.

- Rather than 100 discrete actions, represent this in a smaller continuous space (say of dimension 10)

OPE at Amazon II of II

Solution *Learn* the action embeddings from logged data to address the limitations of MIPS (marginalized IPS).

How? Minimize MSE from a linear regression of reward estimates and actual rewards as a function of large action space. Then proceed as usual.

Marketing Mix Models

What are MMMs?

Estimates how different marketing channels (TV, digital, email, etc.) contribute to business outcomes (sales, installations, etc.)

Use cases

- Need to quantify the ROI of each marketing channel
- Want to forecast future sales under different budget scenarios
- Don't have access to user-level data (e.g., due to cookie restrictions)
- Need a holistic view of all internal and external factors impacting performance
- Used to optimize spend across channels

Companies using MMMs I of II

Coca-Cola Leverages MMM to fine-tune its global media strategy across 200+ markets. Helps decide how much to invest in TV vs. digital vs. local sponsorships.

Meta Built and open-sourced Robyn, a Bayesian MMM tool to help advertisers measure multi-channel ROI in a privacy-first world (no cookies, no problem).

Target Uses MMM to balance online vs. offline media spend. It helps them optimize across paid search, display, loyalty programs, and circular ads.

Companies using MMMs II of II

Netflix While not traditional MMM, Netflix models media mix effects on subscriptions — like how trailers, social buzz, and email campaigns drive new user signups.

Google Launched open-source Meridian, a hierarchical Bayesian MMM.

And many more. . .

Under the hood

Surprise. . . it's regression!

Basically a multiple linear regression:

$$y_{gt} = \beta_0 + \beta_1 \text{TV}_{gt} + \beta_2 \text{Search}_{gt} + \ldots + \varepsilon_{gt}$$

Where y might be sales in geography g in time t

β tells us the contribution of each marketing component

A few common twists

- Seasonality: via time-varying intercepts, holiday indicators, etc.
- Reach: how many people saw the ad
- Frequency: how *often* they saw the ad during the time period
- Other variables: price, product launches, organic search
- Adstock: decaying effects of ads
- Saturation: diminishing returns

Adstock

Allows for ad effects to persist over time

$$\text{Adstock}(x_t, x_{t-1}, \ldots, x_{t-L}; \alpha) = \frac{\sum_{s=0}^{L} \alpha^s x_{t-s}}{\sum_{s=0}^{L} \alpha^s}$$

$x_s \geq 0$ the spend at time period t

$\alpha \in [0, 1]$ is the geometric decay rate (can be fixed or estimated, lower means faster decay)

L is the maximum number of lags

The denominator acts as a normalizer, so that it is a weighted average of past spend

- Sometimes this won't be here

Adstock in R

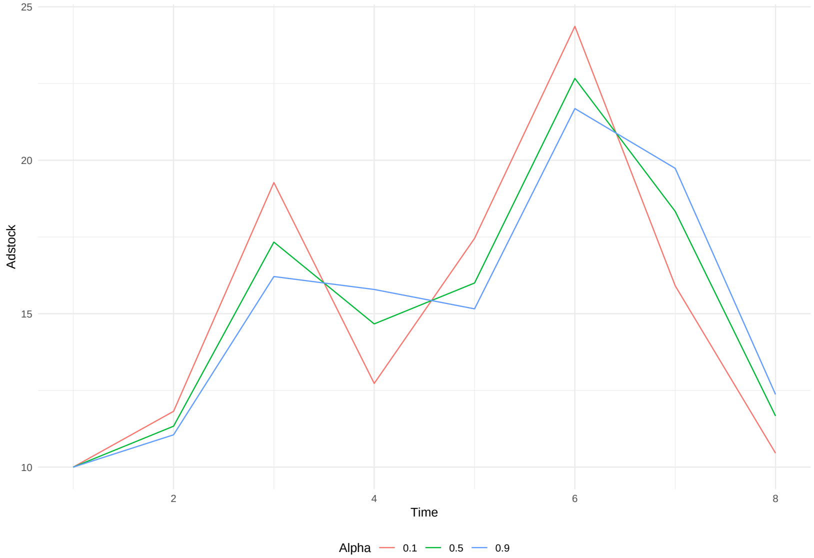
```r
spend = c(10,12,20,12,18,25,15,10)

get_adstock = function(x,alpha = .5,L = 1) {
  n = length(x)
  adstock    = numeric(n)
  adstock[1] = spend[1]
  a = alpha^(0:L)

  for (t in 2:n) {
    w  = a[1:min(t,L+1)]
    adstock[t] = sum(w*x[t:max(t-L,1)])/sum(w)
  }

  out = data.frame(time=1:n,alpha=alpha,adstock)
  return(out)
}
```

Adstock visual



Hill Function

Allows for saturation effects

$$\text{Hill}(x; ec, \text{slope}) = \frac{1}{1 + (\frac{x}{ec})^{-\text{slope}}}$$

$x \geq 0$ is the spend

$ec > 0$ is the half-saturation point

slope controls the shape

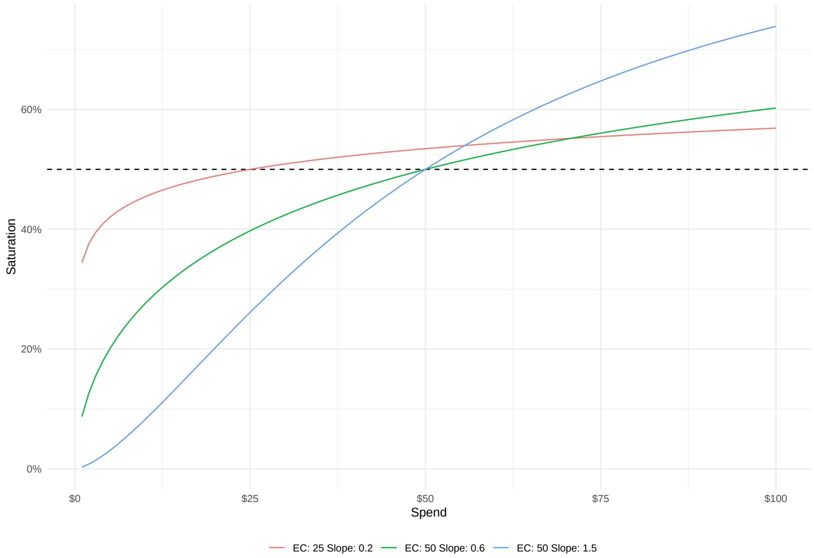
- ≤ 1 is concave
- > 1 is S-shaped, convex for $x < ec$ and concave for $x > ec$

Hill Function in R

```r
hill  = function(x, ec, slope) {
  data.frame(spend=x,ec,slope,
             hill=1/(1+(x/ec)^(-slope)))
}
spend = 1:100

hill_gg = bind_rows(
    hill(spend,25,.2),
    hill(spend,50,.6),
    hill(spend,50,1.5)) %>%
mutate(label = paste0('EC: ',ec,' Slope: ',slope))
```

Hill Function visual



MMM Final Thoughts

Marketing Mix Modeling has seen a resurgence in popularity

It's really just a regression applied to a very specific problem – understanding the contribution of marketing actions on business outcomes

Companies typically run multiple marketing campaigns across multiple channels simultaneously, so attributing the effects to any one action is challenging

When implementing, be especially mindful of the following:

- Seasonality
- Adstock
- Saturation

Course Recap

1) A/B Testing and causal forests
2) Variance reduction, PSM, AIPW, and Double ML
3) Segmentation and cluster analysis
4) Elastic Net and XGBoost (sort of)
5) Multi-Armed Bandits
6) Collaborative filtering, OPE, and MMMs

Final tips

- Focus on **actionable** insights
- Code it up to understand it – you have to engage with it to get it
- Lean on LLMs to start building the *foundation* of knowledge, but don't let it *define* your knowledge
- Companies value those who can synthesize complex ideas into actionable outcomes, so focus on results and impact more than the methods