Recommender Systems

Goals

You will…
… think about Recommender Systems, understand what they are, and ideally think they are pretty cool :)
… see some different ways of approaching the task and some considerations that go into framing the task
… be able to have a fruitful conversation about building a recommender system as (part of) a product
… be able to explain collaborative filtering
… be able to explain matrix factorization, and discuss its relationship to collaborative filtering
… be able to contextualize algorithms you've already learned (typical supervised and unsupervised machine learning) as being potentially valuable parts of recommender products

Goals: you will…
… think about Recommender Systems, understand what they are, and ideally think they are pretty cool :)
… see some different ways of approaching the task and some considerations that go into framing the task
… be able to have a fruitful conversation about building a recommender system as (part of) a product
… be able to explain collaborative filtering
… be able to explain matrix factorization, and discuss its relationship to collaborative filtering
… be able to contextualize algorithms you've already learned (typical supervised and unsupervised machine learning) as being potentially valuable parts of recommender products

What are Recommender Systems?

Ask audience: can you name any recommender systems?

What are Recommender Systems?

- Goal: Offer a user something they might be interested in
  - What is “interest”?
    - Binary yes/no
      - Here’s a song you might like
    - Ranked list
      - Here are suggested bookings
    - Collection of items
      - You might want a memory card with that camera
  - What can we recommend?
    - person, place, thing, experience, company, product, phrase
    - dating, travel, shopping, media, HR

Examples:
OkCupid/Coffee Meets Bagel, TripAdvisor/Spot, Amazon, Pandora, Netflix, Goodreads, Google Search

Inspiration:
Magical when done right. My own experience with Pandora. Enjoyment, external discovery, self discovery, social interactions…

Lots of factors. Understanding **constraints** and **framing** correctly are important for success.

Why this track?

Black Sheep by Metric

Based on what you've told us so far, we're playing this track because it features electric rock instrumentation, electronica influences, a subtle use of vocal harmony, major key tonality and electric pianos.

What are Recommender Systems?

- Goal: Offer something the user cares about
  - another person, another place, another thing, another experience
  - dating, travel, shopping, media
- Field: Blend of AI, HCI, CogPsy
  - personalization, serendipity, motivation, trust, confidence

Amazon 'Reviewing' Its Website After It Suggested Bomb-Making Items

By AMIE TSANG September 20, 2017

A British TV station said Amazon's website was prompting customers to buy all the ingredients to make a bomb.
Richard Drew/Associated Press

LONDON — Amazon said on Wednesday that it was reviewing its website after a British television report said the online retail giant's algorithms were automatically suggesting

What are Recommender Systems?

- Field: Blend of CS, AI, HCI, CogPsy
  - Serendipity
  - Personalization
  - Diversity
  - Persistence
  - Modality
  - Privacy
  - Motivation
  - Trust
  - Confidence

Serendipity: Unexpected, wouldn't have otherwise seen. Harry Potter 7 given 6 is meh. Echo chamber. (Amazon programming book recommendations.) How surprising are the recommendations? Did you recommend something the user wouldn't have otherwise seen? (If you ask your best friend for a single book recommendation of a really good book and they say Harry Potter, they maybe kinda blew it (because you already knew that))
Personalization: Populist versus personalized. Popular can seem dumb; too personal can be creepy. And popular doesn't always work: eg, dating. If you and I are sitting next to each other and each open the app, do we get different recommendations? If not, is it really "smart"??
Diversity: Did you recommend Harry Potter 1, 2, 3, …? "Silver Bullet" vs Shotgun. But also want basket approach (beer with those diapers?). Breadth vs Depth. Breadth *and* Depth?
Persistence: Have you recommended this item before? Did they rate it? Pandora plays a song several times. Did you not rate it because you were so into your music? Or because you had stopped listening? Does it matter if you rated something immediately before and after (perhaps implying that you had your phone on you and were in rating mode)?
Modality: I usually want Sweet Tomatoes, but tonight I'm in a different modality because it's date night and my wife doesn't like Sweet Tomatoes
Privacy: You're generally pretty similar to Moses, and he just bought furry pink handcuffs, do you want some?
Motivation: Pure best match? (Medical Treatment) Overall product value? (Pandora) Specific sale/margin? (Amazon) Upsell? (Wells Fargo)
Trust: What is the motivation? What data is offered, used? How is it being used? Why am I seeing what I'm seeing?
Confidence: How good are the results? How understandable are the mistakes? Harry Potter 7 is an understandable mistake, 50 Hairdos for Terrible People might not be. Confidence over time, especially: building trust first and then branching out.

Lots of factors. Understanding **constraints** and **framing** correctly are important for success.

Frequently bought together

Buy Both: $21.90

Kinds of Data

- User Data
  - The recommendee/active user
  - Are they middle-aged? What's their income? How often do they log in? Where do they live?
- Item Data
  - Is it expensive? How many megapixels? Does it feature Nicholas Cage?
- **Interaction Data**
  - This is about the relationship between users and items
  - Did many users buy/star/like/save/wishlist/rate this item? What was the typical rating of the item?
  - Did this user buy/star/etc many items? What was the user's typical rating behavior?

Encoding Information

- Challenge
  - What does a “like” mean? How about a “view”? How about 10 views? How about a “saved but kept not buying it”? (Did they think it was funny? Are they actually using ‘like’ as a ‘bookmark’ functionality?)
- Framework: Explicit v Implicit
  - Did the user take a clear and explicit action?
    - Thumbs up/down on Pandora
    - Did the user, UI/UX designer, and data scientist all interpret this action to mean the same thing?
  - Did the user take some actions that might indicate something, but which weren’t explicit?
    - Did they view a product? Did they share it? Did they spend a long time on the page? (Was their mouse active while they were on the page?)
- Incentive pollution
  - Did you ask your friend to test out your app? How honest will their feedback be?

Preview: User Value

- When do the recommendations become useful to the end user?
  - Amount of Data: some algorithms…
    - rely on a single user rating many items
    - rely on many users rating many items, collaboratively
    - require lots of user information or item information
    - have a lower or higher ceiling of learning (bias/variance)
  - Time: some algorithms…
    - are easy to update/recalculate, others are very hard
    - might be updated each time user data changes, item data changes, or interaction data changes
  - Company Size
    - the approach used when your company/data/budget is small isn't necessarily what you'll use later

Four Common Approaches

- So, there are many ways to frame the challenge
  - Different data requirements, runtime requirements, types of success
  - Lots of moving parts; different ‘core’ algorithms at the heart of various approaches
- Let’s look at some common frameworks

Classification/Regression

Classification/Regression

- Classifier to predict interaction/rating, using past interactions/ratings as training data
  - Using features of the people I've liked so far on <dating site> as training data, predict/rank who I'll like
  - Using features of the people who have bought product X, predict/rank who else is likely to buy it

Classification/Regression

6’2	Blond	5 o’clock shadow	Y
5’11	Brown	Clean-shaven	N
6’01	Black	Gandalf	N
6’0	Blond	Moustache	**?**

Classification/Regression

- Advantages
  - Familiar tools/algorithms
  - Well studied success metrics (for one classifier)
    - Diagnose bias/variance, swap algorithms, add features
  - Well studied confidence metrics
  - Potential for very personalized results, learning overall trends
  - Potential for good explanations (confidence&trust)
  - Single-user case is OK
  - Information privacy maintained

Classification/Regression

- Limitations
  - Training data required: interactions/ratings (Y)
  - Feature engineering required: meaningful descriptive data (X)
  - Tuning one classifier is tricky; tuning n classifiers?
    - Devops challenge (training, caching)
    - Data Science challenge (success metric for experiments)
  - Time to first utility (from User's POV): interactions, compute time
  - Learning overall trends may be an oversimplification, compressing modalities
    - Complex decision boundaries can be learned w enough data, still a UX issue
  - Rating/Interaction gathering&interpretation creates product restrictions
  - Solipsistic (not inherently social)

Collaborative Filtering

What if I told you…
…you didn't have to do any feature engineering? On either the user or the item?!
…that every single rating from one user helped every other user's recommendations?

Collaborative Filtering often considered "the" recommendation algorithm (but there are many). It's a little old-school relative to MF (later), but classic.

Collaborative Filtering

	A	B	C
Bobbie	0	1	1
Carly	1	0	0
Danielle	1	1	1
Edgar	0	1	**?**

Collaborative Filtering

- Use similarity to other users/products to predict interaction/rating, using past interactions/ratings as training data
  - To the degree to which I'm similar to Bobbie, I may feel like she did about this item. Likewise for Carly, Danielle, Edgar…
  - I'm similar to Bobbie to the degree to which we've agreed on other items

Collaborative Filtering

$$pred(p_i, t_i) = \bar{r}_{p_i} + \frac{\sum_{p_j \in P, i \neq j} sim(p_i, p_j) \times (r_{p_j,t_i} - \bar{r}_{p_j})}{\sum_{p_j \in P, i \neq j} sim(p_i, p_j)}$$

Piece by piece:

Start with my average rating: am I crotchety?

Perturb that by some amount for each user:

- What was the other user's average rating? Are *they* crotchety?
- How did they rate this? Above or below their average?
- To the degree to which I'm similar to them, let's update my predicted rating based on theirs

Collaborative Filtering

- Variations
  - Neighborhoods
    - Similarity amongst n most similar people/items
    - Faster (can also cache similarity)
  - Flip the matrix: items “choose” people rather than vice versa
  - Use a different similarity function: “content-boosted” collaborative filtering
    - Dan and Bob are similar because they have similar Facebook profiles, rather than because they rated things similarly

Collaborative Filtering

- Advantages
  - Content agnostic
    - Feature engineering not required
  - Inherently social
    - Potential for social explanations
    - Popularity naturally built-in, in a way
  - One rating helps many people (“collaborative”)
    - Time to first utility (interactions/ratings)

Collaborative Filtering

- Limitations
  - Content agnostic
    - May not take advantage of all information known about people/ items
  - Inherently social
    - Potentially violates user privacy expectations
    - Less-rated areas can lead to lower-quality recommendations
  - Sparsity … it's really tough
    - Low ratings or overlaps (similarity) can lead to relatively little information going into each prediction
    - High person/item churn can lead to low overlaps
    - Single-user case is a fail case

Collaborative filtering is kinda bimodal: brilliant, or terrible.

Similarity

Similarity

- Go all-in on similarity: do a really good job of figuring out which things are similar to one another
  - Use that similarity to make connections: because I liked A and A is similar to B (in reduced dimensionality subspace), I may like B
  - Often we want to “smooth out” similarity space, and make distance more meaningful (anti-Curse of Dimensionality): Dimensionality Reduction
    - Loss of information, nuances may be lost
    - More feature ‘overlaps’, more ways to compare
    - Depending on alg, can be very explainable (eg LDA topic modeling is relatively easy to interpret)
  - Runtime: run a potentially-computationally-expensive offline dimensionality reduction (eg, LDA/topic modeling/NMF), then index similarity in reduced-dimensionality subspace for fast queries

Similarity

- Advantages
  - User information requirements are very low
    - Interactions/ratings not needed to precompute similarities
    - Time to first utility is 1 rating
  - Potential for high interpretability of similarities and recommendations
  - Modality preserved*
  - Runtime requirements are low
    - Heavy computation can be done offline
  - Single-user use case OK
  - Potentially useful side-effects: similarities, topics, topic distributions

Similarity

- Limitations
  - Item information requirements fairly high
    - Feature engineering required: meaningful descriptive data about items
  - Information about person not taken into account
  - High compute requirements for dimensionality reduction, indexing
  - Can be difficult to evaluate quality of dimensionality reduction
  - Solipsistic (not inherently social)
  - Overall trends ignored; pointwise recommendations only

Matrix Factorization

Matrix Factorization

- What if there are actually a relatively small set of features of items and users that determine the reaction of most users to most items?
  - We might imagine explicit features, but would they have enough coverage?
  - What if they were latent features we could discover?
- Similar to Collaborative Filtering, but rather than neighborhood-scale, global-scale
  - Collaborative Filtering used latent features at local scale

Challenges of Collaborative Filtering

Item-Item I like action movies → rate *Top Gun* and *Mission Impossible* 5s.
→ I'm recommended *Jerry Maguire* even though I won't like it.

User-User I like Tom Cruise → rate *Top Gun* and *Mission Impossible* 5s.
→ I'm recommended *Transformers* even though I won't like it.

What about our friend Linear Regression?

Action, Romance, Comedy, etc.

Tom Criuse, Tom Hanks, Megan Fox, etc.

Long, Short, Subtitles, Foreign, Happy, Sad, etc.

$$\begin{aligned} \textit{Rating Prediction} = \beta_0 &+ \beta_1 \times \textit{actionness} + ... \\ &+ \beta_i \times \textit{foxiness} + ... \\ &+ \beta_j \times \textit{sadness} + \epsilon \end{aligned}$$

… “actionness”?

Matrix Factorization will, roughly, try to find these dimensions

If we treated the latent features as explicit features, we could combine them similarly to what we do for LR.

But how would we get those latent features?

Matrix Factorization attempts to find these features.

The factorization models can be interpreted as a linear combination of bases.

Matrix Factorization Setup

Items

Users

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \\ u_{31} & u_{32} \\ u_{41} & u_{42} \\ u_{51} & u_{52} \end{bmatrix} \times \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \\ v_{21} & v_{22} & v_{23} & v_{24} & v_{25} \end{bmatrix}$$

Figure 9.9: UV-decomposition of matrix M

M U V

We have M. Let's find U and V!

Example from Mining Massive Datasets: http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

Goal: Create a factorization of a sparse data matrix M into U • V such that the reconstruction of M (obtained by U • V) serves as a model

Steps:

Start with bad U and V that don't reconstruct M well at all

Iterate across U and V (different ways to iterate) until U • V reconstruct M well enough (different ways to stop).

… this looks like SVD.
Can we just do SVD?

Items

Users

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \\ u_{31} & u_{32} \\ u_{41} & u_{42} \\ u_{51} & u_{52} \end{bmatrix} \times \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \\ v_{21} & v_{22} & v_{23} & v_{24} & v_{25} \end{bmatrix}$$

Figure 9.9: UV-decomposition of matrix M

M U V

Example from Mining Massive Datasets: http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

We can't do PCA or SVD because they must be computed on a dense data matrix M.
We *could* get around that by imputing missing values, but with what? The mean? 0? Not super satisfying. But! This is what sklearn does when it says it factorizes sparse matrices. Boo!

So we can't do SVD because of sparsity,
what can we do?
Let's start with a wrong answer, and then
iteratively improve it!

Items

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

Users

Figure 9.10: Matrices U and V with all entries 1

U V P

Example from Mining Massive Datasets: http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

Let's initialize U and V with something simple. 1s will do; we'll look at other ways later.

We've initialized; how wrong are we? RMSE

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix}$$

M

Items

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

Users

Figure 9.10: Matrices U and V with all entries 1

U V P

Example from Mining Massive Datasets: http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

So we initialized with 1s. This means our current reconstruction is all 2s. M (reproduced in the upper right) is our goal. How far off are we?
RMSE
What happens in the third row? We decide not to count the values that aren't there! So the differences are: {0, 1, -1, 2}, and the Sum of Squares is 6
Exercise: what's the total Sum of Squares? (18 + 7 + 6 + 23 + 21 = 75)
Will the RMSE and SSE be minimized at the same point? (yes, so we don't really need to divide and take the square root. For funsies, the RMSE here is sqrt(75/23)=1.806)

Now, how can we improve our reconstruction a bit?

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix}$$

M

Items

$$\begin{bmatrix} x & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} - \begin{bmatrix} x+1 & x+1 & x+1 & x+1 & x+1 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

Users

Figure 9.11: Making u_{11} a variable

U V P

Example from Mining Massive Datasets: http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

Let’s make a small improvement. Looking at the upper left cell…
How much does it contribute to the error term?

Minimize x's contribution to the Sum of Squared Error (and therefore RMSE)

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix}$$

M

Items

$$\begin{bmatrix} x & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} - \begin{bmatrix} x+1 & x+1 & x+1 & x+1 & x+1 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

Users

P

Figure 9.11: Making u_{11} a variable

$$(5-(x+1))^2+(2-(x+1))^2+(4-(x+1))^2+(4-(x+1))^2+(3-(x+1))^2$$

$$(4-x)^2+(1-x)^2+(3-x)^2+(3-x)^2+(2-x)^2$$

Example from Mining Massive Datasets: http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

Let's make a small improvement. Looking at the upper left cell…
How much does it contribute to the error term?
We want the value of x that minimizes the sum, so we take the derivative and set that equal to 0, then solve for x.
Exercise time! :)

Take the derivative, set equal to 0, solve for x

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix}$$

M

Items

$$\begin{bmatrix} x & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} - \begin{bmatrix} x+1 & x+1 & x+1 & x+1 & x+1 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

Users

P

Figure 9.11: Making u_{11} a variable

$$(5-(x+1))^2+(2-(x+1))^2+(4-(x+1))^2+(4-(x+1))^2+(3-(x+1))^2$$
$$(4-x)^2+(1-x)^2+(3-x)^2+(3-x)^2+(2-x)^2$$
$$-2\times\big((4-x)+(1-x)+(3-x)+(3-x)+(2-x)\big)=0$$

Example from Mining Massive Datasets: http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

-2(13-5x = 0)
x = 2.6

We've found a better x, how much did it help us?

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix}$$

M

Items

$$\begin{bmatrix} 2.6 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 3.6 & 3.6 & 3.6 & 3.6 & 3.6 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

Users

P

Figure 9.12: The best value for u_{11} is found to be 2.6

Example from Mining Massive Datasets: http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

How much has the SSE been reduced in the first row?
It's been reduced from 18 to 5.2 in the first row
Total SSE has been reduced from 75 to 62.2

Your turn!
What value for y minimizes the SSE?

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix}$$

M

Items

$$\begin{bmatrix} 2.6 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} y & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2.6y+1 & 3.6 & 3.6 & 3.6 & 3.6 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \end{bmatrix}$$

Users

P

Figure 9.13: v_{11} becomes a variable y

Example from Mining Massive Datasets: http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

Now your turn: can you find out the value for y that minimizes the SSE of the first column?

Take the derivative, set equal to 0, solve for y

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix}$$

M

Items

$$\begin{bmatrix} 2.6 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} y & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2.6y+1 & 3.6 & 3.6 & 3.6 & 3.6 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \end{bmatrix}$$

Users

P

Figure 9.13: v_{11} becomes a variable y

$$(5-(2.6y+1))^2+(3-(y+1))^2+(2-(y+1))^2+(2-(y+1))^2+(4-(y+1))^2$$

$$(4-2.6y)^2+(2-y)^2+(1-y)^2+(1-y)^2+(3-y)^2$$

$$-2\times(2.6(4-2.6y)+(2-y)-(1-y)+(1-y)+(3-y))=0$$

Example from Mining Massive Datasets: http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

y = 17.4/10.76 = 1.617

We've found a better y, how much did it help us?

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix}$$

M

Items

$$\begin{bmatrix} 2.6 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1.617 & 1 & 1 & 1 & 1 \\ 1 & & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 5.204 & 3.6 & 3.6 & 3.6 & 3.6 \\ 2.617 & 2 & 2 & 2 & 2 \\ 2.617 & 2 & 2 & 2 & 2 \\ 2.617 & 2 & 2 & 2 & 2 \\ 2.617 & 2 & 2 & 2 & 2 \end{bmatrix}$$

Users

P

Figure 9.14: Replace y by 1.617

Example from Mining Massive Datasets: http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

Result after y has been improved

What happens when there's missing data?

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix}$$

M

Items

$$\begin{bmatrix} 2.6 & 1 \\ 1 & 1 \\ z & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1.617 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 5.204 & 3.6 & 3.6 & 3.6 & 3.6 \\ 2.617 & 2 & 2 & 2 & 2 \\ 1.617z+1 & z+1 & z-1 & z+1 & z+1 \\ 2.617 & 2 & 2 & 2 & 2 \\ 2.617 & 2 & 2 & 2 & 2 \end{bmatrix}$$

Users

P

Figure 9.15: u_{31} becomes a variable z

$$(2-(1.617z+1))^2+(3-(z+1))^2+(1-(z+1))^2+(4-(z+1))^2$$

$$(1-1.617z)^2+(2-z)^2+(-z)^2+(3-z)^2$$

$$-2\times(1.617(1-1.617z)+(2-z)+(-z)+(3-z))=0$$

Example from Mining Massive Datasets: http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

We're taking this approach because unlike SVD, it can handle missing data. So what happens when there's missing data?
What about when an element of U or V, z in this case, contributes to reconstructing missing data?
Simply don't include the missing cell in the terms contributing to the SSE! Notice how there are only 4 terms included here rather than 5 as before.
z = 6.617/5.615 = 1.178

What about the general case?

1 $$p_{rj} = \sum_{k=1}^{d} u_{rk} v_{kj} = \sum_{k \neq s} u_{rk} v_{kj} + x v_{sj}$$

2 $$(m_{rj} - p_{rj})^2 = \left(m_{rj} - \sum_{k \neq s} u_{rk} v_{kj} - x v_{sj}\right)^2$$

3 $$\sum_{j} \left(m_{rj} - \sum_{k \neq s} u_{rk} v_{kj} - x v_{sj}\right)^2$$

4 $$\sum_{j} -2 v_{sj} \left(m_{rj} - \sum_{k \neq s} u_{rk} v_{kj} - x v_{sj}\right) = 0$$

$$x = \frac{\sum_j v_{sj} \left(m_{rj} - \sum_{k \neq s} u_{rk} v_{kj}\right)}{\sum_j v_{sj}^2} \qquad y = \frac{\sum_i u_{ir} \left(m_{is} - \sum_{k \neq r} u_{ik} v_{ks}\right)}{\sum_i u_{ir}^2}$$

Example from Mining Massive Datasets: http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

1: We care about changing u_rs; we'll replace it with an x so we can focus on it. It only affects the elements in row r of P, so we only care about p_rj
2: If m_rj is a nonblank entry of M, its contribution to the SSE is shown in 2
3: We can write the SSE that are affected by x=u_rs as 3
4: Take the derivative, set equal to 0, and solve for x to minimize SSE and RMSE
We get the formula for x on the left. Analogously, we can solve for v_rs=y in V, shown on the right.

Practicalities

- Preprocessing
  - Subtract from each nonblank element in M the average rating of user i
  - Subtract from each nonblank element in M the average rating of item j
  - Do these in either order, or figure out what you'd subtract from each and subtract the average of that! (Subtract the average of the average rating of user i and item j)
  - Un-normalize predictions afterwards to bring them back to original scale
  - (Or use penalized/regularized user/item baselines)

Example from Mining Massive Datasets: http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

Practicalities

- Initialization
  - Start with each element the same value
    - Ie the average of nonblank elements of M
  - Randomly perturb the values to set different seeds
    - Perturb according to Normal Distribution? Uniform?
- Optimization
  - What order to visit elements of U and V?
    - row-by-row
    - permutation of elements
    - each element should be considered once per round
- Convergence
  - When to stop?
    - If improvement in RMSE drops below threshold
    - If max improvement of an element drops below threshold

Example from Mining Massive Datasets: http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

Practicalities

- Overfitting
  - Use a learning rate and only move partway towards the improved value
  - Stop before convergence/set high threshold
  - Take many UV decompositions, average their results (like random forests!)
  - Build in a penalty (λ) on the magnitudes of U and V, |Ui| and |Vj|
    - Can (should?) also include user and item baselines, b, in the cost function, with their own (or shared) λ

Example from Mining Massive Datasets: http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

Matrix Factorization

- Advantages
  - Decent with sparsity
    - as long as we regularize
  - Prediction is fast, just an inner product
  - Can inspect latent features for topical meaning
    - Can add non-negativity constraint to help interpretability
  - Can be extended to include side information

Matrix Factorization

- Limitations
  - Need to refactorize with new data, very slow
  - Fails in the cold start case
  - Not great open source tools for huge matrices
  - Difficult to tune directly to the type of recommendation you want to make

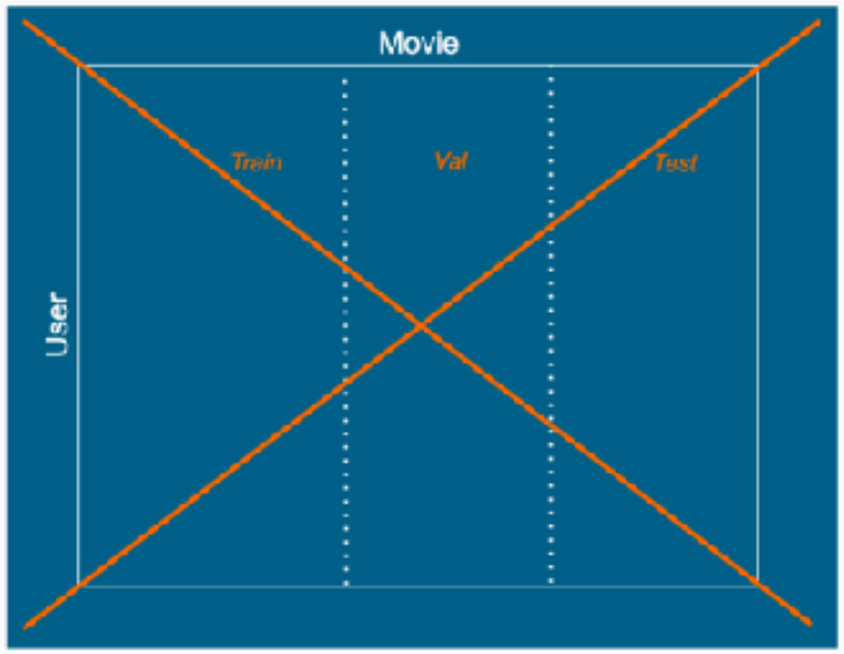
Four Common Approaches

- Classification/Regression
- Collaborative Filtering
- Similarity (Dimensionality Reduction)
- Matrix Factorization

Four Common Approaches

- Recommendation can be framed in many ways
  - These approaches can be stacked & combined & inverted, etc.
    - Reduced dimensions as input to classification?
    - Collaborative Filtering matrix blanks filled in with regression output?
    - Similarity function based on dimensionality reduction?
    - Inversion of person<>item framing?

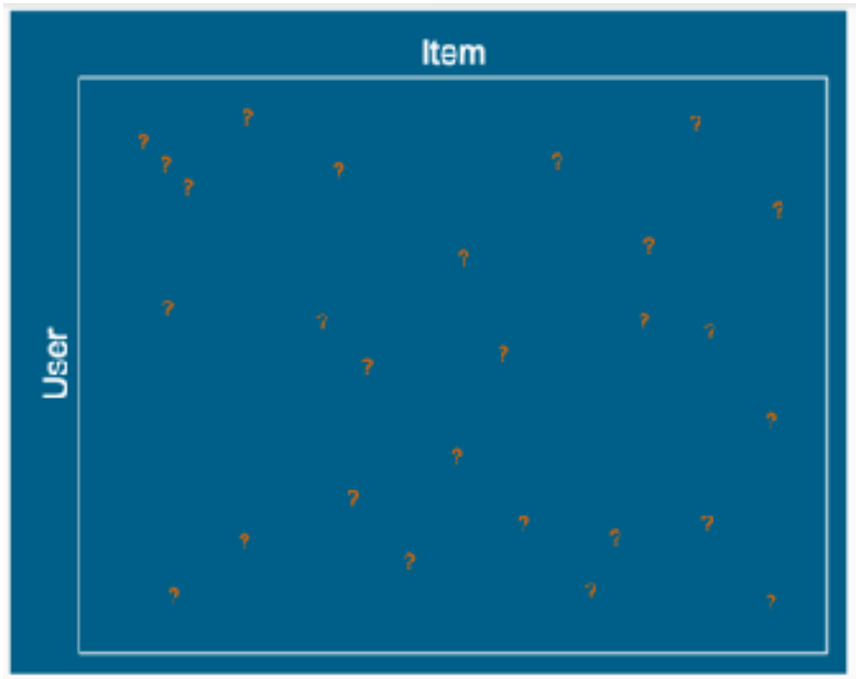
Evaluating Performance



Quick note: The label “Movie” can be seen more generally as “Item”.

Why is this bad? Remember, we’re taking similarities between items (movies, in this case). So if we train on only part of the items, how can we possible make predictions about items we have no data about? They’ll all be seen as equal (no info about any of them when we go to make predictions).

Evaluating Performance

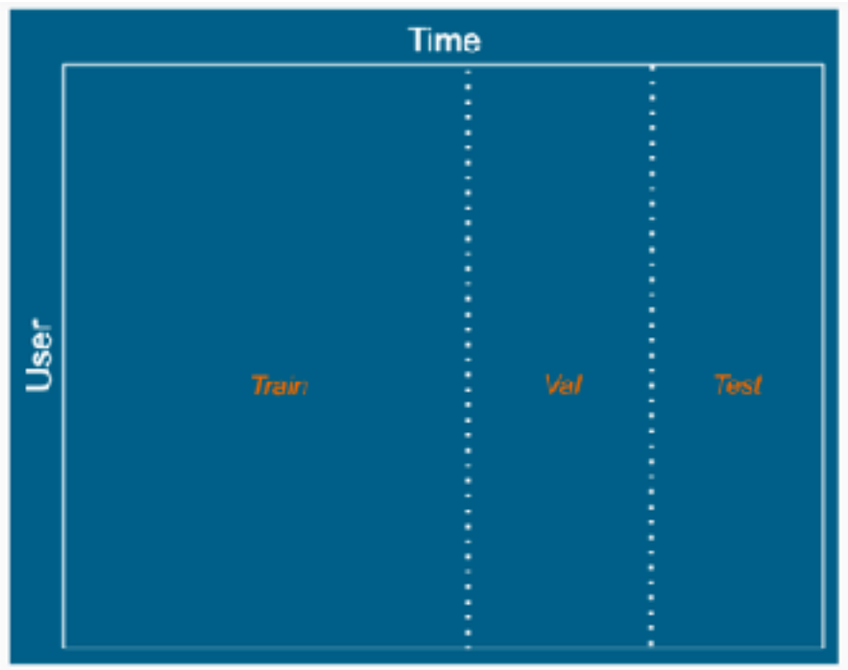


Cross-validation is all about trying to measure our models future performance in the field.

In the field, we're not really making rating predictions, but rather we're trying to recommend the thing the user would want to see next. So, what we're measuring here is different from what our model should be doing well in the field. Two ways to look at this:
1. We want to recommend the thing the user would want to see next, but we don't actually have that information, so we can't properly test our model, really. (E.g. Our predicted "top 10" probably will not have ratings by the user.)
2. We only care that our model recommends things in the right order, and we only care about the ordering of the top-rated items. So, in this cross-validation method, if the target is 2, and we predict 1, that's not really so bad, but we'll still get punished by MSE.

So… ideally we would A/B test our recommenders. But we still need to chose which ones to user as A and B!

Evaluating Performance



This might capture more of how the recommender will perform in the wild, especially if the items have temporal behavior (as “new releases” on netflix probably do, but probably not items for sale on Amazon). The idea is that since we’re making future predictions, we’ll make the holdout set the future. This still has one of the same problems as the previous slide: in the wild, we only care that the top recommendations are good, but we still can’t actually measure that. E.g. Say our model recommends 10 movies… we look in the holdout set (from the future), and the user still hasn’t rated those movies. We CAN’T evaluate those recommendations as either good OR bad.

Additional Considerations

- User state (already owns X, needs Y?)
  - Basket analysis
- Time
  - User preference shift (used to like X; decay?)
  - Item ‘newness’
- Overall success evaluation (A/B test bad recs?)

Additional Considerations

- Deploying the Recommender
  - In the middle of the night, compute similarities, factorize, predict or reconstruct, cache predictions, use throughout the next day?
  - At request time, calculate on the fly for the most up-to-date recommendation?