

Recommenders

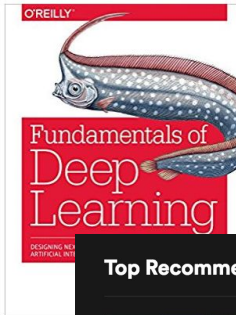
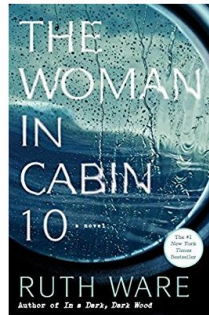
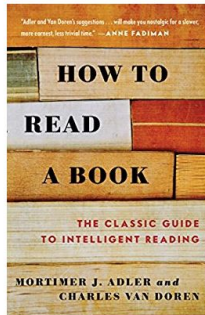
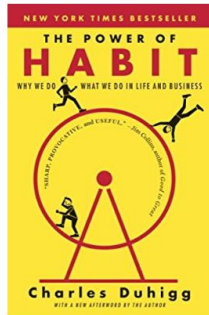
Natalie Hunt



- Where are recommenders used? (Hint: everywhere!)
- What does our dataset look like?
- What are the high-level approaches to building a recommender?
 - Content-based
 - Collaborative filtering
 - Matrix factorization
- How do we evaluate our recommender system?
- How to deal with “cold start”?
- What are the computational performance concerns?

Where are recommenders used?

Recommended for You in Kindle Books



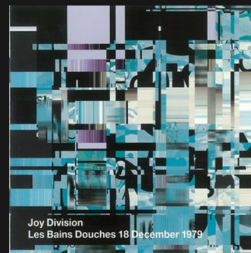
coursera

My Courses | Course Catalog

We have recommendations for you

We combed our catalog and found courses and Specializations that we think match your interests. Browse our recommendations below, and start learning something new today.

Top Recommendations For You



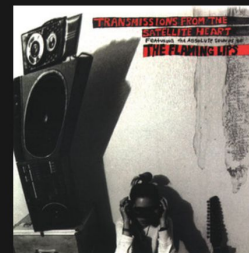
Les Bains Douches
Joy Division



Beautiful Things (Deluxe)
Anthony Green



Spirit Phone
Lemon Demon



Transmissions From The Satellite
Heart
The Flaming Lips

Top Picks for Natalie



Business Goals:

What will the user **like**?

What will the user **buy**?

What will the user **click**?

Name a business that
cares are each of these,
and tell us why they care.

Data Science Canon:

Netflix's \$1,000,000 Prize (Oct. 2006 - July 2009)

NETFLIX

Netfix Prize

COMPLETED

Home Rules Leaderboard Update

Leaderboard

Showing Test Score. [Click here to show quiz score](#)

Display top 20 leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31

Goal: Beat Netflix's own recommender by 10%.

Took almost 3 years.

The winning team used gradient boosted decision trees over the predictions of **500** other models.

Netflix never deployed the winning algorithm.

Let's learn recommenders.

Today we'll learn:

1. How to **build** a recommender,
2. How to **evaluate** your recommender, and
3. How to **deploy** your recommender.

... and tomorrow we'll do a case study with recommender systems.

Popularity:

- Make the **same** recommendation to **every** user, based only on the popularity of an item.
- E.g. Twitter “Moments”

Content-based (aka, Content filtering):

- Predictions are made based on the properties/characteristics of an item.
- User behavior is **not** considered.
- E.g. Pandora Radio

Collaborative filtering:

- Only consider past user behavior. (**not** content properties...)
- User-User similarity: ...
- Item-Item similarity: ...
- E.g.
 - Netflix & Amazon Recommendations,
 - Google Ads,
 - Facebook Ads, Search, Friends Rec., News feed, Trending news, Rank Notifications, Rank Comments

Matrix Factorization Methods:

- Find latent features (aka, factors)

What does our dataset look like?

User	Item				
	A	B	C	D	...
	Al	1	?	2	?
	Bob	?	2	3	4
	Cat	3	?	1	5
	Dan	?	2	?	?
	Ed	2	?	?	1
	...				

We have explicit ratings, plus a bunch of missing values.

What company might have data like this?



Btw, we call this the *utility matrix*.

What does our dataset look like?

User	Item				
	A	B	C	D	...
	Al	0	1	0	1
	Bob	0	0	1	0
	Cat	0	1	1	1
	Dan	1	0	0	1
	Ed	0	1	0	0
	...				

We have implicit feedback,
and no missing values.

What company
might have data
like this?



Btw, we call this
the *utility matrix*.



	Item				
	A	B	C	D	...
User	Al	1	?	2	?
	Bob	?	2	3	4
	Cat	3	?	1	5

We look at all pairs of users and calculate their similarity.

How can we calculate the similarity of these row vectors?
(We'll get there.)

User	Item			
	A	B	C	D
	1	?	2	?
	?	2	3	4
	3	?	1	5
	?	2	?	?
	2	?	?	1
	...			

We look at all pairs of items and calculate their similarity.

How can we calculate the similarity of these column vectors?
(We'll get there.)

User-User:

		Item				
User		A	B	C	D	...
	Al	1	?	2	?	
	Bob	?	2	3	4	
	Cat	3	?	1	5	

Let: $m = \text{\#users},$ $n = \text{\#items}$

We want to compute the similarity of all pairs.

What is the algorithmic efficiency of each approach?


Item-Item:

		Item			
User		A	B	C	D
	Al	1	?	2	?
	Bob	?	2	3	4
	Cat	3	?	1	5
	Dan	?	2	?	?
	Ed	2	?	?	1
	...				

User-User: $O(m^2n)$ **Item-Item:** $O(mn^2)$

Which one is better?


Similarity Metric using Euclidean Distance

What's the range? 

$$\text{dist}(a, b) = ||a - b|| = \sqrt{\sum_i (a_i - b_i)^2}$$

But we're interested in a **similarity**, so let's do this instead:

When use
this?

What's the range? 

$$\text{similarity}(a, b) = \frac{1}{1 + \text{dist}(a, b)}$$

Similarity Metric using Pearson Correlation

What's the range?

$$\text{pearson}(a, b) = \frac{\text{cov}(a, b)}{\text{std}(a) * \text{std}(b)} = \frac{\sum_i (a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_i (a_i - \bar{a})^2} \sqrt{\sum_i (b_i - \bar{b})^2}}$$


When use this?

But we're interested in a **similarity**, so let's do this instead:

What's the range?

$$\text{similarity}(a, b) = 0.5 + 0.5 * \text{pearson}(a, b)$$


Similarity Metric using Cosine Similarity

What's the range? 

$$\cos(\theta_{a,b}) = \frac{a \cdot b}{||a|| ||b||} = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2} \sqrt{\sum_i b_i^2}}$$


But we're interested in a **standardized similarity**, so let's do this instead:

When use
this?

What's the range? 

$$\text{similarity}(a, b) = 0.5 + 0.5 * \cos(\theta_{a,b})$$

Similarity Metric using Jaccard Index

What's the range?  $\text{similarity}(a, b) = \frac{|U_a \cap U_b|}{|U_a \cup U_b|}$

U_k denotes the set of users who rated item k

When use this?

The Similarity Matrix

Pick a similarity metric, create the similarity matrix:

	item 1	item 2	item 3	...
item 1	1	0.3	0.2	...
item 2	0.3	1	0.7	...
item 3	0.2	0.7	1	...
...

Say user u hasn't rated item i . We want to predict the rating that this user *would* give this item.

$$\text{rating}(u, i) = \frac{\sum_{j \in I_u} \text{similarity}(i, j) * r_{u,j}}{\sum_{j \in I_u} \text{similarity}(i, j)}$$

I_u = set of items rated by user u

$r_{u,j}$ = user u 's rating of item j

We order by descending predicted rating for a single user, and recommend the top k items to the user.

This calculation of predicted ratings can be very costly. To mitigate this issue, we will only consider the n most similar items to an item when calculating the prediction.

$$\text{rating}(u, i) = \frac{\sum_{j \in I_u \cap N_i} \text{similarity}(i, j) * r_{u,j}}{\sum_{j \in I_u \cap N_i} \text{similarity}(i, j)}$$

I_u = set of items rated by user u

$r_{u,j}$ = user u 's rating of item j

N_i is the n items which are most similar to item i

Deploying the recommender

In the middle of the night:

- Compute similarities between all pairs of items.
- Compute the neighborhood of each item.

At request time:

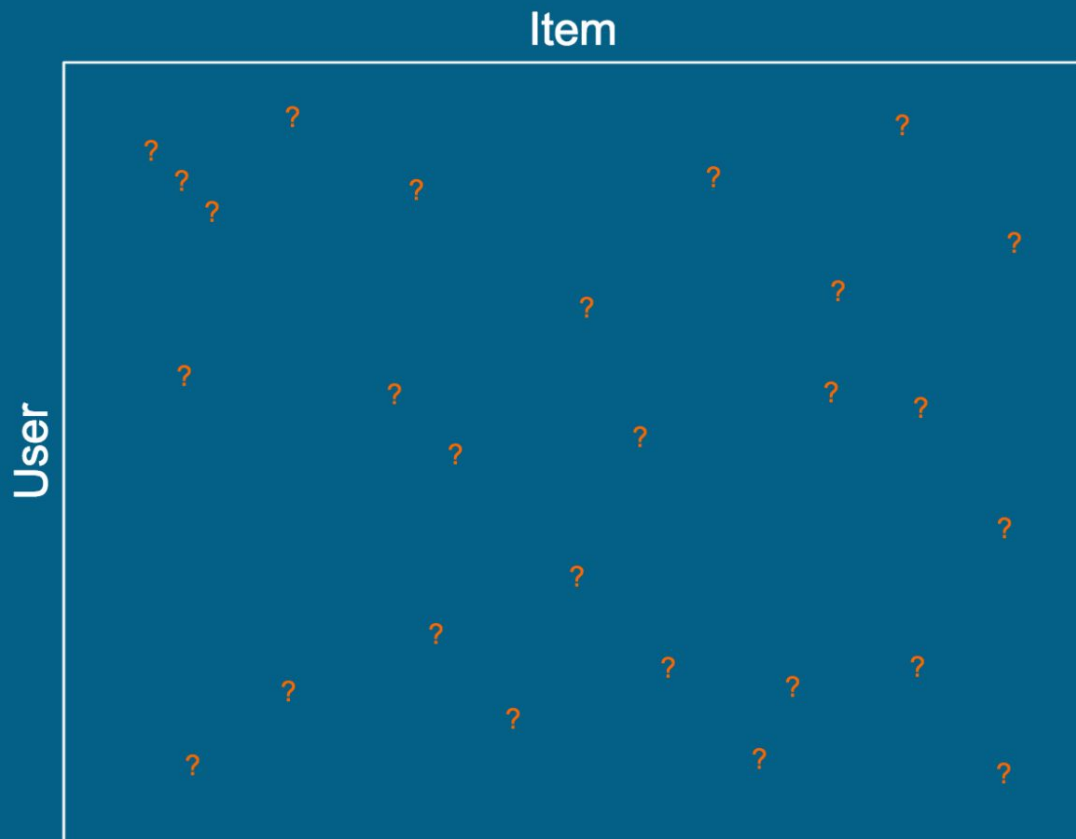
- Predict scores for candidate items, and make a recommendation.

How do we evaluate our recommender system?

Is it possible to do cross-validation like normal?

Before we continue, let's review: Why do we perform cross-validation?

Quick warning: Recommenders are inherently hard to validate. There is a lot of discussion in academia (research papers) and industry (here, Kaggle, Netflix, etc) about this. There is no ONE answer for all datasets.



For this slide, the question marks denote the holdout set (**not** missing values).

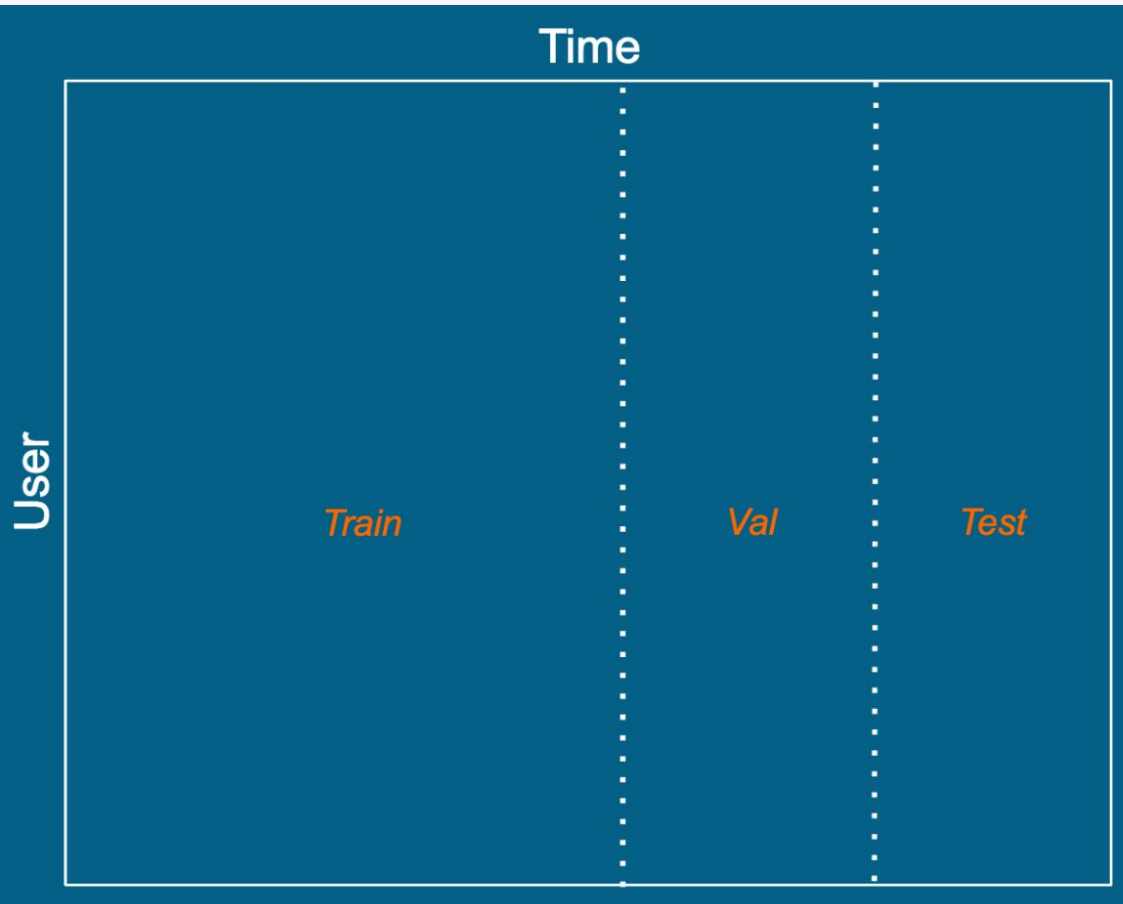
We can calculate MSE between the targets and our predictions over the holdout set.

(K-fold cross-validation is optional.)

Recall: Why do we perform cross-validation?

Why isn't the method above a true estimate of a recommender's performance in the field?

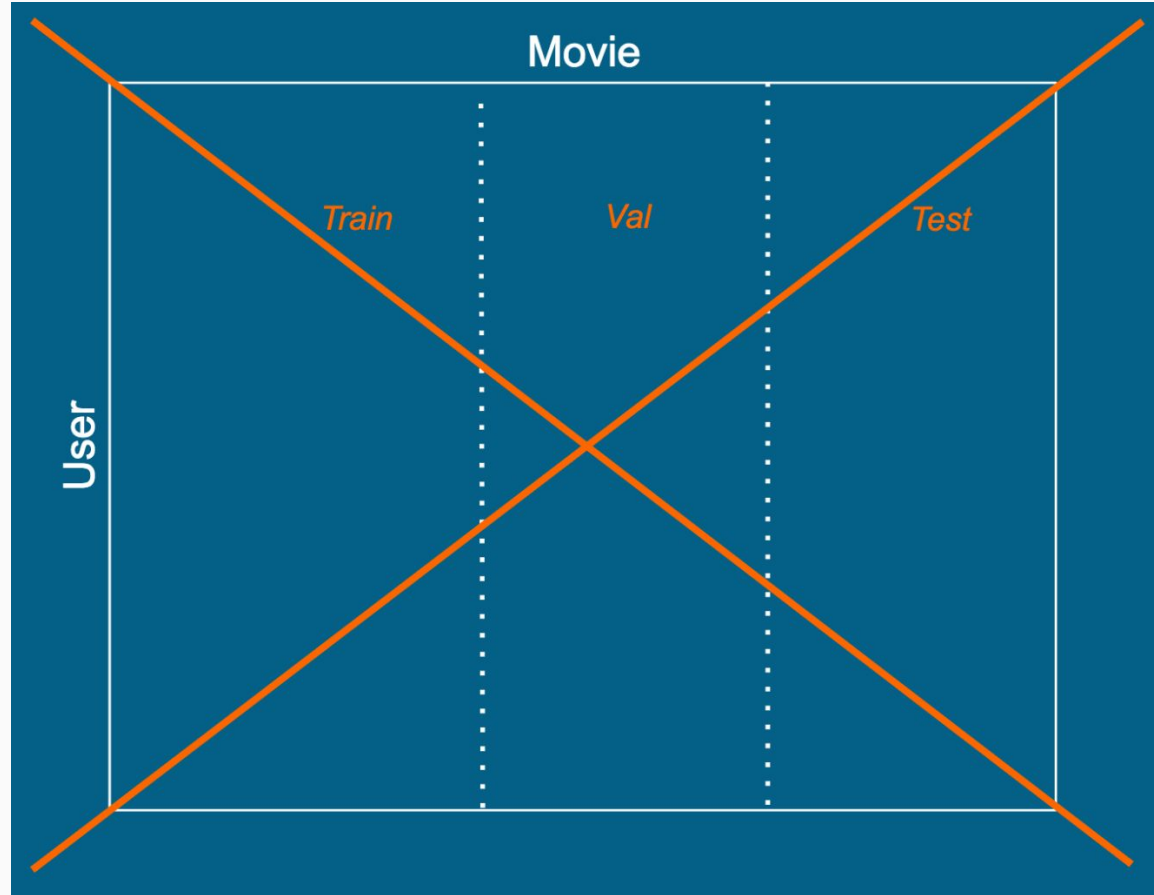
Why would A/B testing be better?



What's the deal with this?

I.e. Why might we prefer doing this instead of the more “normal” cross-validation from the previous slide?

DON'T DO THIS! Why?



How to deal with “cold start”?

Scenario: A new user signs up.

What will our recommender do (assume we're using item-item similarities)?

One strategy: Force users to rate 5 items as part of the signup process. AND/OR Recommend popular items at first.

Scenario: A new item is introduced.

What will our recommender do (assume we're using item-item similarities)?

One strategy: Put it in the “new releases” section until enough users rate it AND/OR use item metadata if any exists.

How to deal with “cold start”?

Scenario: A new user signs up.
What will our recommender do
(assume we're YouTube and
we're using item popularity to
make recommendations)?

This really isn't a problem...

Scenario: A new item is introduced.
What will our recommender do
(assume we're Youtube and we're
using item popularity to make
recommendations)?

One strategy: Don't use total
number of views as the popularity
metric (we'd have a *rich-get-richer*
situation). Use something else...