Recommender Systems

Schwartz

August 16, 2017

Recommender Systems: the ultimate win win?

Stephen Covey's *The 7 Habits of Highly Effective People* are:

Self-Mastery

1. *Be Proactive* in your Circle of Influence/Circle of Concern
2. *Begin with the End in Mind*
3. *Put First Things First*:

	Important	Not Important
Urgent	Do This	AVOID*
Not Urgent	Then this	Never do this

*Tyranny of the Urgent

Interdependence

4. *Think Win-Win*: why are recommender systems win-wins?
5. *Seek First to Understand, Then to be Understood*
6. *Synergize*

Sharpening the Saw

7. *"Always be Getting Better"*

Objectives

- Recommender Systems
  - Popularity
  - Content-Based
  - **Collaborative Filtering**
- Cold Start
- Matrix Factorization
  - SVD vs UVD vs NMF
  - UVD via SGD
  - Bias modeling
  - Regularization
  - Performance
- Evalutation

The Ultimate Win-Win Opportunity

- Know what the user will *enjoy*: e.g., Facebook
- Know what the user will *click*: e.g., GoogleAds
- Know what the user will *buy*: e.g., Amazon

The Ultimate Win-Win Opportunity

- Know what the user will *enjoy*: e.g., Facebook
- Know what the user will *click*: e.g., GoogleAds
- Know what the user will *buy*: e.g., Amazon

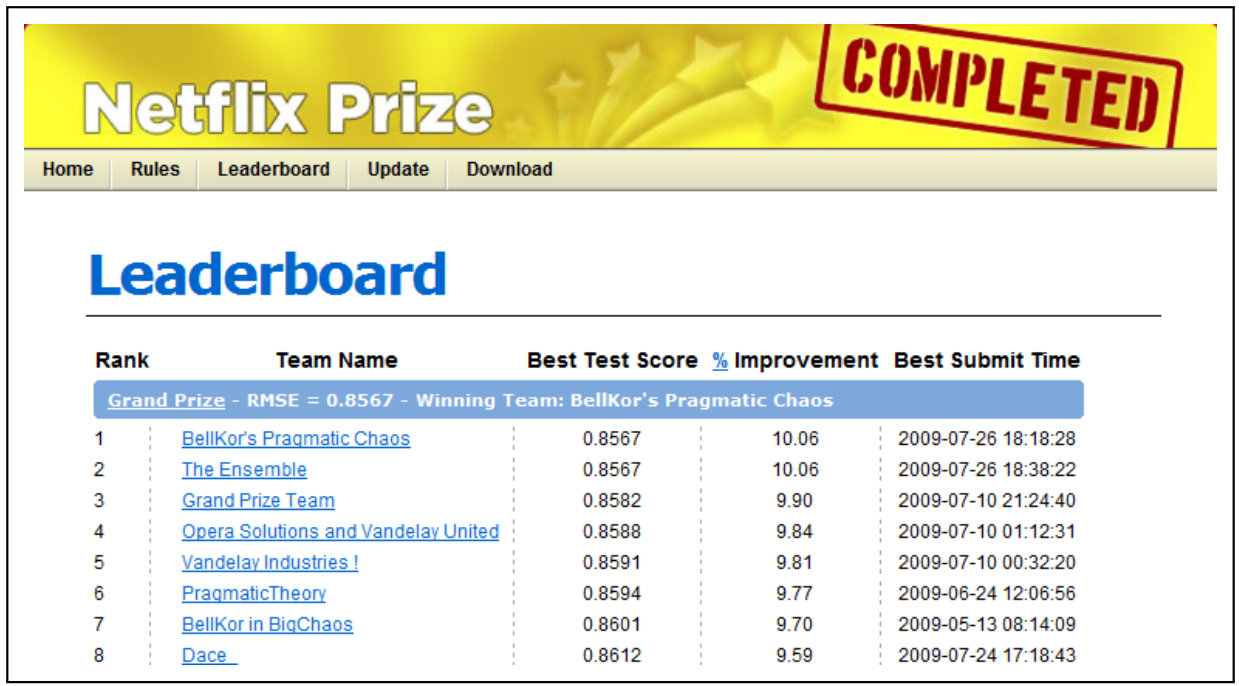
Train: 480,189 users gave 17,770 movies 100,480,507 ratings (1-5)

Test: RMSE for 2,817,131 ratings; average user rated over 200 movies; average movie rated by over 5000 users

- Netflix would award the $1M prize for beating Netflix's own model by 10% (RMSE of 0.9514 to 0.8563)

The Ultimate Win-Win Opportunity

- Know what the user will *enjoy*: e.g., Facebook
- Know what the user will *click*: e.g., GoogleAds
- Know what the user will *buy*: e.g., Amazon



Train: 480,189 users gave 17,770 movies 100,480,507 ratings (1-5)

Test: RMSE for 2,817,131 ratings; average user rated over 200 movies; average movie rated by over 5000 users

- Netflix would award the $1M prize for beating Netflix's own model by 10% (RMSE of 0.9514 to 0.8563)
- It took almost 3 years; "progress prizes" were awarded every year; 8.43% improvement after one year
- Final top two teams were merger of several top teams and Netflix never implemented the winning solution

Types of Recommender Systems

Types of Recommender Systems

Popularity

- Same recommendation for all
- Based on item popularity
- E.g. Twitter “Moments”

Types of Recommender Systems

Popularity

- Same recommendation for all
- Based on item popularity
- E.g. Twitter “Moments”

Content-Based

- I.e., Content Filtering
- Predictions using item features
  - Does not utilize user behavior
  - Does utilize search criterion
- E.g., Pandora Radio

Types of Recommender Systems

Popularity

- Same recommendation for all
- Based on item popularity
- E.g. Twitter "Moments"

Content-Based

- I.e., Content Filtering
- Predictions using item features
  - Does not utilize user behavior
  - Does utilize search criterion
- E.g., Pandora Radio

Collaborative Filtering

- User-User similarity
- Item-Item similarity
  - Does utilize user behavior
  - Does not utilize item content
- E.g., Netflix, Amazon

Types of Recommender Systems

Popularity

- Same recommendation for all
- Based on item popularity
- E.g. Twitter "Moments"

Content-Based

- I.e., Content Filtering
- Predictions using item features
- Does not utilize user behavior
- Does utilize search criterion
- E.g., Pandora Radio

Collaborative Filtering

- User-User similarity
- Item-Item similarity
- Does utilize user behavior
- Does not utilize item content
- E.g., Netflix, Amazon

Matrix Factorization Methods

- Find and Leverage Latent Features (factors)
- Expands NMF methodology

Types of Recommender Systems

Popularity

- Same recommendation for all
- Based on item popularity
- E.g. Twitter "Moments"

Content-Based

- I.e., Content Filtering
- Predictions using item features
- Does not utilize user behavior
- Does utilize search criterion
- E.g., Pandora Radio

Collaborative Filtering

- User-User similarity
- Item-Item similarity
- Does utilize user behavior
- Does not utilize item content
- E.g., Netflix, Amazon

Matrix Factorization Methods

- Find and Leverage Latent Features (factors)
- Expands NMF methodology

Many Recommender Systems are Hybrids of These!

Collaborative Filtering

Utility Matrix (the data)

n users and p items

	Item				
	A	B	C	D	$\cdots$
Al	1	?	2	?	
Bob	?	2	3	4	
Cat	3	?	1	5	
Dan	?	2	?	?	
Ed	2	?	?	1	
$\vdots$					

Table: Netflix Rating Data

User-Item Utility Matrix

	Item				
	A	B	C	D	$\cdots$
Al	0	?	0	?	
Bob	?	0	1	1	
Cat	1	?	0	1	
Dan	?	0	?	?	
Ed	0	?	?	0	
$\vdots$					

Table: Youtube Next Data

User-Item Utility Matrix

User Similarity

n users and p items

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

Table: Netflix Rating Data

User-Item Utility Matrix

	Item				
	A	B	C	D	⋯
Al	0	?	0	?	
Bob	?	0	1	1	
Cat	1	?	0	1	
Dan	?	0	?	?	
Ed	0	?	?	0	
⋮					

Table: Youtube Next Data

User-Item Utility Matrix

$$O(n \times n \times p)$$

Item Similarity

n users and p items

	Item				
	A	B	C	D	$\cdots$
Al	1	?	2	?	
Bob	?	2	3	4	
Cat	3	?	1	5	
Dan	?	2	?	?	
Ed	2	?	?	1	
$\vdots$					

Table: Netflix Rating Data

User-Item Utility Matrix

	Item				
	A	B	C	D	$\cdots$
Al	0	?	0	?	
Bob	?	0	1	1	
Cat	1	?	0	1	
Dan	?	0	?	?	
Ed	0	?	?	0	
$\vdots$					

Table: Youtube Next Data

User-Item Utility Matrix

$$O(p \times p \times n)$$

What's easier: Item-Item Similarity or User-User Similarity?

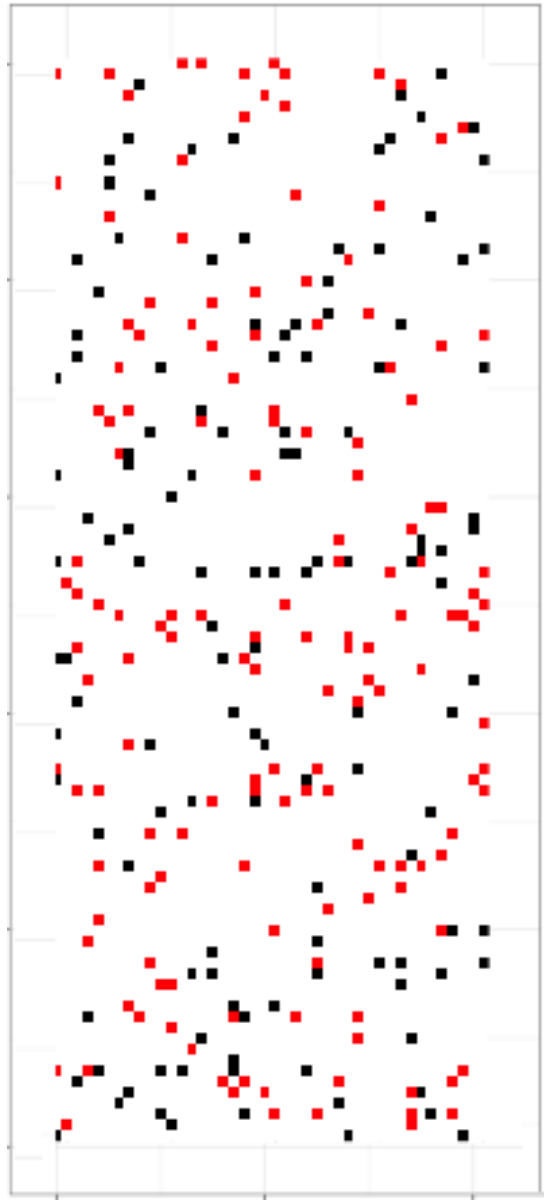
What's easier: Item-Item Similarity or User-User Similarity?

Item-Item

What's easier: Item-Item Similarity or User-User Similarity?

Item-Item

Many businesses have $n > p$

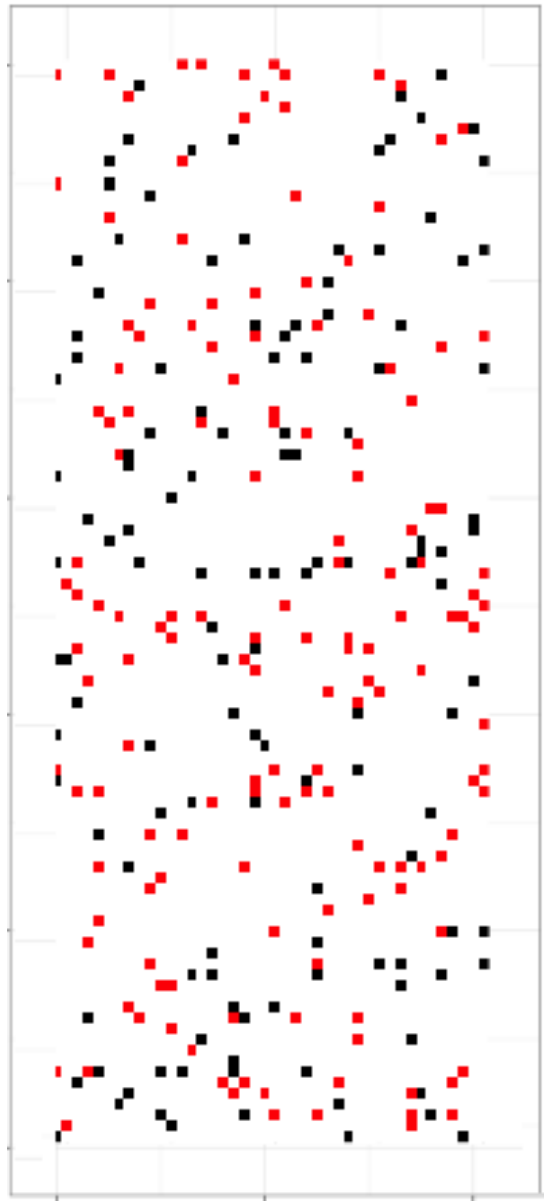


What's easier: Item-Item Similarity or User-User Similarity?

Item-Item

Many businesses have $n > p$

1. Computation:
   $O(p \times p \times n) < O(n \times n \times p)$



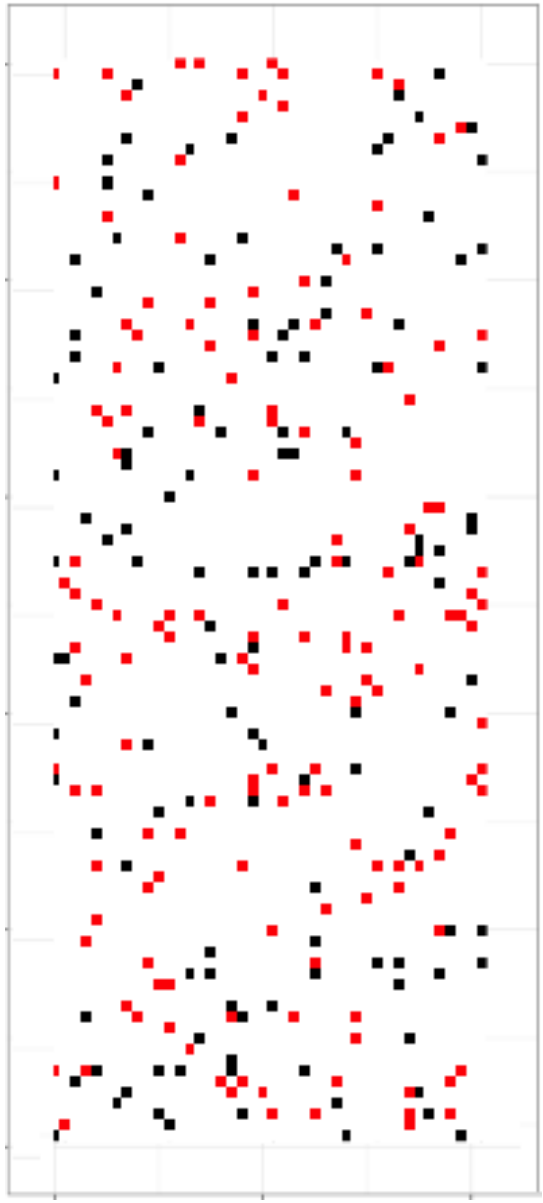
What's easier: Item-Item Similarity or User-User Similarity?

Item-Item

Many businesses have $n > p$

1. Computation:
   $O(p \times p \times n) < O(n \times n \times p)$
2. Stability:
   $Item_a$-$Item_b$ > $User_1$-$User_2$ overlap

Less "cold start":
quicker to onboard new items



Similarity Metrics I

Similarity Metrics I

- **Euclidean**

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

Similarity Metrics I

- **Euclidean**

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

- **Pearson Correlation**

$$Pearson(a, b) = \frac{Cov(a, b)}{Std(a)Std(b)} = \frac{\sum_i (a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_i (a_i - \bar{a})^2}\sqrt{\sum_i (b_i - \bar{b})^2}}$$

Similarity Metrics I

- **Euclidean**

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

- **Pearson Correlation**

$$Pearson(a,b) = \frac{Cov(a,b)}{Std(a)Std(b)} = \frac{\sum_i (a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_i (a_i - \bar{a})^2}\sqrt{\sum_i (b_i - \bar{b})^2}}$$

- **Cosine Similarity**

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

Similarity Metrics I

- **Euclidean**

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

- **Pearson Correlation**

$$Pearson(a,b) = \frac{Cov(a,b)}{Std(a)Std(b)} = \frac{\sum_i (a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_i (a_i - \bar{a})^2}\sqrt{\sum_i (b_i - \bar{b})^2}}$$

- **Cosine Similarity**

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

- *How do these work?*

Similarity Metrics I

- **Euclidean**

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

- **Pearson Correlation**

$$Pearson(a,b) = \frac{Cov(a,b)}{Std(a)Std(b)} = \frac{\sum_i (a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_i (a_i - \bar{a})^2}\sqrt{\sum_i (b_i - \bar{b})^2}}$$

- **Cosine Similarity**

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

- *How do these work? What do they do for similarly rated items?*

Similarity Metrics II (similar $\rightarrow$ 1; dissimilar $\rightarrow$ 0)

- **Euclidean**

$$EuclideanSimilarity(a, b) = \frac{1}{1 + Dist(a, b)}$$

- **Pearson Correlation**

$$PearsonSimilarity(a, b) = \frac{1}{2} + \frac{1}{2} Pearson(a, b)$$

- **Cosine Similarity**

$$CosineSimilarity(a, b) = \frac{1}{2} + \frac{1}{2} Cos(\theta_{a,b})$$

Similarity Metrics II (similar $\rightarrow$ 1; dissimilar $\rightarrow$ 0)

- **Euclidean**

$$EuclideanSimilarity(a, b) = \frac{1}{1 + Dist(a, b)}$$

- **Pearson Correlation**

$$PearsonSimilarity(a, b) = \frac{1}{2} + \frac{1}{2} Pearson(a, b)$$

- **Cosine Similarity**

$$CosineSimilarity(a, b) = \frac{1}{2} + \frac{1}{2} Cos(\theta_{a,b})$$

- **Jaccard Index**

$$JaccardSimilarity(a, b) = \frac{|U_a \cap U_b|}{|U_a \cup U_b|}$$

where U_{item} is the set of users who rated *item*

Similarity Matrix and... how to make recommendation?

	Item				
	a	b	c	d	$\cdots$
Al	1	?	2	?	
Bob	?	2	3	4	
Cat	3	?	1	5	
Dan	?	2	?	?	
Ed	2	?	?	1	
$\vdots$					

Table: Netflix Rating Data X

Similarity Matrix and... how to make recommendation?

	Item				
	a	b	c	d	$\cdots$
Al	1	?	2	?	
Bob	?	2	3	4	
Cat	3	?	1	5	
Dan	?	2	?	?	
Ed	2	?	?	1	
$\vdots$					

Table: Netflix Rating Data X

	Item			
	a	b	c	$\cdots$
item a	1	s_{12}	s_{13}	
item b	s_{21}	1	s_{23}	
item c	s_{31}	s_{32}	1	
$\vdots$				

Table: Similarity Matrix S

Similarity Matrix and... how to make recommendation?

	Item				
	a	b	c	d	$\cdots$
Al	1	?	2	?	
Bob	?	2	3	4	
Cat	3	?	1	5	
Dan	?	2	?	?	
Ed	2	?	?	1	
$\vdots$					

Table: Netflix Rating Data X

	Item			
	a	b	c	$\cdots$
item a	1	s_{12}	s_{13}	
item b	s_{21}	1	s_{23}	
item c	s_{31}	s_{32}	1	
$\vdots$				

Table: Similarity Matrix S

Similarity Matrix and... how to make recommendation?

	Item				
	a	b	c	d	⋯
Al	1	?	2	?	
Bob	?	2	3	4	
Cat	3	?	1	5	
Dan	?	2	?	?	
Ed	2	?	?	1	
⋮					

Table: Netflix Rating Data X

	Item			
	a	b	c	⋯
item a	1	s_{12}	s_{13}	
item b	s_{21}	1	s_{23}	
item c	s_{31}	s_{32}	1	
⋮				

Table: Similarity Matrix S

For user u and item i we can predict user rating based on a weighted average of the other items the user has rated I_u

$$PredictedRating(u,i) = \frac{\sum_{j^* \in I_u} similarity(i,j^*) X_{i,j^*}}{\sum_{j^* \in I_u} similarity(i,j^*)}$$

Similarity Matrix and... how to make recommendation?

	Item				
	a	b	c	d	⋯
Al	1	?	2	?	
Bob	?	2	3	4	
Cat	3	?	1	5	
Dan	?	2	?	?	
Ed	2	?	?	1	
⋮					

Table: Netflix Rating Data X

	Item			
	a	b	c	⋯
item a	1	s_{12}	s_{13}	
item b	s_{21}	1	s_{23}	
item c	s_{31}	s_{32}	1	
⋮				

Table: Similarity Matrix S

For user u and item i we can predict user rating based on a weighted average of the other items the user has rated I_u

$$PredictedRating(u,i) = \frac{\sum_{j^* \in I_u \cap N_i} similarity(i,j^*) X_{i,j^*}}{\sum_{j^* \in I_u \cap N_i} similarity(i,j^*)}$$

N_i is the set of the n items most similar to item i

The Cold-Start Problem

What about a new *user*?

The Cold-Start Problem

What about a new *user*?

- Why can't we give this user recommendations?

The Cold-Start Problem

What about a new *user*?

- Why can't we give this user recommendations?
- Ask user to rate 5 item on sign-up?

The Cold-Start Problem

What about a new *user*?

- Why can't we give this user recommendations?
- Ask user to rate 5 item on sign-up?
- Popularity Recommender?

The Cold-Start Problem

What about a new *user*?

- Why can't we give this user recommendations?
- Ask user to rate 5 item on sign-up?
- Popularity Recommender?

What about a new *item*?

- Why can't we recommend this item?

The Cold-Start Problem

What about a new *user*?

- Why can't we give this user recommendations?
- Ask user to rate 5 item on sign-up?
- Popularity Recommender?

What about a new *item*?

- Why can't we recommend this item?
- "New Content" section?

The Cold-Start Problem

What about a new *user*?

- Why can't we give this user recommendations?
- Ask user to rate 5 item on sign-up?
- Popularity Recommender?

What about a new *item*?

- Why can't we recommend this item?
- "New Content" section?
- Content-Based?
- Popularity Recommender?

Matrix Factorization Methods

UV Decomposition (UVD)

$$\underset{n\times p}{X} \approx \underset{n\times k}{U} \quad \underset{k\times p}{V}$$

$$X_{ij} \approx U_{i.}V_{.j} = \hat{X}_{ij}$$

$$\underset{U,V}{argmin} \sum_{i,j}(X_{ij} - U_{i.}V_{.j})^2$$

UVD

UV Decomposition (UVD)

$$\underset{n\times p}{X} \approx \underset{n\times k}{U} \quad \underset{k\times p}{V}$$

$$X_{ij} \approx U_{i.}V_{.j} = \hat{X}_{ij}$$

$$\underset{U,V}{argmin} \sum_{i,j}(X_{ij} - U_{i.}V_{.j})^2$$

SVD **NMF** **UVD**

UV Decomposition (UVD)

$$\underset{n\times p}{X} \approx \underset{n\times k}{U} \quad \underset{k\times p}{V}$$

$$X_{ij} \approx U_{i.}V_{.j} = \hat{X}_{ij}$$

$$\underset{U,V}{argmin} \sum_{i,j}(X_{ij} - U_{i.}V_{.j})^2$$

SVD

- $X = U\Sigma V^T$

NMF

- $X \approx WH; W, H \geq 0$

UVD

- $X \approx UV$

UV Decomposition (UVD)

$$\underset{n\times p}{X} \approx \underset{n\times k}{U} \quad \underset{k\times p}{V}$$

$$X_{ij} \approx U_{i.} V_{.j} = \hat{X}_{ij}$$

$$\underset{U,V}{argmin} \sum_{i,j} (X_{ij} - U_{i.} V_{.j})^2$$

SVD

- $X = U\Sigma V^T$
- U, V^T orthogonal

NMF

- $X \approx WH; W, H \geq 0$
- W, H non orthogonal

UVD

- $X \approx UV$
- U, V non orthogonal

UV Decomposition (UVD)

$$\underset{n\times p}{X} \approx \underset{n\times k}{U} \quad \underset{k\times p}{V}$$

$$X_{ij} \approx U_{i.} V_{.j} = \hat{X}_{ij}$$

$$\underset{U,V}{argmin} \sum_{i,j} (X_{ij} - U_{i.} V_{.j})^2$$

SVD

- $X = U\Sigma V^T$
- U, V^T orthogonal
- single solution

NMF

- $X \approx WH; W, H \geq 0$
- W, H non orthogonal
- non-unique solutions

UVD

- $X \approx UV$
- U, V non orthogonal
- non-unique solutions

UV Decomposition (UVD)

$$\underset{n\times p}{X} \approx \underset{n\times k}{U}\ \underset{k\times p}{V}$$

$$X_{ij} \approx U_{i.}V_{.j} = \hat{X}_{ij}$$

$$\underset{U,V}{argmin} \sum_{i,j}(X_{ij} - U_{i.}V_{.j})^2$$

SVD

- $X = U\Sigma V^T$
- U, V^T orthogonal
- single solution
- analytical solution

NMF

- $X \approx WH; W, H \geq 0$
- W, H non orthogonal
- non-unique solutions
- iterative optimization

UVD

- $X \approx UV$
- U, V non orthogonal
- non-unique solutions
- iterative optimization

UV Decomposition (UVD)

$$\underset{n\times p}{X} \approx \underset{n\times k}{U} \quad \underset{k\times p}{V}$$

$$X_{ij} \approx U_{i\cdot} V_{\cdot j} = \hat{X}_{ij}$$

$$\underset{U,V}{\operatorname{argmin}} \sum_{i,j} (X_{ij} - U_{i\cdot} V_{\cdot j})^2$$

SVD

- $X = U\Sigma V^T$
- U, V^T orthogonal
- single solution
- analytical solution
- no missing values

NMF

- $X \approx WH; W, H \geq 0$
- W, H non orthogonal
- non-unique solutions
- iterative optimization
- missing values okay

UVD

- $X \approx UV$
- U, V non orthogonal
- non-unique solutions
- iterative optimization
- missing values okay

UV Decomposition (UVD)

$$\underset{n\times p}{X} \approx \underset{n\times k}{U} \quad \underset{k\times p}{V}$$

$$X_{ij} \approx U_{i\cdot} V_{\cdot j} = \hat{X}_{ij}$$

$$\underset{U,V}{\operatorname{argmin}} \sum_{i,j} (X_{ij} - U_{i\cdot} V_{\cdot j})^2$$

SVD

- $X = U\Sigma V^T$
- U, V^T orthogonal
- single solution
- analytical solution
- no missing values
- singular values

NMF

- $X \approx WH; W, H \geq 0$
- W, H non orthogonal
- non-unique solutions
- iterative optimization
- missing values okay
- tunable k

UVD

- $X \approx UV$
- U, V non orthogonal
- non-unique solutions
- iterative optimization
- missing values okay
- tunable k

How do we learn U and V?

How do we learn U and V?

- **Gradient Descent**

$$\frac{\partial}{\partial U_{i'k}} \sum_{i,j} (X_{ij} - U_{i\cdot} \ V_{\cdot j})^2 = \sum_{j} -2(X_{i'j} - U_{i'k} \ V_{kj}) V_{kj}$$

$$\frac{\partial}{\partial V_{kj'}} \sum_{i,j} (X_{ij} - U_{i\cdot} \ V_{\cdot j})^2 = \sum_{i} -2(U_{ij'} - U_{ik} \ V_{kj'}) U_{ik}$$

How do we learn U and V?

- **Gradient Descent**

$$\frac{\partial}{\partial U_{i'k}} \sum_{i,j} (X_{ij} - U_{i\cdot} \;\; V_{\cdot j})^2 = \sum_{j} -2(X_{i'j} - U_{i'k} \;\; V_{kj}) V_{kj}$$

$$\frac{\partial}{\partial V_{kj'}} \sum_{i,j} (X_{ij} - U_{i\cdot} \;\; V_{\cdot j})^2 = \sum_{i} -2(U_{ij'} - U_{ik} \;\; V_{kj'}) U_{ik}$$

- **Alternating Lease Squares (ALS)**
  1. Update $V_{\cdot j}$ using OLS: $X_{\cdot j} = \mathbf{U} V_{\cdot j} + \epsilon_{\cdot j}$

$$\begin{bmatrix} & | & \\ & | & \\ & | & \end{bmatrix} = \begin{bmatrix} \rightarrow\rightarrow \\ \rightarrow\rightarrow \\ \rightarrow\rightarrow \end{bmatrix} \begin{bmatrix} & \downarrow & \\ & \downarrow & \\ & \downarrow & \end{bmatrix}$$

  2. Update $W_{i\cdot}$ using OLS: $X_{i\cdot}^T = \mathbf{V^T} U_{i\cdot}^T + \epsilon_{i\cdot}^T$

$$\begin{bmatrix} - & - \\ & \end{bmatrix}^T = \left(\begin{bmatrix} \rightarrow\rightarrow \\ \end{bmatrix} \begin{bmatrix} \downarrow & \downarrow & \downarrow & \downarrow \\ \downarrow & \downarrow & \downarrow & \downarrow \\ \downarrow & \downarrow & \downarrow & \downarrow \end{bmatrix} \right)^T$$

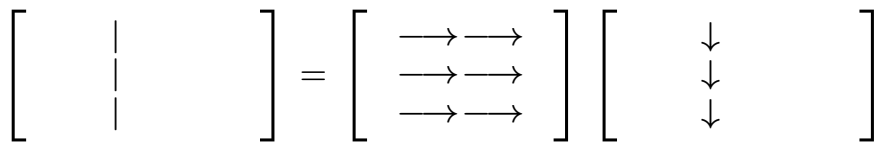
How do we learn U and V?

- **Gradient Descent**

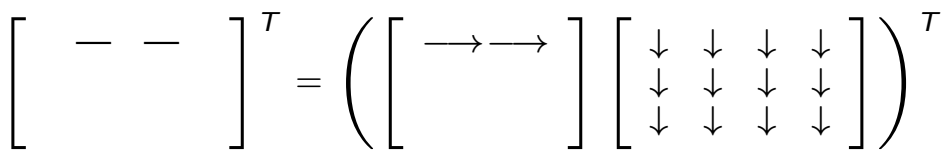
$$\frac{\partial}{\partial U_{i'k}} \sum_{i,j} (X_{ij} - U_{i\cdot} \ V_{\cdot j})^2 = \sum_{j} -2(X_{i'j} - U_{i'k} \ V_{kj}) V_{kj}$$

$$\frac{\partial}{\partial V_{kj'}} \sum_{i,j} (X_{ij} - U_{i\cdot} \ V_{\cdot j})^2 = \sum_{i} -2(U_{ij'} - U_{ik} \ V_{kj'}) U_{ik}$$

- **Alternating Lease Squares (ALS)**
  1. Update $V_{\cdot j}$ using OLS: $X_{\cdot j} = \mathbf{U} V_{\cdot j} + \epsilon_{\cdot j}$



  2. Update $W_{i\cdot}$ using OLS: $X_{i\cdot}^T = \mathbf{V^T} U_{i\cdot}^T + \epsilon_{i\cdot}^T$



- **Lee and Seung's "multiplicative update rules"**
  Initialize non-negative H and W; update with non-negativity constraint:

$$W'_{ik} = W_{ik} \frac{(XH^T)_{ik}}{(WHH^T)_{ik}} \qquad H'_{kj} = H_{kj} \frac{(W^T X)_{kj}}{(W^T WH)_{kj}}$$

Numerical Estimation Comparisons

Alternating Least Squares

Multiplicative Updates

Stochastic Gradient Decent

Numerical Estimation Comparisons

Alternating Least Squares

- Simple idea/implementation
- Parallelizes very well
- Available in Spark/MLlib

Stochastic Gradient Decent

- Faster than ALS (per CPU)

Multiplicative Updates

- Easy implementation
- Faster than SGD

Numerical Estimation Comparisons

Alternating Least Squares

- Simple idea/implementation
- Parallelizes very well
- Available in Spark/MLlib

Stochastic Gradient Decent

- Faster than ALS (per CPU)
- Need to turn learning rate
- Thought to outperform ALS

Multiplicative Updates

- Easy implementation
- Faster than SGD
- No turning parameter

Numerical Estimation Comparisons

Alternating Least Squares

- Simple idea/implementation
- Parallelizes very well
- Available in Spark/MLlib
- Can handle NA's with care

Stochastic Gradient Decent

- Faster than ALS (per CPU)
- Need to turn learning rate
- Thought to outperform ALS
- Works with NA's

Multiplicative Updates

- Easy implementation
- Faster than SGD
- No turning parameter
- Only works for NMF
- Can't handle NA's

UVD + SGD is best in class

Used by the winning Netflix Challenge entry

- No need to impute missing values
- Can model time dynamics (evolving user preference)
- Plus it can do all of these:

$$\underset{U,V}{argmin} \sum_{i,j} (X_{ij} - (U_{i\cdot} V_{\cdot j}))^2$$

UVD + SGD is best in class

Used by the winning Netflix Challenge entry

- No need to impute missing values
- Can model time dynamics (evolving user preference)
- Plus it can do all of these:

$$\underset{U,V,\beta_{Ui},\beta_{Ij},\mu}{\mathit{argmin}} \sum_{i,j}(X_{ij} - (\mu + \beta_{Ui} + \beta_{Ij} + U_{i\cdot}V_{\cdot j}))^2$$

UVD + SGD is best in class

Used by the winning Netflix Challenge entry

- No need to impute missing values
- Can model time dynamics (evolving user preference)
- Plus it can do all of these:

$$\underset{U,V,\beta_{Ui},\beta_{Ij},\mu}{\mathit{argmin}}$$

$$\sum_{i,j}(X_{ij} - (\mu + \beta_{Ui} + \beta_{Ij} + U_{i.}V_{.j}))^2$$

$$+\lambda_2\left(\beta_{Ui}^2 + \beta_{Ij}^2\right)$$

UVD + SGD is best in class

Used by the winning Netflix Challenge entry

- No need to impute missing values
- Can model time dynamics (evolving user preference)
- Plus it can do all of these:

$$\underset{U,V,\beta_{Ui},\beta_{Ij},\mu}{\mathit{argmin}}$$

$$\sum_{i,j}(X_{ij} - (\mu + \beta_{Ui} + \beta_{Ij} + U_{i.}V_{.j}))^2$$

$$+ \lambda_1 \left(||U_{i.}||^2 + ||V_{.j}||^2 \right) + \lambda_2 \left(\beta_{Ui}^2 + \beta_{Ij}^2 \right)$$

UVD + SGD is best in class

Used by the winning Netflix Challenge entry

- No need to impute missing values
- Can model time dynamics (evolving user preference)
- Plus it can do all of these:

$$\underset{U,V,\beta_{Ui},\beta_{Ij},\mu}{\mathit{argmin}}$$

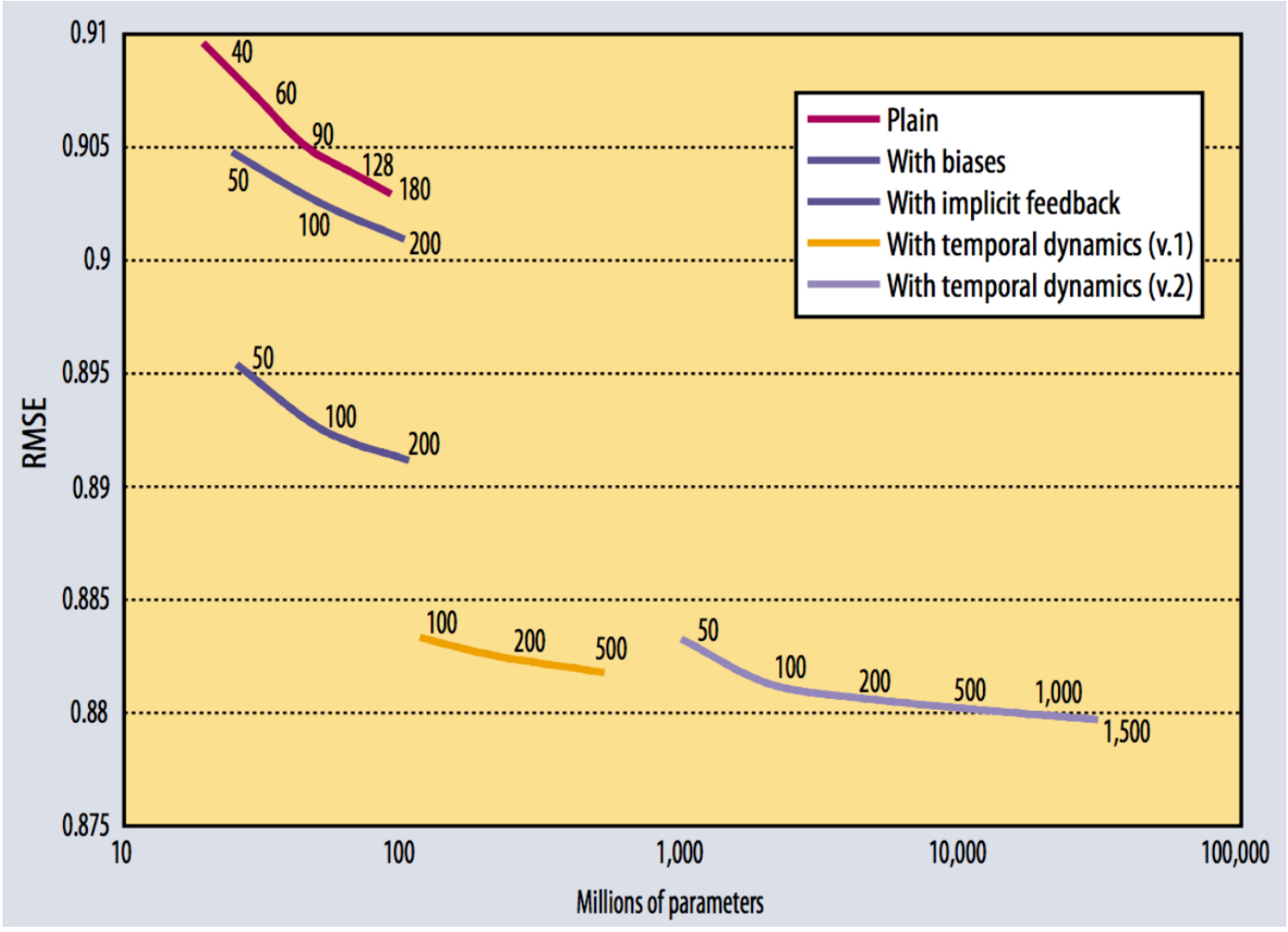
$$\sum_{i,j}(X_{ij} - (\mu + \beta_{Ui} + \beta_{Ij} + U_{i.}V_{.j}))^2$$

$$+ \lambda_1 \left(||U_{i.}||^2 + ||V_{.j}||^2 \right) + \lambda_2 \left(\beta_{Ui}^2 + \beta_{Ij}^2 \right)$$

And Don't Forget *k*!!!

Matrix Factorization Methods on the Netflix Challenge

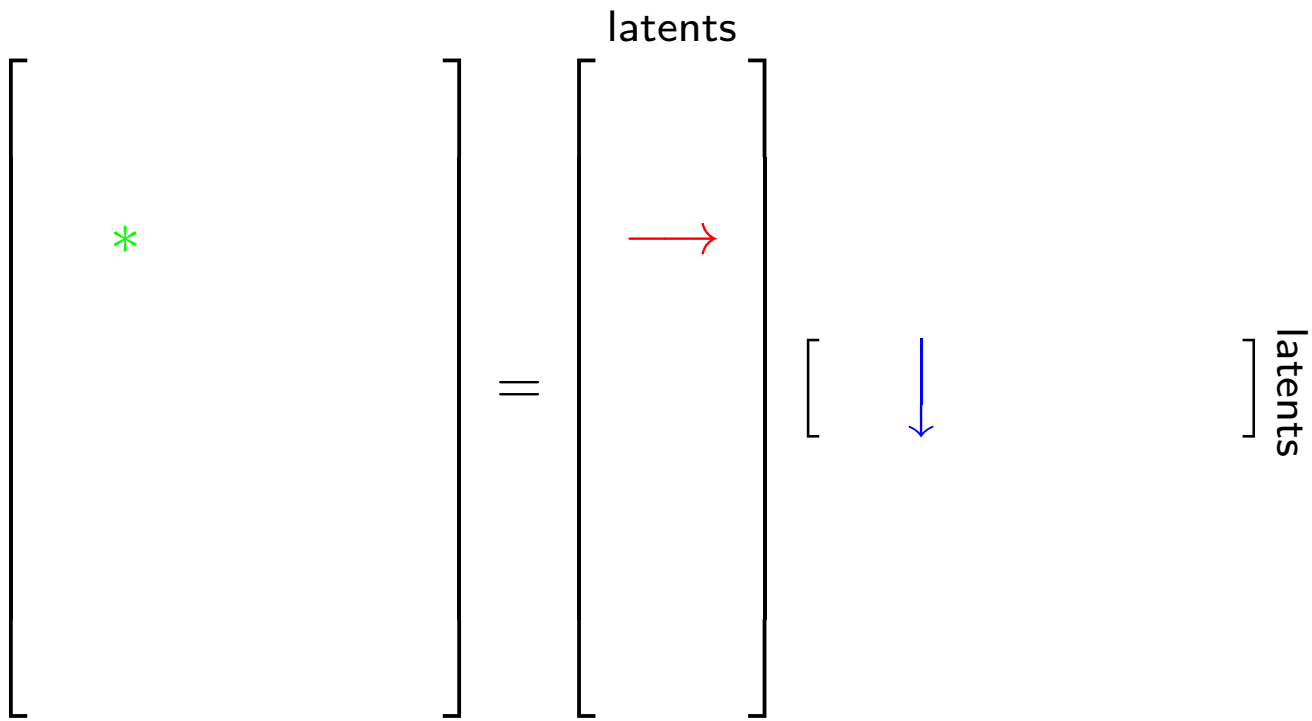
Netflix's in-house model achieves RMSE=0.9514



"Matrix Factorization Techniques for Recommender Systems"

What are the *"latent factors"*?

$$X = U \quad V$$

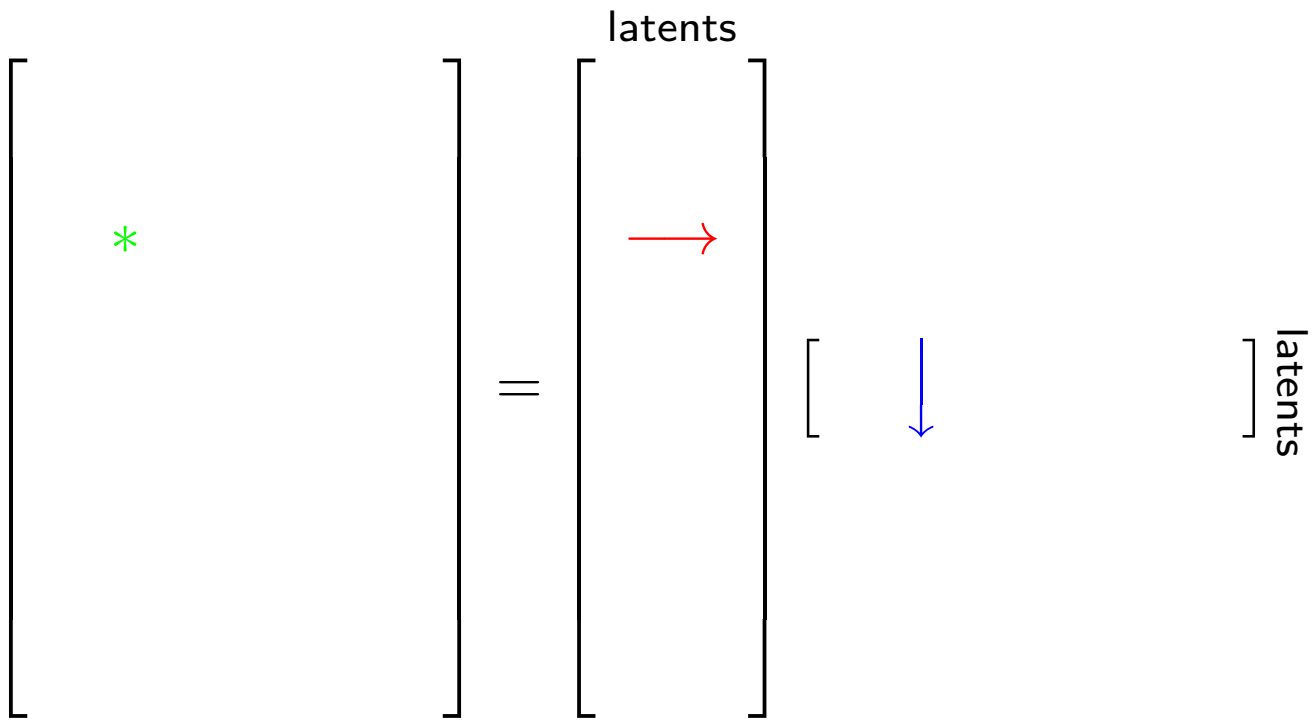


"item composition"

"user parts"

What are the *"latent factors"*?

$$X = U \quad V$$



so rows are "item topics" $\longrightarrow$ *"item composition"*

"user parts" $\longleftarrow$ *and columns are "user archetypes"*

Wrap Up

$$\underset{U,V,\beta_{Ui},\beta_{Ij},\mu}{argmin} \sum_{i,j} (X_{ij} - \left(\mu + \beta_{Ui} + \beta_{Ij} + U_{i\cdot} V_{\cdot j}\right))^2 + \lambda_1 \left(||U_{i\cdot}||^2 + ||V_{\cdot j}||^2\right) + \lambda_2 \left(\beta_{Ui}^2 + \beta_{Ij}^2\right)$$

PROS

CONS

Wrap Up

$$\underset{U,V,\beta_{Ui},\beta_{Ij},\mu}{\textit{argmin}} \sum_{i,j} (X_{ij} - \left(\mu + \beta_{Ui} + \beta_{Ij} + U_{i\cdot} V_{\cdot j}\right))^2 + \lambda_1 \left(||U_{i\cdot}||^2 + ||V_{\cdot j}||^2\right) + \lambda_2 \left(\beta_{Ui}^2 + \beta_{Ij}^2\right)$$

PROS

- Regularization handles sparsity

CONS

Wrap Up

$$\underset{U,V,\beta_{Ui},\beta_{Ij},\mu}{\textit{argmin}} \sum_{i,j} (X_{ij} - \left(\mu + \beta_{Ui} + \beta_{Ij} + U_{i\cdot} V_{\cdot j}\right))^2 + \lambda_1 \left(||U_{i\cdot}||^2 + ||V_{\cdot j}||^2\right) + \lambda_2 \left(\beta_{Ui}^2 + \beta_{Ij}^2\right)$$

PROS

- Regularization handles sparsity
- Fast dot product prediction

CONS

Wrap Up

$$\underset{U,V,\beta_{Ui},\beta_{Ij},\mu}{argmin} \sum_{i,j} (X_{ij} - \left(\mu + \beta_{Ui} + \beta_{Ij} + U_{i\cdot} V_{\cdot j}\right))^2 + \lambda_1 \left(||U_{i\cdot}||^2 + ||V_{\cdot j}||^2\right) + \lambda_2 \left(\beta_{Ui}^2 + \beta_{Ij}^2\right)$$

PROS

- Regularization handles sparsity
- Fast dot product prediction
- Latent feature interpretation

CONS

Wrap Up

$$\underset{U,V,\beta_{Ui},\beta_{Ij},\mu}{\textit{argmin}} \sum_{i,j} (X_{ij} - \left(\mu + \beta_{Ui} + \beta_{Ij} + U_{i\cdot} V_{\cdot j}\right))^2 + \lambda_1 \left(||U_{i\cdot}||^2 + ||V_{\cdot j}||^2\right) + \lambda_2 \left(\beta_{Ui}^2 + \beta_{Ij}^2\right)$$

PROS

- Regularization handles sparsity
- Fast dot product prediction
- Latent feature interpretation
- Can include auxiliary data

CONS

Wrap Up

$$\underset{U,V,\beta_{Ui},\beta_{Ij},\mu}{argmin} \sum_{i,j} (X_{ij} - \left(\mu + \beta_{Ui} + \beta_{Ij} + U_{i\cdot} V_{\cdot j}\right))^2 + \lambda_1 \left(||U_{i\cdot}||^2 + ||V_{\cdot j}||^2\right) + \lambda_2 \left(\beta_{Ui}^2 + \beta_{Ij}^2\right)$$

PROS

- Regularization handles sparsity
- Fast dot product prediction
- Latent feature interpretation
- Can include auxiliary data

CONS

- Re-factorizing new data is slow

Wrap Up

$$\underset{U,V,\beta_{Ui},\beta_{Ij},\mu}{\textit{argmin}} \sum_{i,j} (X_{ij} - \left(\mu + \beta_{Ui} + \beta_{Ij} + U_{i\cdot} V_{\cdot j}\right))^2 + \lambda_1 \left(||U_{i\cdot}||^2 + ||V_{\cdot j}||^2\right) + \lambda_2 \left(\beta_{Ui}^2 + \beta_{Ij}^2\right)$$

PROS

- Regularization handles sparsity
- Fast dot product prediction
- Latent feature interpretation
- Can include auxiliary data

CONS

- Re-factorizing new data is slow
- No strong libraries available...

Wrap Up

$$\underset{U,V,\beta_{Ui},\beta_{Ij},\mu}{\textit{argmin}} \sum_{i,j} (X_{ij} - \left(\mu + \beta_{Ui} + \beta_{Ij} + U_{i\cdot} V_{\cdot j}\right))^2 + \lambda_1 \left(||U_{i\cdot}||^2 + ||V_{\cdot j}||^2\right) + \lambda_2 \left(\beta_{Ui}^2 + \beta_{Ij}^2\right)$$

PROS

- Regularization handles sparsity
- Fast dot product prediction
- Latent feature interpretation
- Can include auxiliary data

CONS

- Re-factorizing new data is slow
- No strong libraries available...
- Fails at cold-start problem

Wrap Up

$$\underset{U,V,\beta_{Ui},\beta_{Ij},\mu}{\textit{argmin}} \sum_{i,j} (X_{ij} - \left(\mu + \beta_{Ui} + \beta_{Ij} + U_{i\cdot} V_{\cdot j}\right))^2 + \lambda_1 \left(||U_{i\cdot}||^2 + ||V_{\cdot j}||^2\right) + \lambda_2 \left(\beta_{Ui}^2 + \beta_{Ij}^2\right)$$

PROS

- Regularization handles sparsity
- Fast dot product prediction
- Latent feature interpretation
- Can include auxiliary data

CONS

- Re-factorizing new data is slow
- No strong libraries available...
- Fails at cold-start problem
- Success is hard to nail down and thus hard to tune against

Wrap Up

$$\underset{U,V,\beta_{Ui},\beta_{Ij},\mu}{\textit{argmin}} \sum_{i,j} (X_{ij} - \left(\mu + \beta_{Ui} + \beta_{Ij} + U_{i\cdot} V_{\cdot j}\right))^2 + \lambda_1 \left(||U_{i\cdot}||^2 + ||V_{\cdot j}||^2\right) + \lambda_2 \left(\beta_{Ui}^2 + \beta_{Ij}^2\right)$$

PROS

- Regularization handles sparsity
- Fast dot product prediction
- Latent feature interpretation
- Can include auxiliary data

CONS

- Re-factorizing new data is slow
- No strong libraries available...
- Fails at cold-start problem
- Success is hard to nail down and thus hard to tune against

If you have a system it's easy to deploy:

At Request Time: Serve up best prediction score

or use factorization dot product to prediction score

Wrap Up

$$\underset{U,V,\beta_{Ui},\beta_{Ij},\mu}{\textit{argmin}} \sum_{i,j} (X_{ij} - \left(\mu + \beta_{Ui} + \beta_{Ij} + U_{i\cdot} V_{\cdot j}\right))^2 + \lambda_1 \left(||U_{i\cdot}||^2 + ||V_{\cdot j}||^2\right) + \lambda_2 \left(\beta_{Ui}^2 + \beta_{Ij}^2\right)$$

PROS

- Regularization handles sparsity
- Fast dot product prediction
- Latent feature interpretation
- Can include auxiliary data

CONS

- Re-factorizing new data is slow
- No strong libraries available...
- Fails at cold-start problem
- Success is hard to nail down and thus hard to tune against

If you have a system it's easy to deploy:

At Request Time: Serve up best prediction score

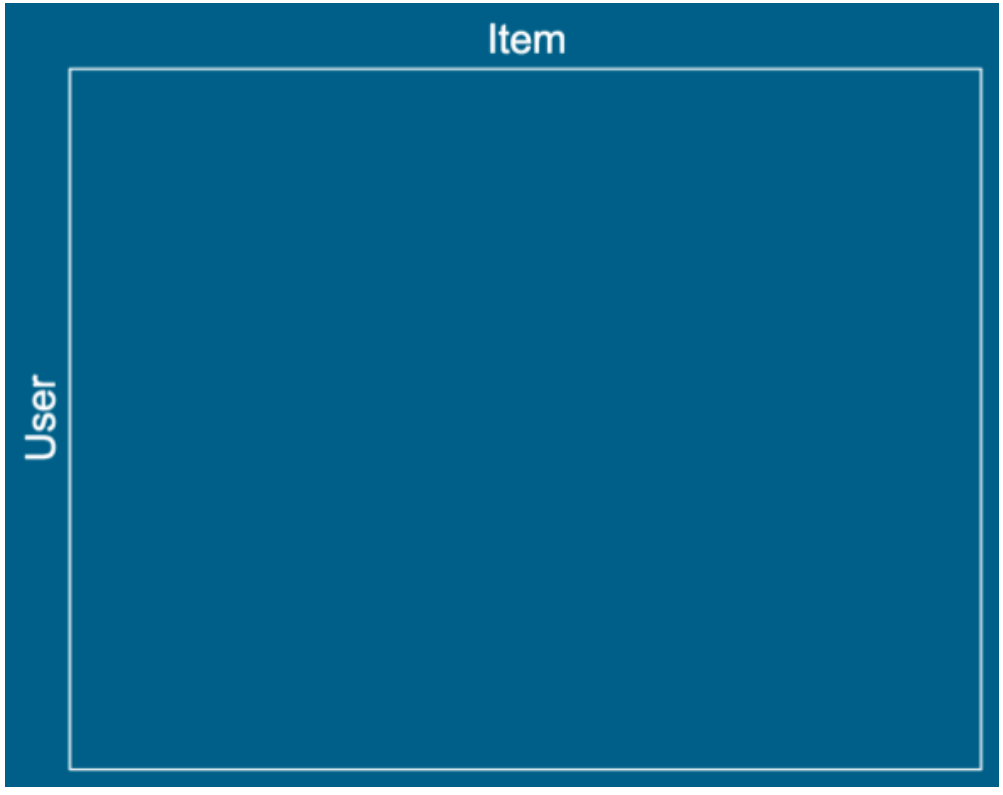
or use factorization dot product to prediction score

During Down Time: Compute pairwise similarity, N_i, and predictions

or re-factorize matrix based on newly acquired data

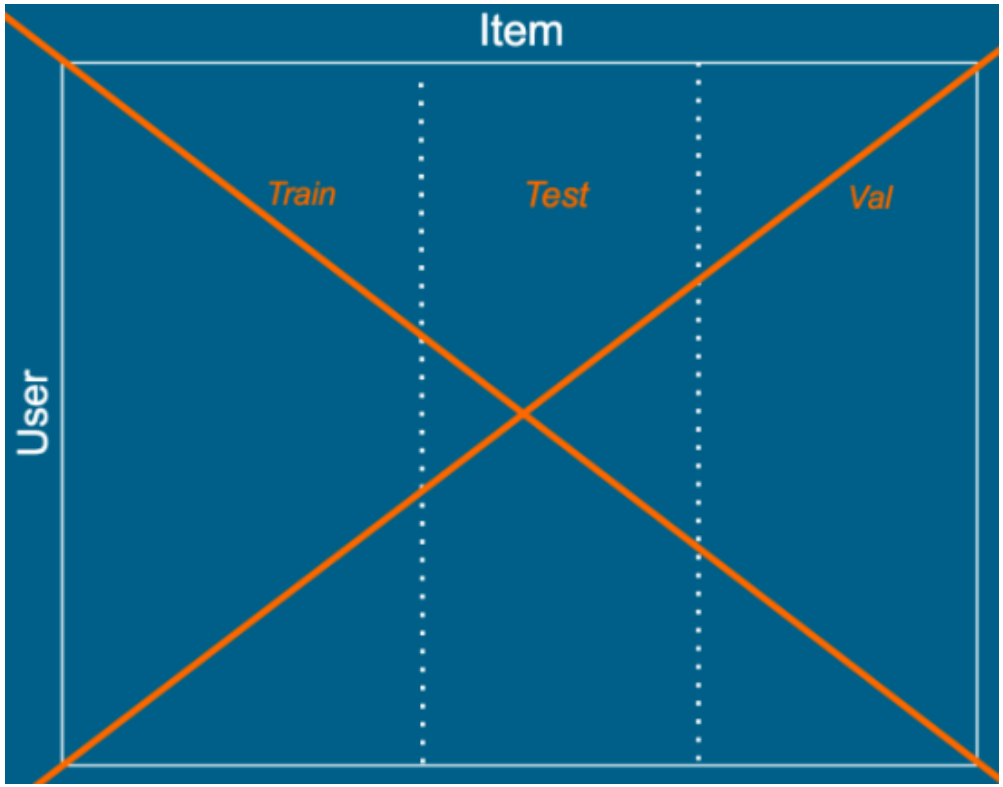
BIG FINISH: *Evaluating the Recommender*

The difficulty is how do you get a system in the first place?



BIG FINISH: *Evaluating the Recommender*

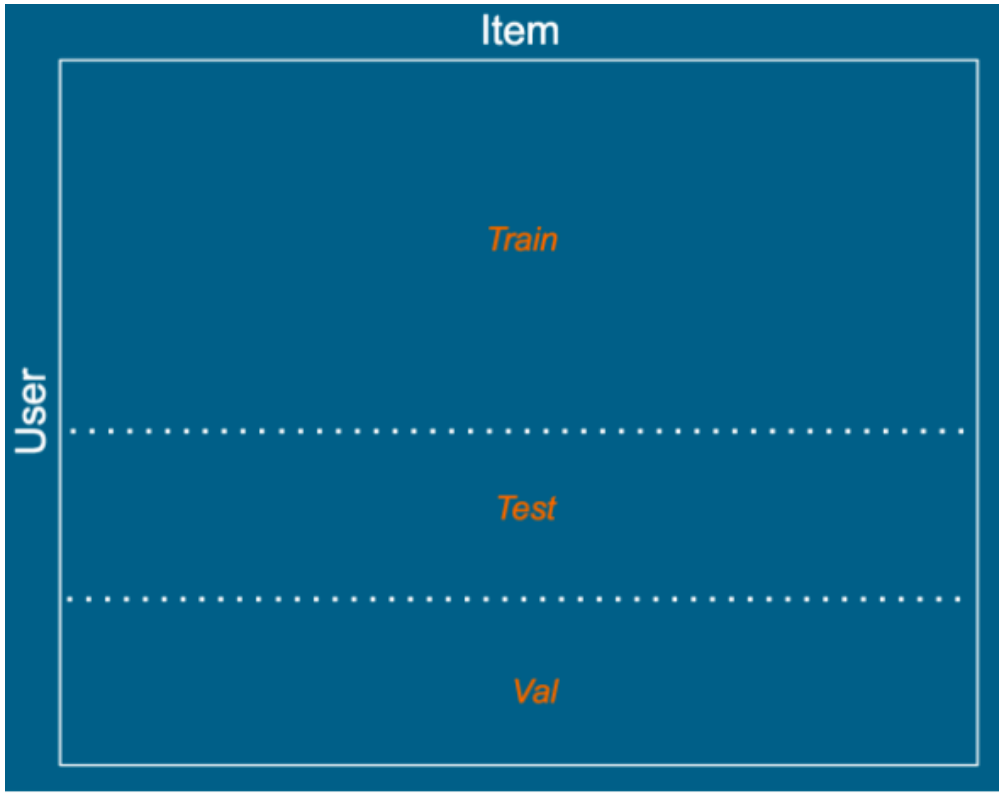
The difficulty is how do you get a system in the first place?



- Leaving out items: doesn't works (?)

BIG FINISH: *Evaluating the Recommender*

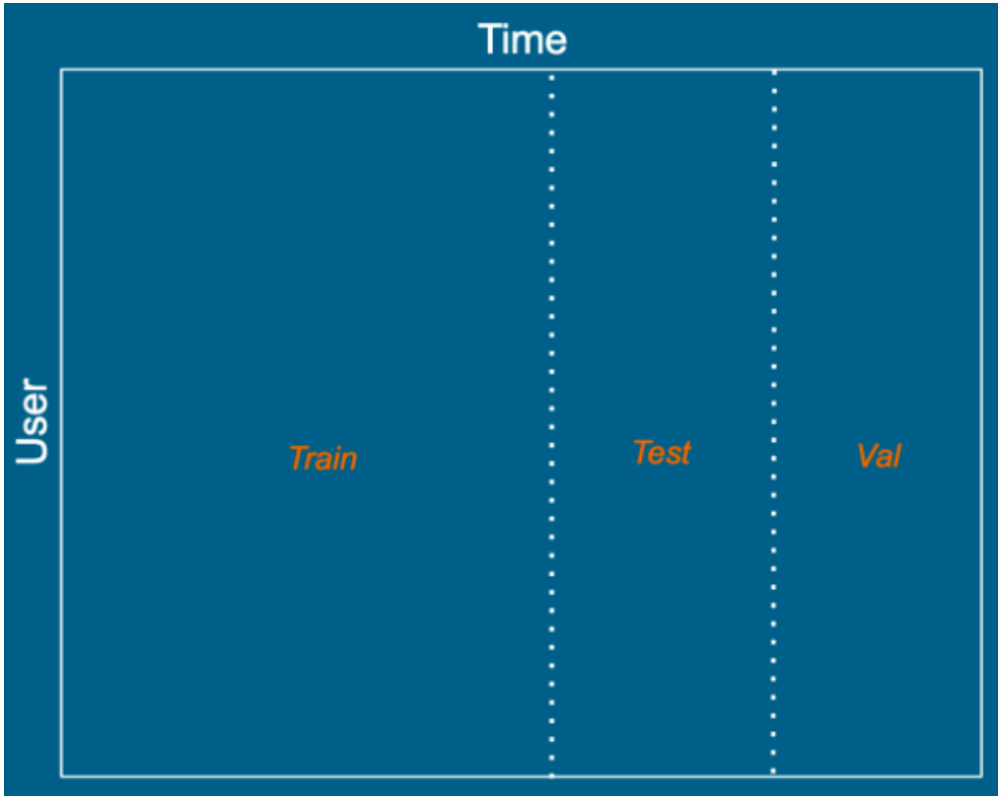
The difficulty is how do you get a system in the first place?



- Leaving out items: doesn't works (?)
- Leaving out users: kind of works? (?)

BIG FINISH: *Evaluating the Recommender*

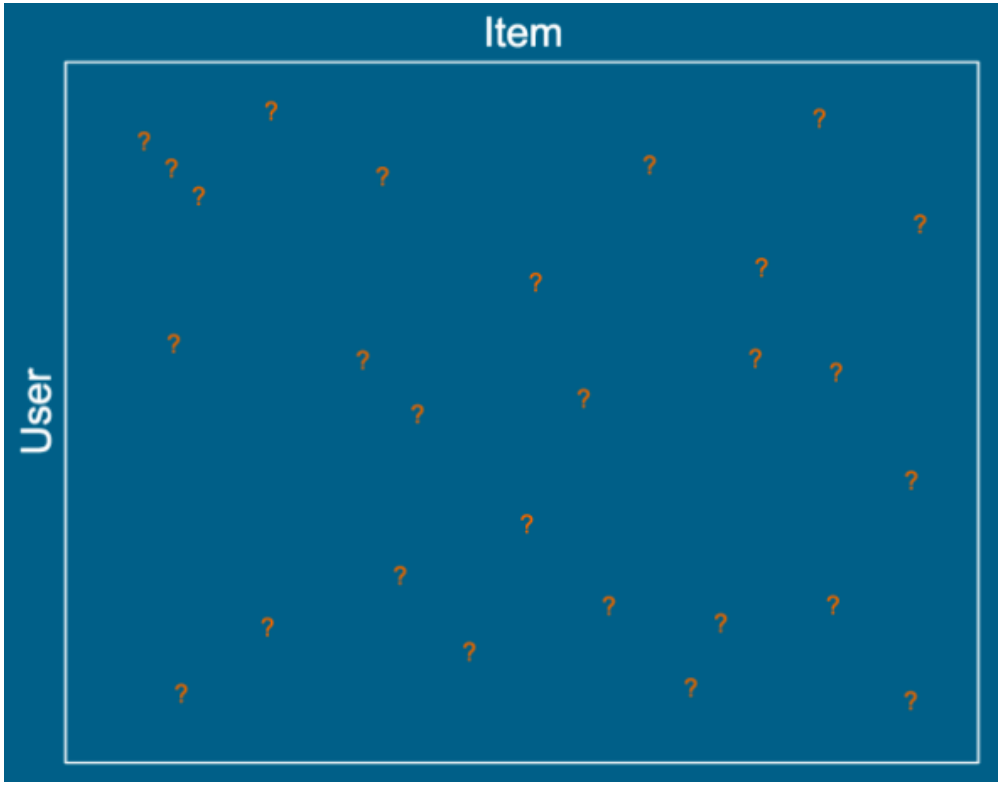
The difficulty is how do you get a system in the first place?



- Leaving out items: doesn't works (?)
- Leaving out users: kind of works? (?)
- Scoring new users/itmes: a little late... (?)
  - User/Items Cold-Start

BIG FINISH: *Evaluating the Recommender*

The difficulty is how do you get a system in the first place?

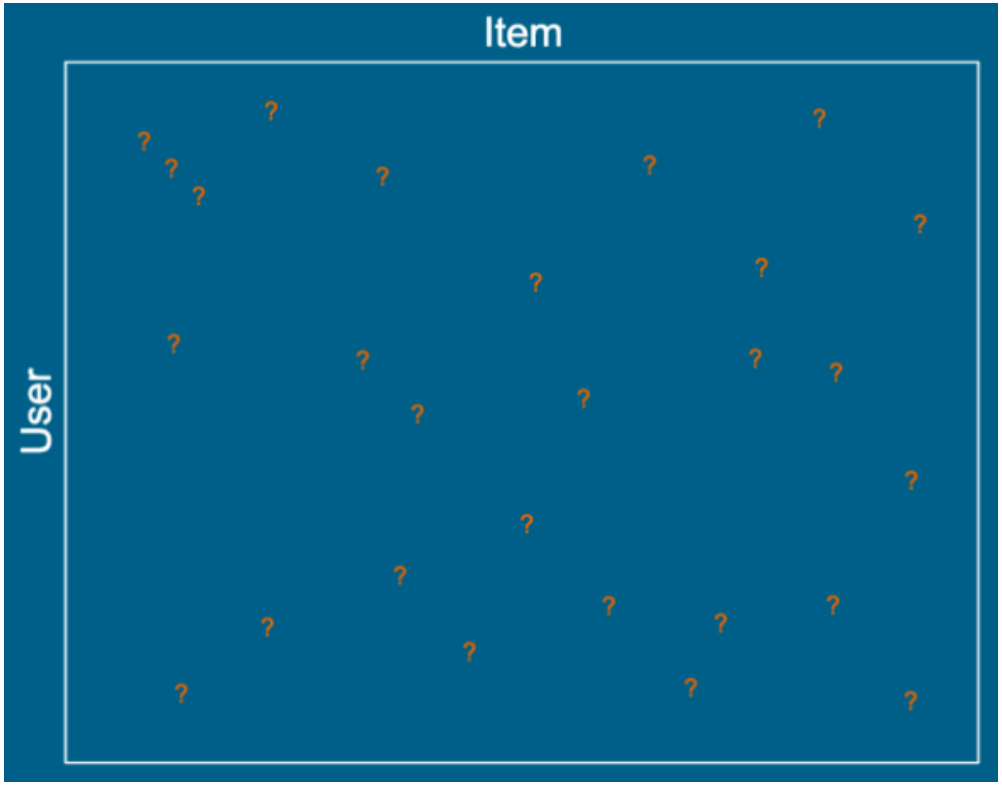


- Scoring random user-items: train/test-ish (?)

- Leaving out items: doesn't works (?)
- Leaving out users: kind of works? (?)
- Scoring new users/itmes: a little late... (?)
  - User/Items Cold-Start

BIG FINISH: *Evaluating the Recommender*

The difficulty is how do you get a system in the first place?

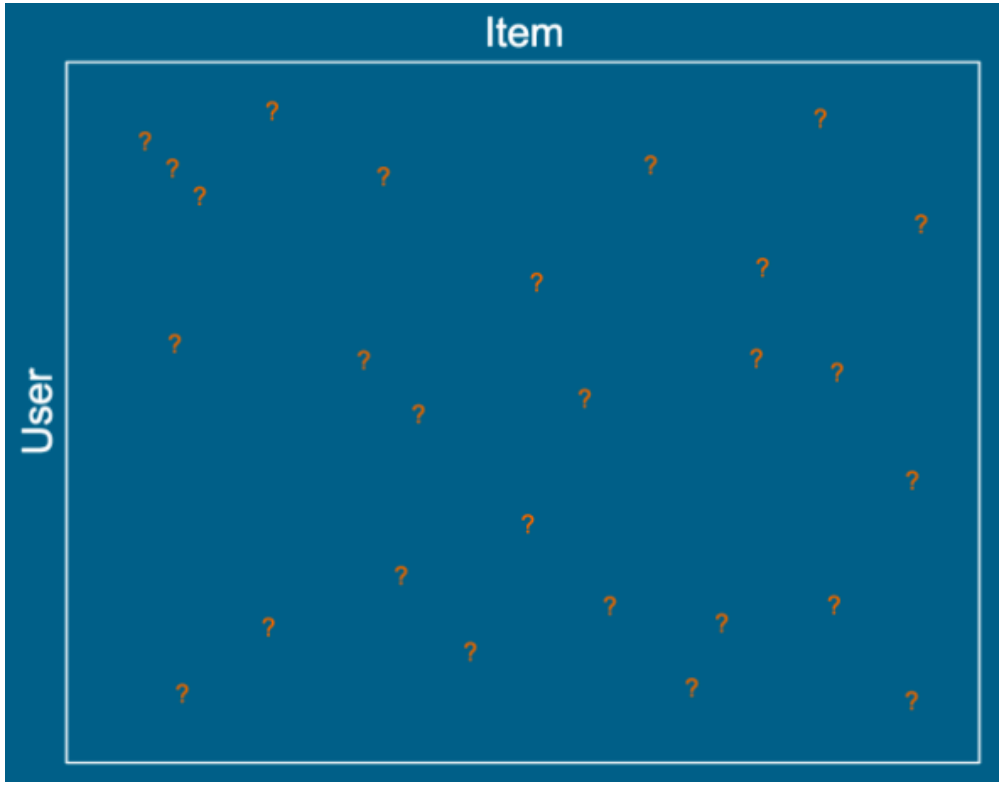


- Scoring random user-items: train/test-ish (?)
- Amazon really cares about your **buying**, not rating

- Leaving out items: doesn't works (?)
- Leaving out users: kind of works? (?)
- Scoring new users/itmes: a little late... (?)
  - User/Items Cold-Start

BIG FINISH: *Evaluating the Recommender*

The difficulty is how do you get a system in the first place?



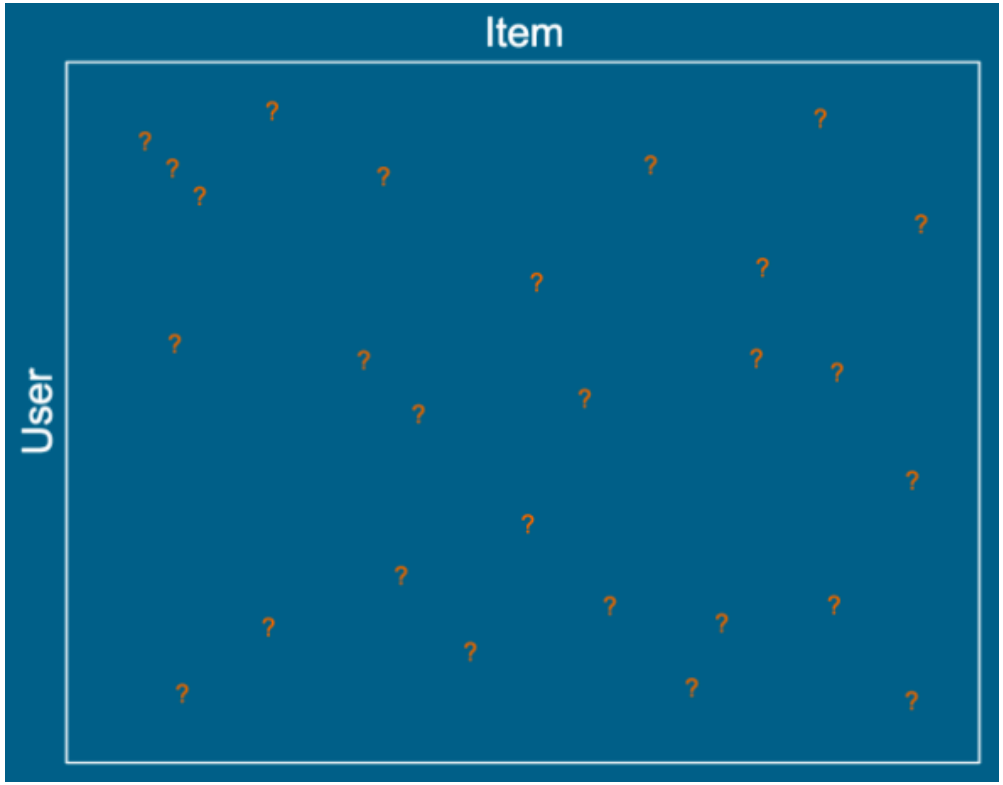
- Scoring random user-items: train/test-ish (?)
- Amazon really cares about your **buying**, not rating

  and what you want and your scores may be kinda different

- Leaving out items: doesn't works (?)
- Leaving out users: kind of works? (?)
- Scoring new users/itmes: a little late... (?)
  - User/Items Cold-Start

BIG FINISH: *Evaluating the Recommender*

The difficulty is how do you get a system in the first place?



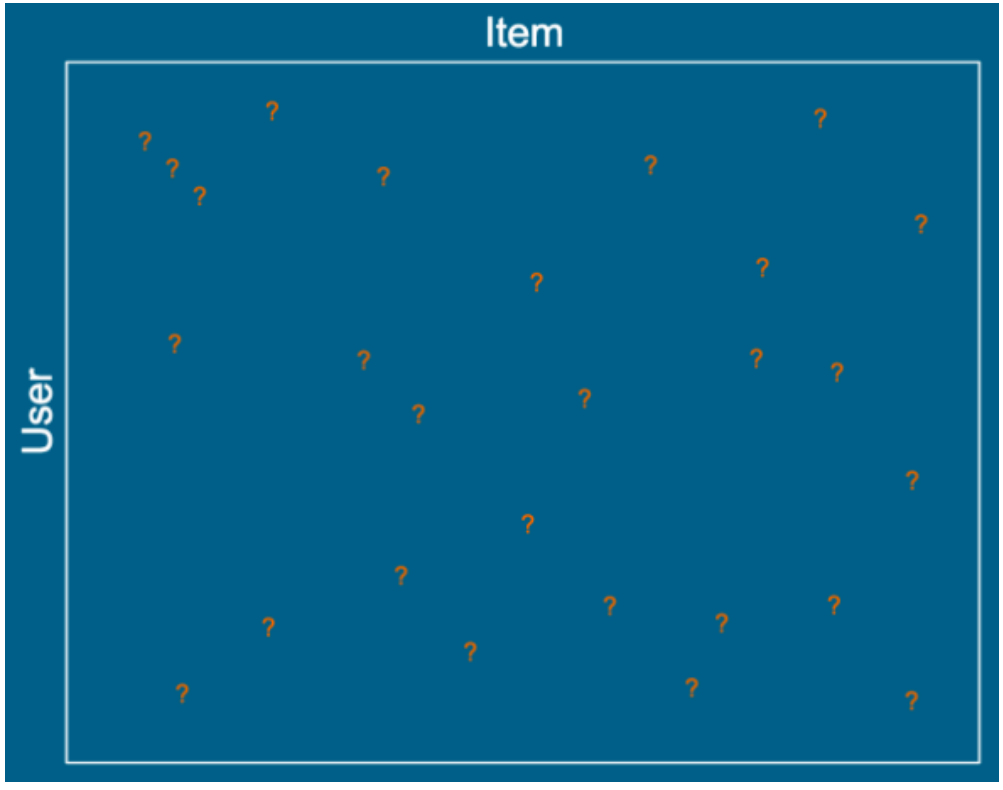
- Scoring random user-items: train/test-ish (?)
- Amazon really cares about your **buying**, not rating

  and what you want and your scores may be kinda different
- Low scores aren't recommended so you shouldn't test on them

- Leaving out items: doesn't works (?)
- Leaving out users: kind of works? (?)
- Scoring new users/itmes: a little late... (?)
  - User/Items Cold-Start

BIG FINISH: *Evaluating the Recommender*

The difficulty is how do you get a system in the first place?



- Leaving out items: doesn't works (?)
- Leaving out users: kind of works? (?)
- Scoring new users/itmes: a little late... (?)
  - User/Items Cold-Start

- Scoring random user-items: train/test-ish (?)
- Amazon really cares about your **buying**, not rating

  and what you want and your scores may be kinda different
- Low scores aren't recommended so you shouldn't test on them
- You don't have the items and scores that you should test on... that's the point: catch twenty-two...

BIG FINISH: *Evaluating the Recommender*

The difficulty is how do you get a system in the first place?

So many challenges...
A/B testing, anyone?

- Leaving out items: doesn't works (?)
- Leaving out users: kind of works? (?)
- Scoring new users/itmes: a little late... (?)
  - User/Items Cold-Start

- Scoring random user-items: train/test-ish (?)
- Amazon really cares about your **buying**, not rating

  and what you want and your scores may be kinda different
- Low scores aren't recommended so you shouldn't test on them
- You don't have the items and scores that you should test on... that's the point: catch twenty-two...

bleh