

# Recommender Systems






Sewoong Oh

CSE/STAT 416

University of Washington

# Personalization is a successful use of learning from data

- Facebook advertisements from browsing history
- Amazon, YouTube, Netflix recommendations from user choices
- Input: user preferences (or activities)
- Goal: select (a small set of) items the user will like

					
User 1	5				3
User 2		2		4	
User 3			3		
User 4	1				
User 5			4		
User 6		5			2

- Challenge: sparsity
- Key idea: **collaborative filtering**  
a user might like something, if similar users liked it

# Challenges in recommender systems

- Some feedback are implicit
  - explicit feedback: rating, purchase history, ranking
  - Implicit feedback: browsing history, TV viewing pattern
    - Implicit feedback requires pre-processing of data such as time spent, clicked, interval, etc.
- We seek diversity which is not easy to impose because users are multifaceted
  - A person with Linear Algebra textbook does not need another one, but Top-k recommendation might stick to k linear algebra textbooks
  - We don't want to recommend just Marvel movies
- Cold-start is hard
  - Recommendations for new user/movie with no data is hard
  - Need to use additional features/contexts (Netflix 20 questions)

# Challenges in recommender systems

- Interests change over time, but dynamic models are hard to train
  - Users preferences change over time
  - Movies perception changes over time
- Given millions of users and hundreds of thousands of movies, we need a scalable (i.e. fast) algorithm
  - We need to exploit that data is sparse

# Approach 0: popularity

- No personalization
- Netflix: trending now (average number of viewers)
- NY times: popular article (average views)

# Approach 1: classifier

- Train a classifier on
  - $x =$  (user features and movie features)
  - $y =$  liked (+1) or not (-1)
- Output: +1 (recommend) or -1 (do not recommend)
- Pros
  - personalized
  - flexible to include additional features like time
- Cons
  - Useful features are hard to get
  - Empirical performance not as good as **Collaborative Filtering**

# Approach 2: co-occurrences

- “People who bought X also bought ...”
- Construct a normalized co-occurrence matrix  $C$  where both rows and columns indicate items

$$C_{ij} = \frac{\# \text{ of people who bought } i \text{ and } j}{\# \text{ of people who bought } i \text{ or } j}$$

- This is a symmetric matrix:  $C_{ij} = C_{ji}$
- For a user who bought {milk, diapers}




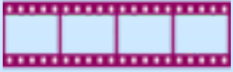














$$\text{Score}(\text{baby wipes}, user) = \frac{C_{\text{baby wipes}, \text{milk}} + C_{\text{baby wipes}, \text{diapers}}}{2}$$

- If user bought similar items, then give more score for “baby wipes”

# Approach 3: matrix factorization

- Movie recommendations
- Users watch movies and give ratings
- But each user only rates a few

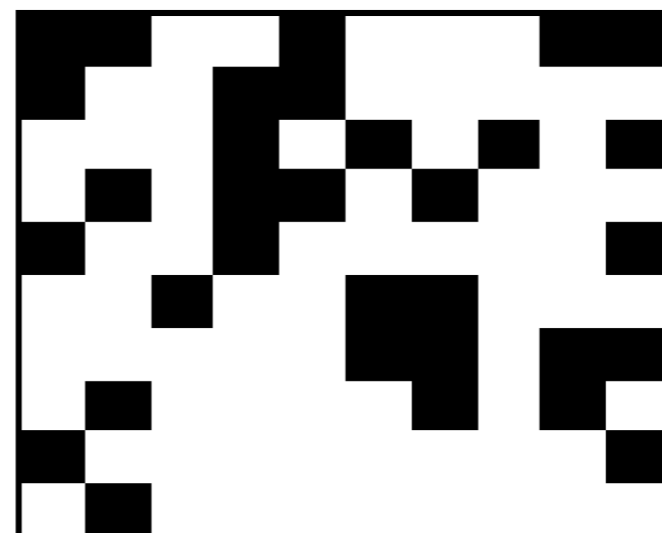
Input Data

User	Movie	Rating
		★ ★ ★ ☆ ☆
		★ ★ ★ ★ ★
		★ ★ ☆ ☆ ☆
		★ ★ ☆ ☆ ☆
		★ ★ ★ ★ ☆
		★ ☆ ☆ ☆ ☆
		★ ★ ★ ☆ ☆
		★ ★ ★ ★ ★
		★ ★ ★ ★ ☆

Input Data in a matrix form

					
User 1	5				3
User 2		2		4	
User 3			3		
User 4	1				
User 5			4		
User 6		5			2

=



# Matrix completion problem

Rating =

- Black cells indicate  $\text{Rating}(\text{user}, \text{movie})$  known
- White cells indicate  $\text{Rating}(\text{user}, \text{movie})$  unknown
- Each cell has values in  $\{1, 2, 3, 4, 5\}$
- Goal: predict missing entries



# Premise: Suppose we have $d$ types of movies

- We can describe each movie  $\mathbf{v}$  with feature vector  $\mathbf{R}_v$ 
  - How much is the movie *action, romance, drama, ...*
  - $\mathbf{R}_v = [0.3, 0.01, 1.5, \dots]$
- We can describe a user  $\mathbf{u}$  with feature vector  $\mathbf{L}_u$ 
  - How much she likes *action, romance, drama, ...*
  - $\mathbf{L}_u = [2.3, 0, 0.7, \dots]$
- Perhaps we can find such features that the rating can be predicted as the **inner product** of those two vectors
- $\text{Rating}(u,v) = 0.3*2.3 + 0.01*0 + 1.5*0.7 + \dots$
- This allows you to predict how a user will rate a movie, that she has not seen yet

# Product recommendations

- Suppose the following features have been learned, which movie should we recommend to user #3?

User ID	Feature
1	(2, 0)
2	(1, 1)
3	(0, 1)
4	(2, 1)

Call this 4x2 matrix  $L$

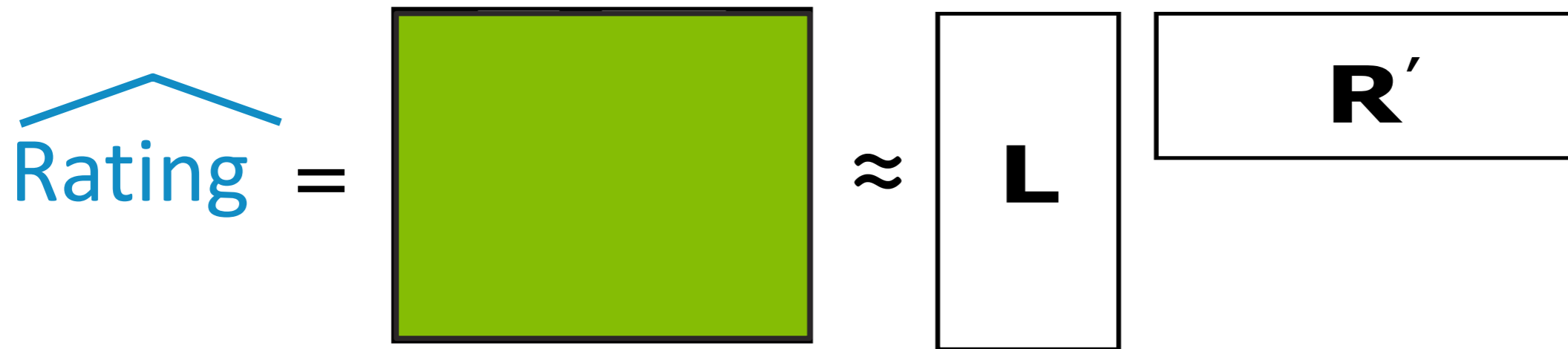
Movie ID	Feature vector
1	(3, 1)
2	(1, 2)
3	(2, 1)

Call this 3x2 matrix  $R$

- Such prediction can be computed for all (user,movie) pairs
- And be written in a matrix form:

$$\begin{array}{|c|c|c|} \hline 6 & 2 & 4 \\ \hline 4 & 3 & 3 \\ \hline 1 & 2 & 1 \\ \hline 7 & 4 & 5 \\ \hline \end{array} = \begin{array}{|c|c|} \hline 2 & 0 \\ \hline 1 & 1 \\ \hline 0 & 1 \\ \hline 2 & 1 \\ \hline \end{array} \begin{array}{|c|c|c|} \hline 3 & 1 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

# Predictions in a matrix form



The diagram illustrates the matrix equation for ratings. On the left, the word "Rating" is written in blue, with a blue line above it forming a roof-like shape. This is followed by an equals sign, a solid green square representing a matrix, an approximation symbol (≈), a white square with a black border containing the letter "L", and a white rectangle with a black border containing the letter "R'".

- Ratings matrix is the product of  $L$  and  $R$ : user feature matrix and movie feature matrix
- How do we learn the feature matrices from data?
- When we have all the ratings, then it is easy
  - PCA gives optimal factorization  $L$  and  $R$  in terms of reconstruction error
- This automatically discovers the right topics from data
- But, if we have all the ratings, we don't need to predict anything

# Matrix factorizations are not unique

- Let's say we have an exact factorization  $M = L \cdot R^T$

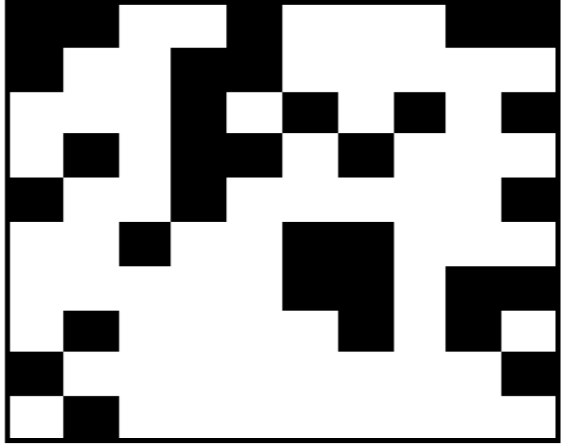


$$\begin{array}{c} M \\ \begin{array}{|c|c|c|} \hline 6 & 2 & 4 \\ \hline 4 & 3 & 3 \\ \hline 1 & 2 & 1 \\ \hline 7 & 4 & 5 \\ \hline \end{array} \end{array} = \begin{array}{c} L \\ \begin{array}{|c|c|} \hline 2 & 0 \\ \hline 1 & 1 \\ \hline 0 & 1 \\ \hline 2 & 1 \\ \hline \end{array} \end{array} \begin{array}{c} R^T \\ \begin{array}{|c|c|c|} \hline 3 & 1 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \end{array}$$

- There are infinitely many factorizations, which give the exactly same  $M$
- For example, we can scale up the user features and scale down the movie ones, so that the ratings do not change

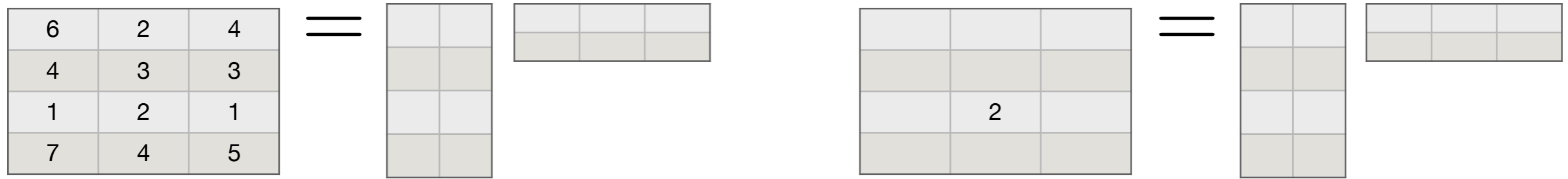
$$\begin{array}{c} M \\ \begin{array}{|c|c|c|} \hline 6 & 2 & 4 \\ \hline 4 & 3 & 3 \\ \hline 1 & 2 & 1 \\ \hline 7 & 4 & 5 \\ \hline \end{array} \end{array} = \begin{array}{c} L \\ \begin{array}{|c|c|} \hline 4 & 0 \\ \hline 2 & 2 \\ \hline 0 & 2 \\ \hline 4 & 2 \\ \hline \end{array} \end{array} \begin{array}{c} R^T \\ \begin{array}{|c|c|c|} \hline 1.5 & 0.5 & 1.0 \\ \hline 0.5 & 1.0 & 0.5 \\ \hline \end{array} \end{array}$$

- Precisely, for any invertible matrix  $Q$ ,  $(LQ, RQ^{-T})$  give the same matrix as  $(L, R)$  since  $LQ \cdot (RQ^{-T})^T = LQ \cdot Q^{-T} R = LR = M$

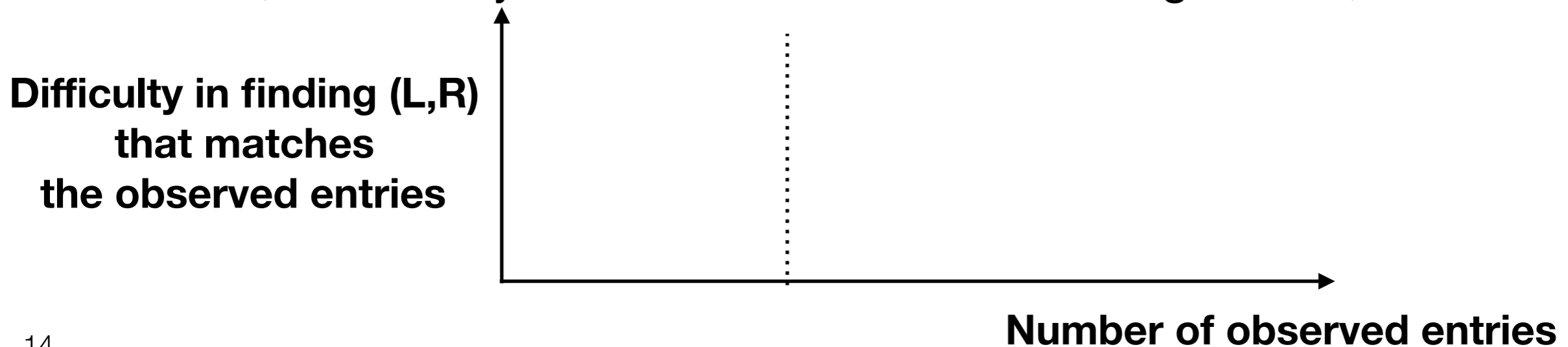
# From factorization to Matrix completion

Rating =   $\approx$   

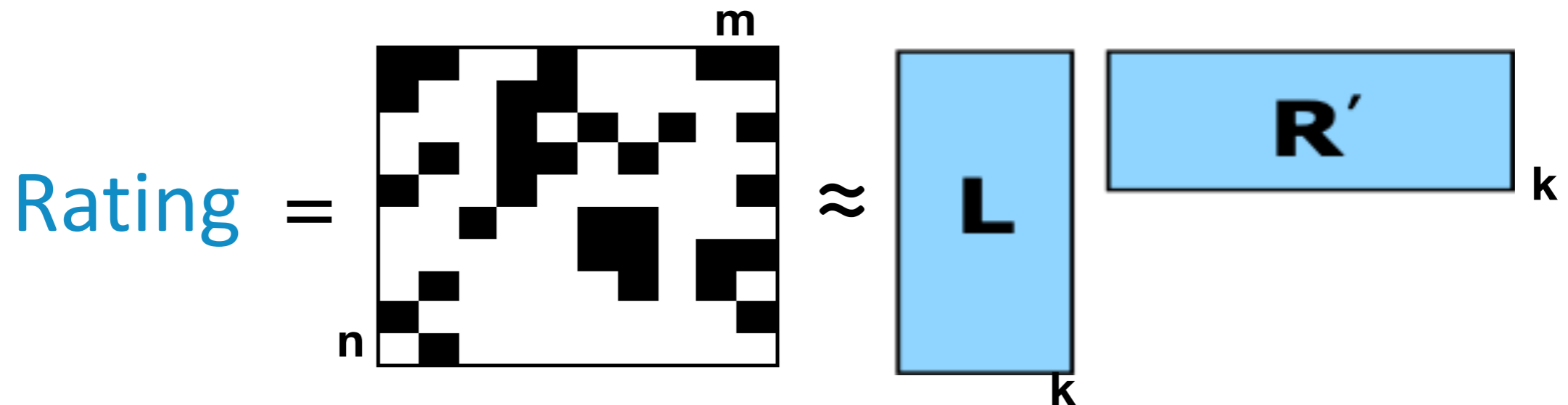
- In reality, we only have partial observations of the ratings matrix
- We **fit** the best  $\mathbf{L}$  and  $\mathbf{R}$ , to the observed ratings
- There has been many efficient algorithms to find factorization based on partial observations, a.k.a. **matrix completion problem**
- We suppose there are  $m$  movies and  $n$  users, and  $k$  topics, and the ground truth matrix  $M$  is generated by  $M = \mathbf{L}_0 \mathbf{R}_0^T$  for the form above
- No matter how many entries I observed, there are multiple choices of parameters  $(\mathbf{L}, \mathbf{R})$  that will match all the entries
  - because, if  $(\mathbf{L}, \mathbf{R})$  matches the entries, so does  $(\mathbf{L}\mathbf{Q}, \mathbf{R}\mathbf{Q}^{-T})$



- But when can we solve this problem?
- That is how many entries do we need to see, in order for our prediction to be accurate?
- One extreme: suppose we observe all entries, then
  - Any factorization methods like singular value decomposition (SVD) will provide (one of the) correct factorizations
  - And, this correct, i.e. resulting  $M = LR^T$
- Another extreme: suppose we observe one entry, then
  - It is easy to match the entry observed
  - But, most likely this is incorrect on the missing entries, i.e.  $M \neq LR^T$

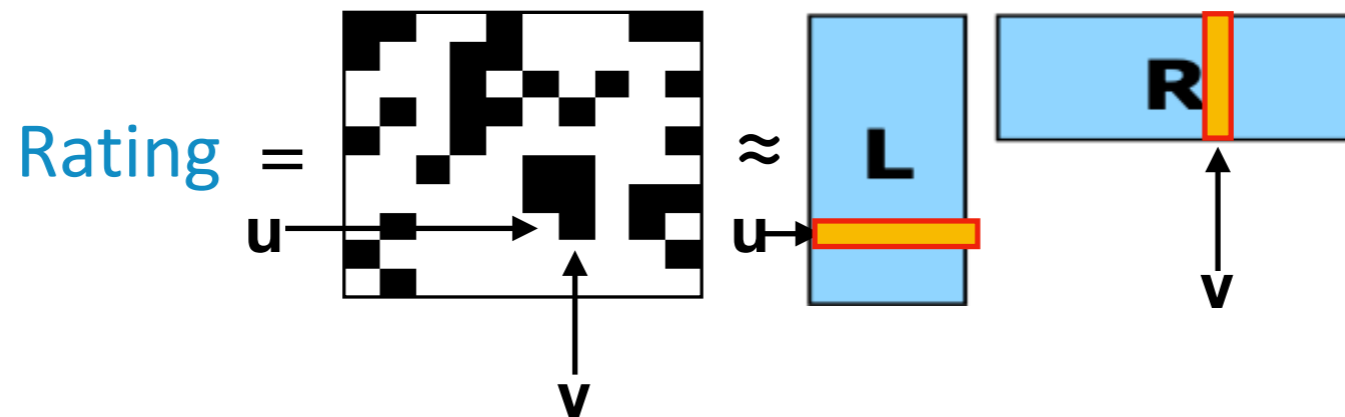


# From factorization to Matrix completion



- If there are  $m$  movies and  $n$  users, and  $k$  topics, then how many parameters do we have in our factorization model  $L$  and  $R$ ?  
degrees-of-freedom =  $k*m+k*n$
- This is also sometimes called the degrees-of-freedom in the problem.
- How many entries do we observe if we have the full matrix?
- How many entries do you think we need, to accurately reconstruct the ground truth  $L$  and  $R$  that generated the data?

# Algorithmic solution for matrix completion



- How do we write a program to find  $(L, R)$  matching the observed entries?
- Machine learning approach:
  - Write a loss function and minimize

$$\text{minimize}_{L, R} \sum_{u, v: r_{uv} \neq ?} \left( \underbrace{(LR^T)_{uv}}_{L_u^T R_v} - r_{uv} \right)^2$$

- Coordinate descent is popular in solving this optimization



# Coordinate descent

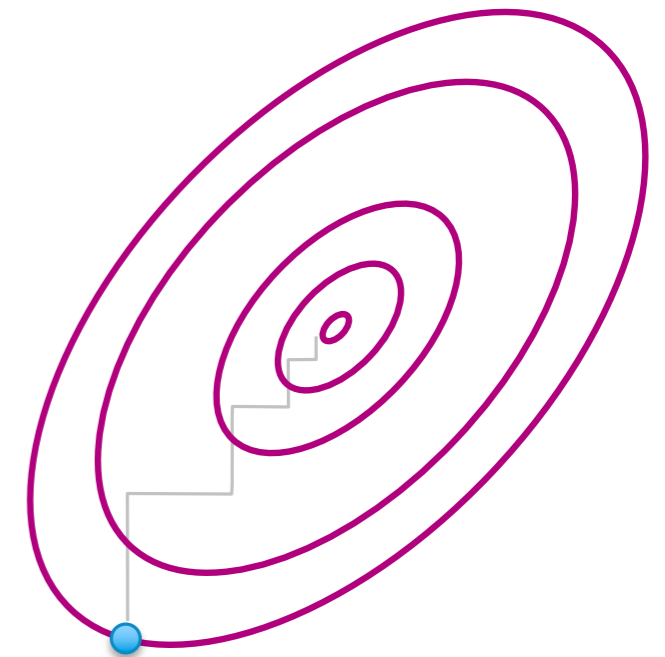
- Consider an optimization problem (in 2-dimensions for illustration purposes)

$$\text{minimize}_{w_0, w_1} \quad g(w_0, w_1)$$

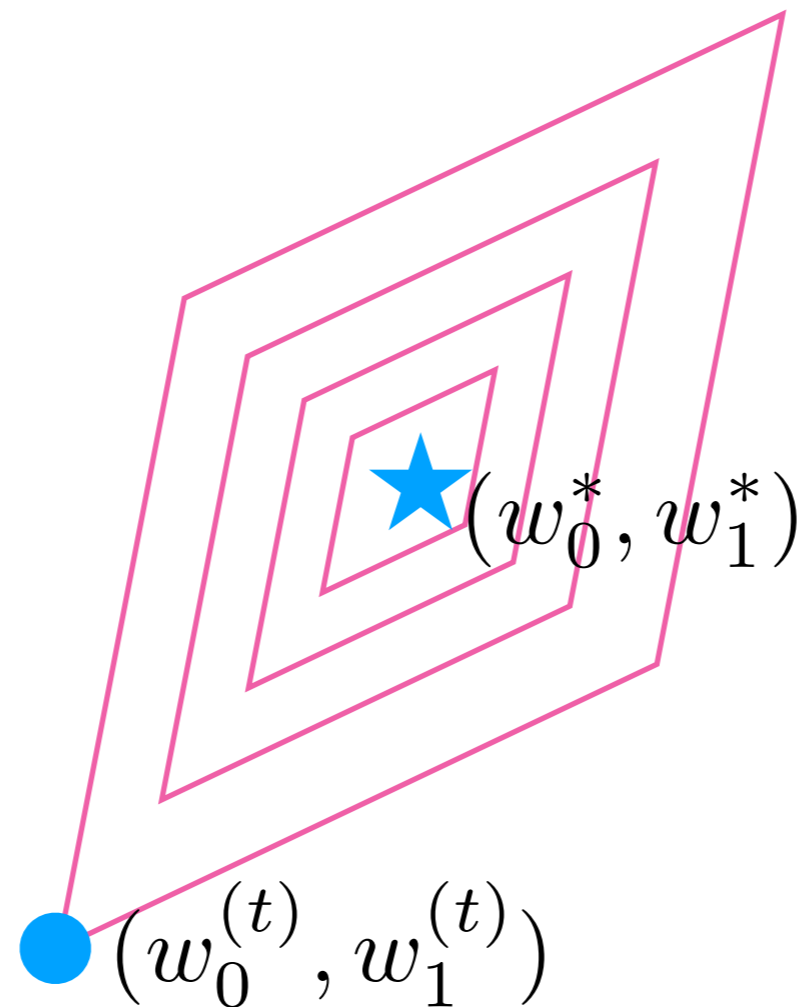
- One method is called **coordinate descent**
- Initialize  $(w_0, w_1)$  to be random or smart initialization
- While not converged, repeat
  - Pick a coordinate  $j$  in  $\{0, 1\}$  (either random, round-robin, etc.)

$$w_j \leftarrow \arg \min_{w_j} g(w_0, w_1)$$

- Main idea:
  - Minimizing over 1 coordinate is much easier
  - No need to choose step-size
  - This is guaranteed to find optimal solution, under some constraints
- When does it fail?



# Coordinate descent



- Coordinate descent successfully finds the optimal solution if  $g(\cdot)$  is **strongly convex** and **smooth**

# Coordinate descent for matrix completion

$$\text{minimize}_{L,R} \sum_{u,v:r_{uv} \neq ?} \left( \underbrace{(LR^T)_{uv}}_{L_u^T R_v} - r_{uv} \right)^2$$

- Initialize  $(L,R)$
- Repeat
  - Fix  $R$  and optimize over  $L$
  - Fix  $L$  and optimize over  $R$

- First insight:

$$\min_{L_1, \dots, L_n} \sum_{(u,v):r_{uv} \neq ?} (L_u^T R_v - r_{uv})^2$$

$$= \min_{L_1, \dots, L_n} \sum_{u=1}^n \left\{ \sum_{v:r_{uv} \neq ?} (L_u^T R_v - r_{uv})^2 \right\}$$

$$= \sum_{u=1}^n \left\{ \min_{L_u} \sum_{v:r_{uv} \neq ?} (L_u^T R_v - r_{uv})^2 \right\}$$

This only involves each row of  $L$  independently, and can be solved as a separate optimization for each row

# Coordinate descent for matrix completion

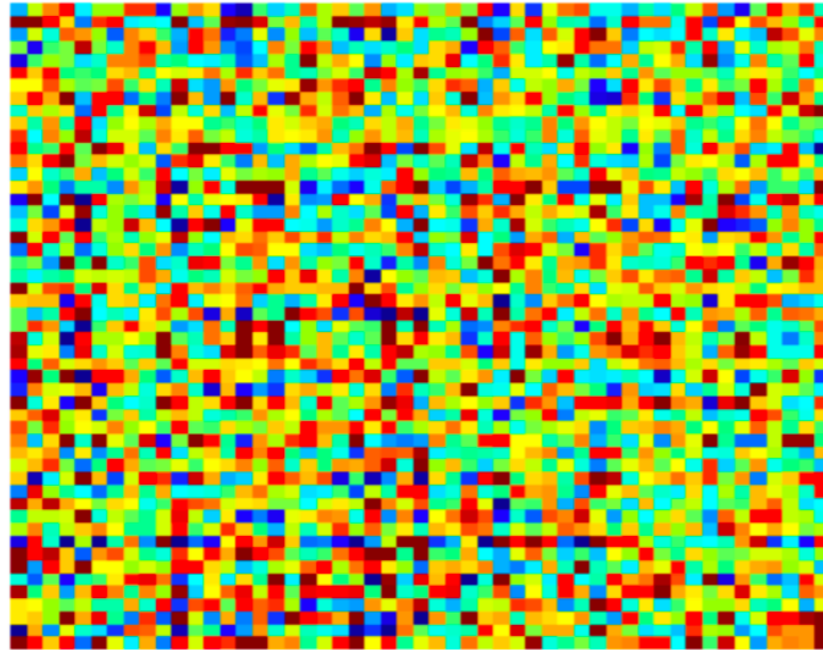
- We broke down the problem into solving multiple inner optimizations of the form:

$$\min_{L_u} \sum_{v:r_{uv} \neq ?} (L_u^T R_v - r_{uv})^2$$

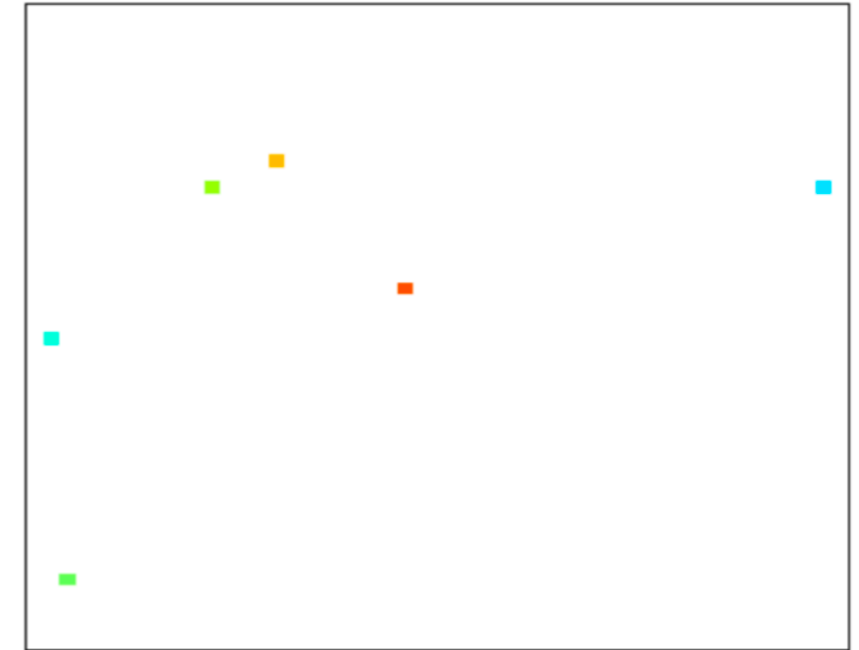
- Second insight:
  - And this is the standard linear regression with quadratic loss
  - Many efficient solvers exist + can be solved in a closed form too
-

# Example: $2000 \times 2000$ rank-8 random matrix

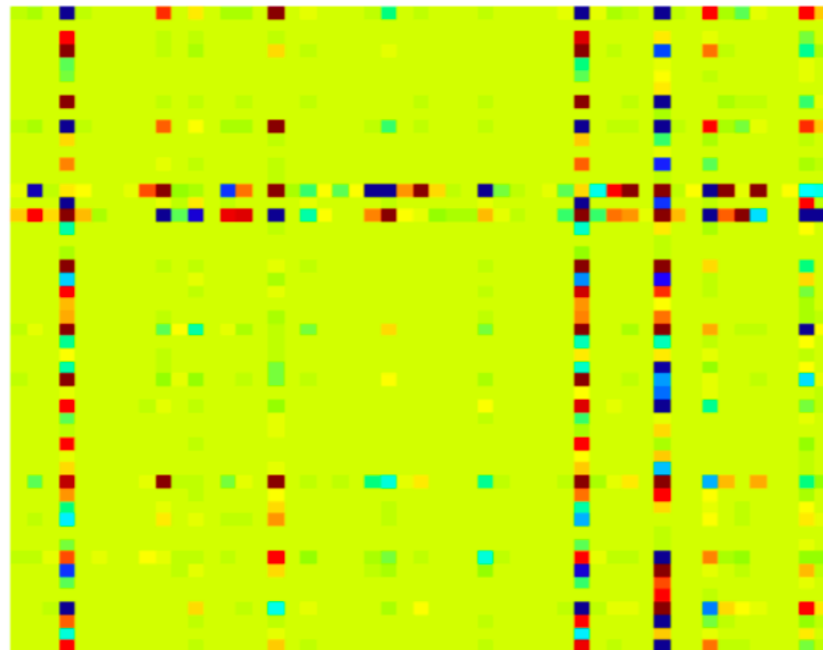
low-rank matrix  $M$



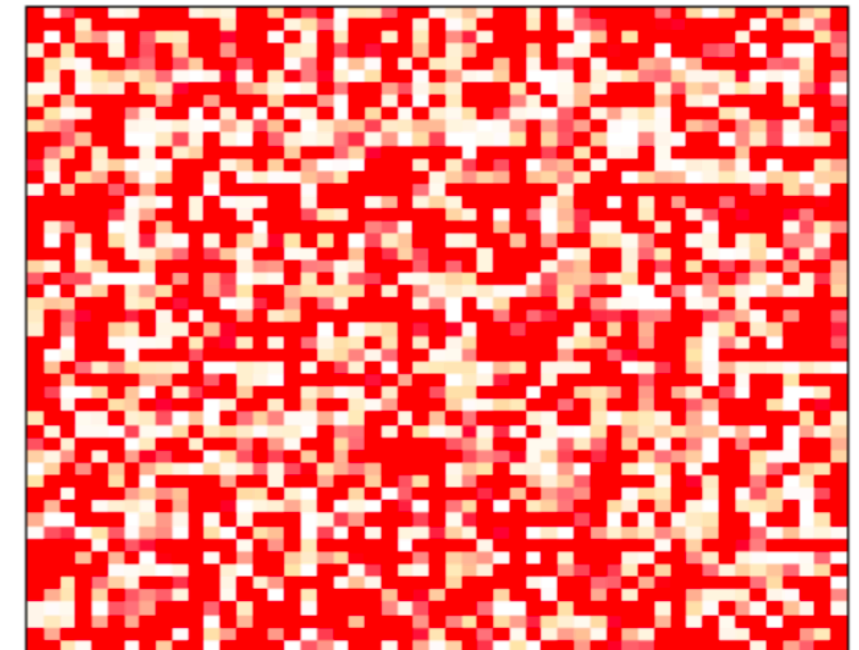
sampled matrix  $M^E$



OPTSPACE output  $\hat{M}$



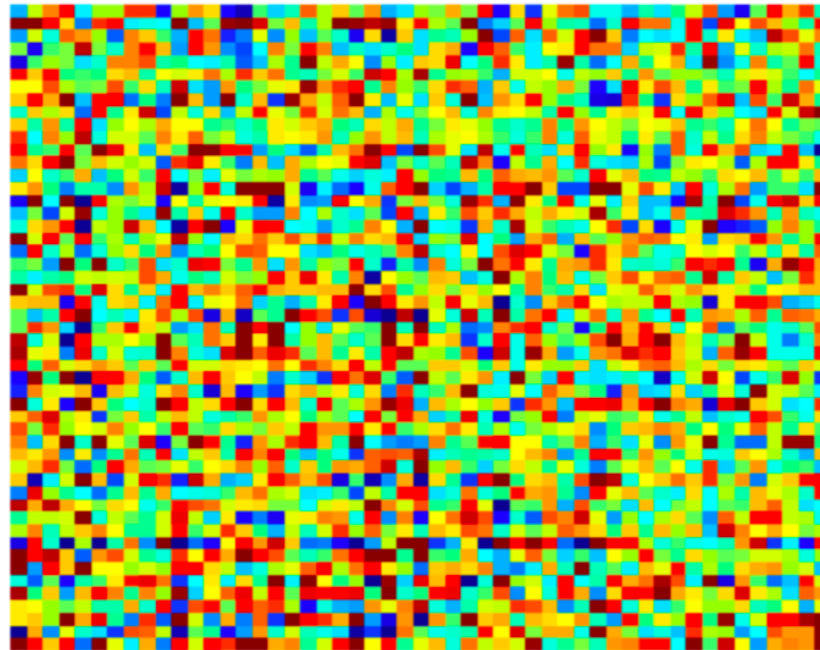
squared error  $(M - \hat{M})^2$



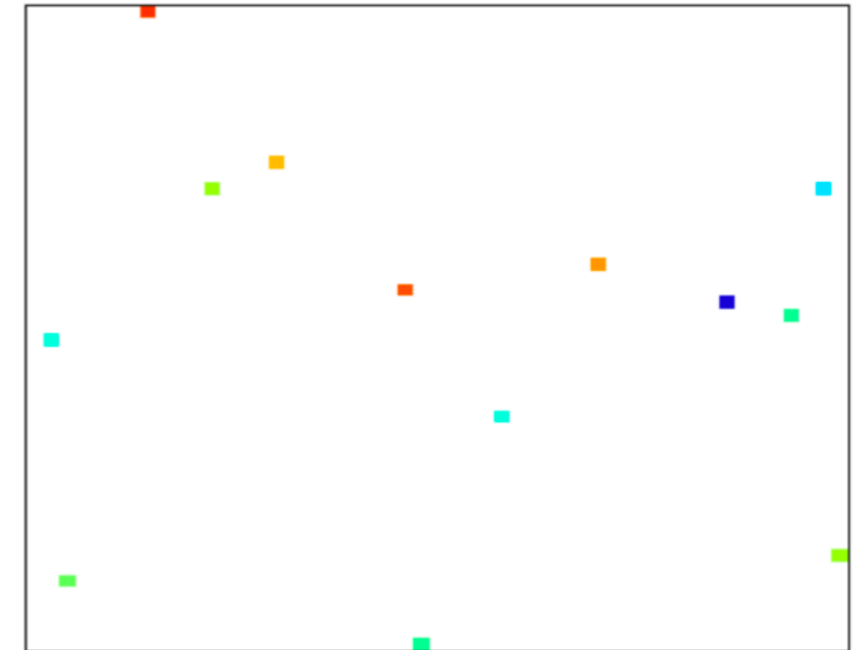
0.25% sampled

# Example: $2000 \times 2000$ rank-8 random matrix

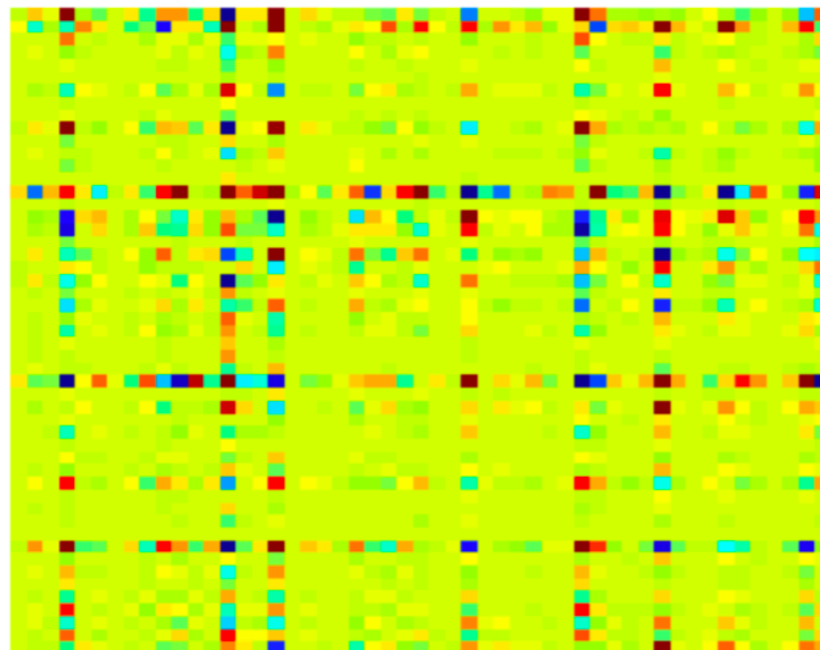
low-rank matrix  $M$



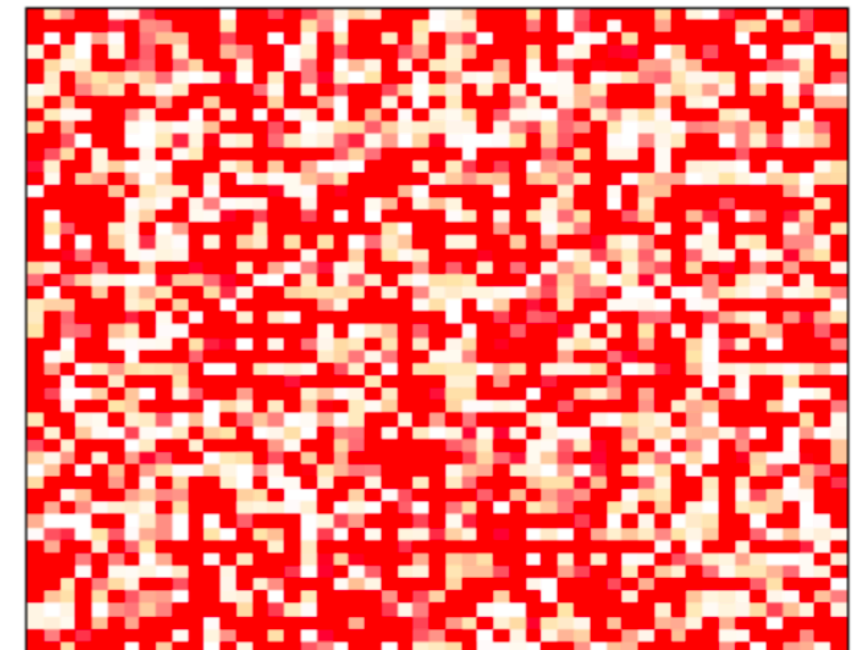
sampled matrix  $M^E$



OPTSPACE output  $\hat{M}$



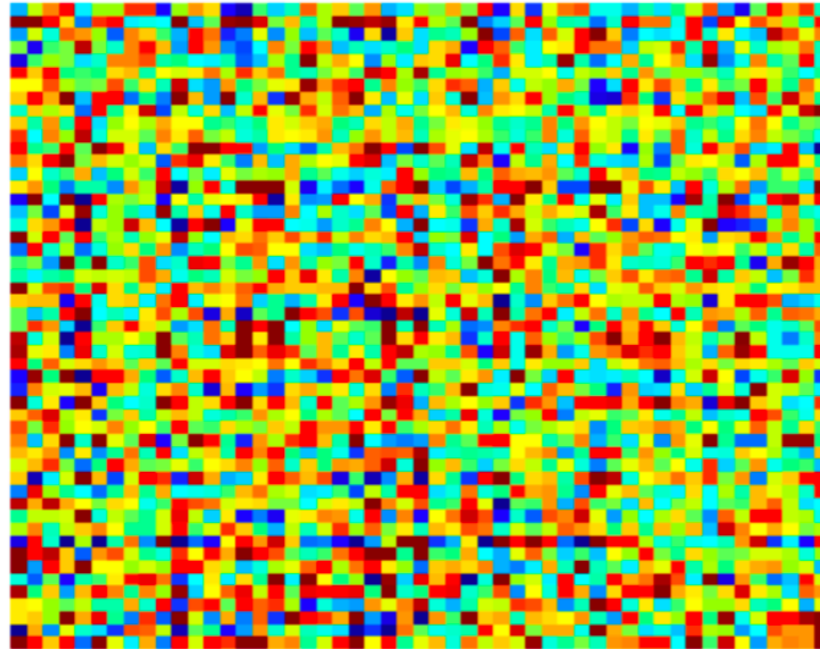
squared error  $(M - \hat{M})^2$



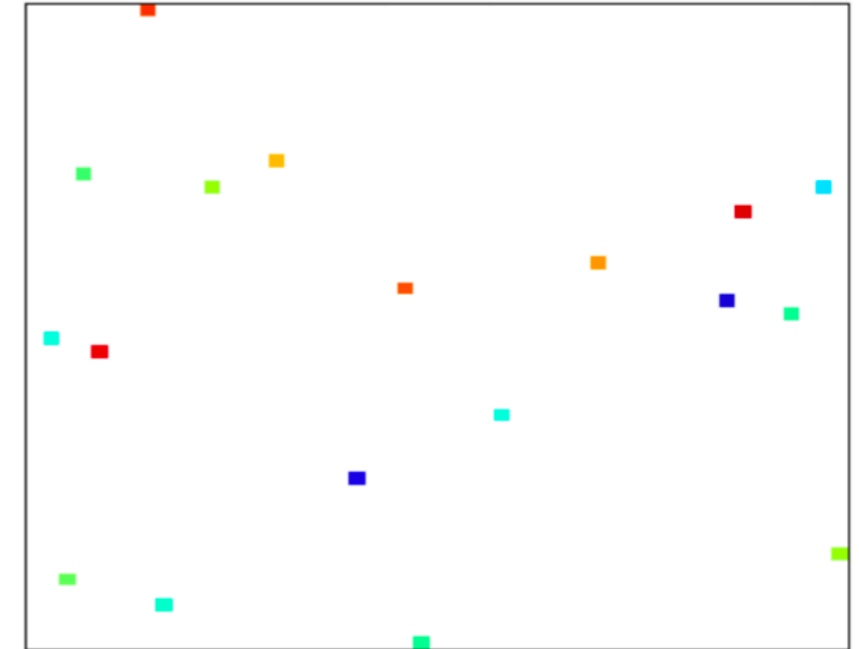
0.50% sampled

# Example: $2000 \times 2000$ rank-8 random matrix

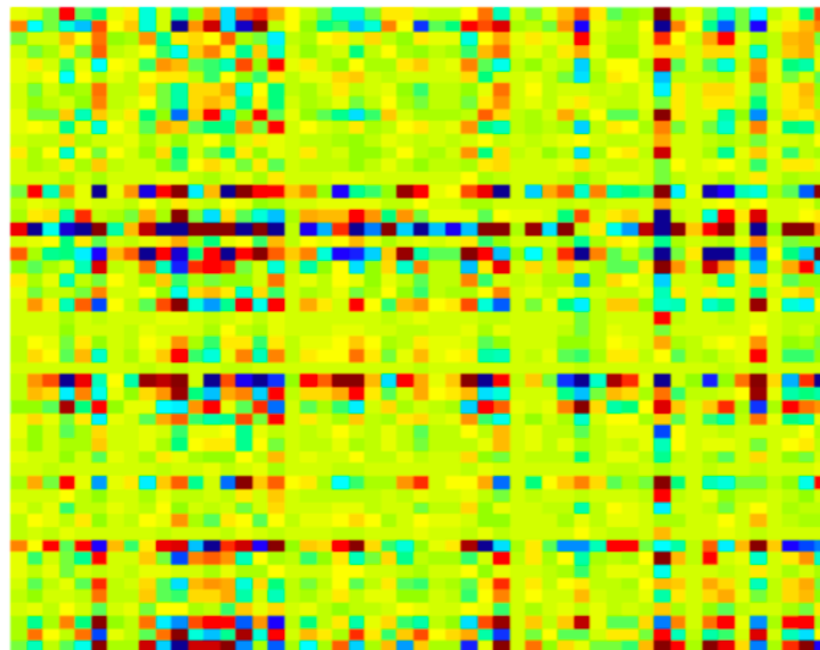
low-rank matrix  $M$



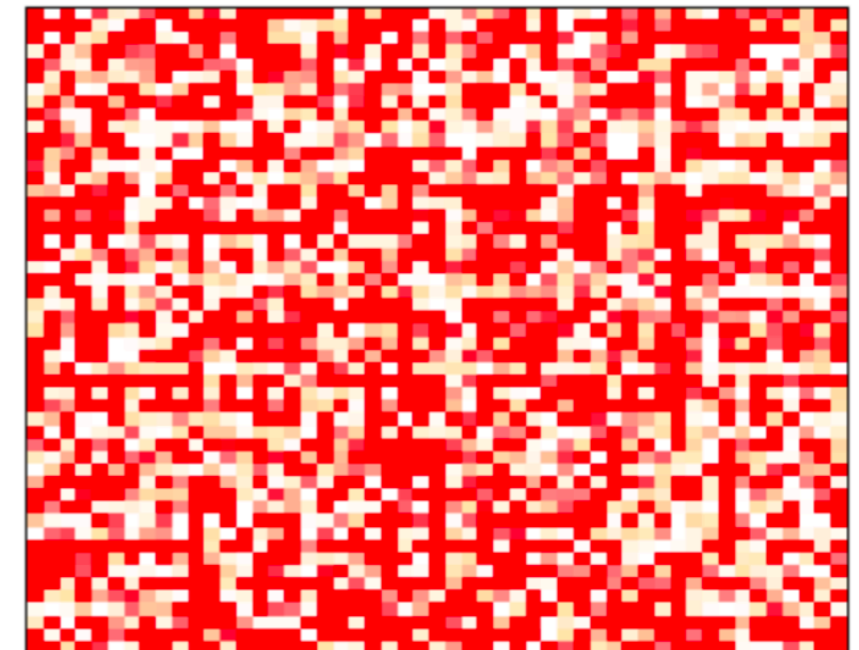
sampled matrix  $M^E$



OPTSPACE output  $\hat{M}$



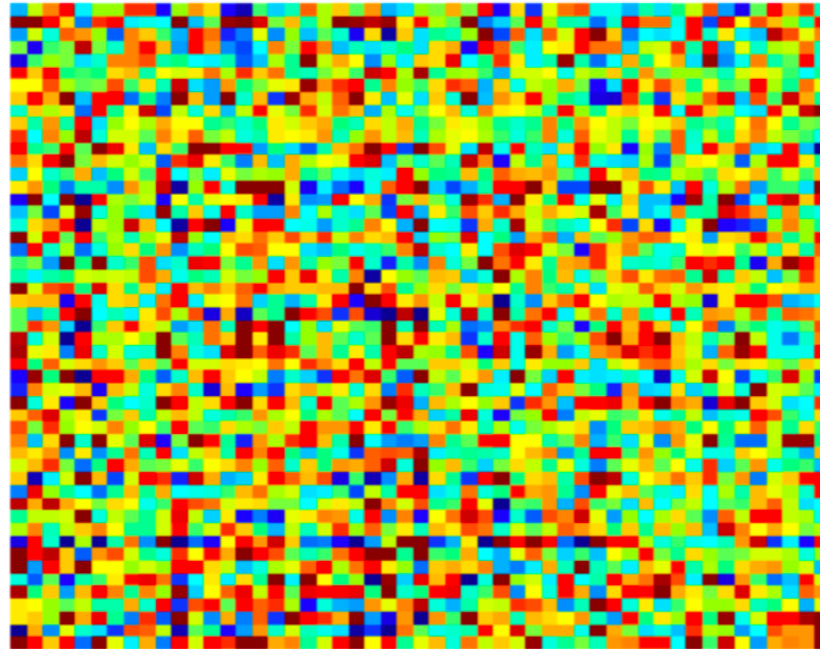
squared error  $(M - \hat{M})^2$



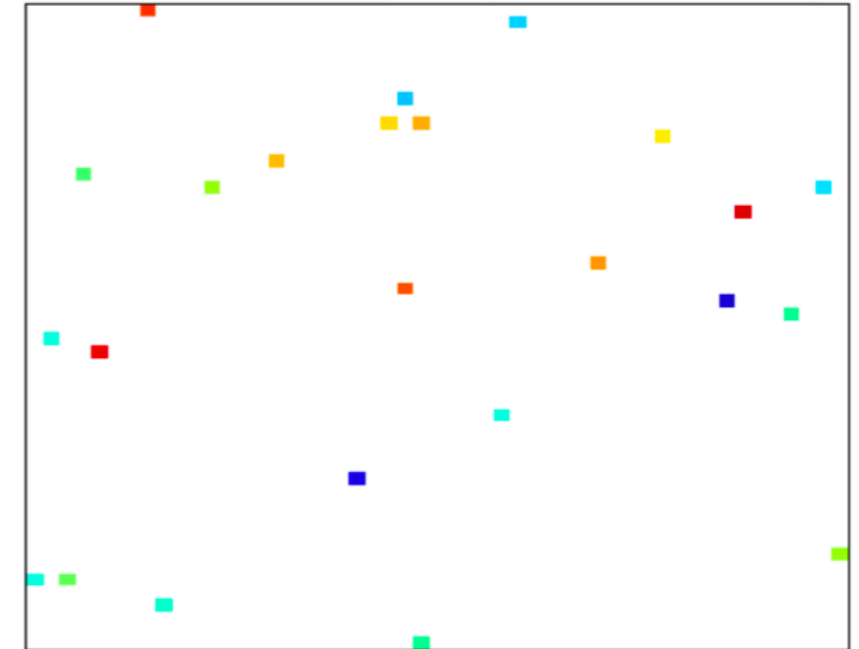
0.75% sampled

# Example: $2000 \times 2000$ rank-8 random matrix

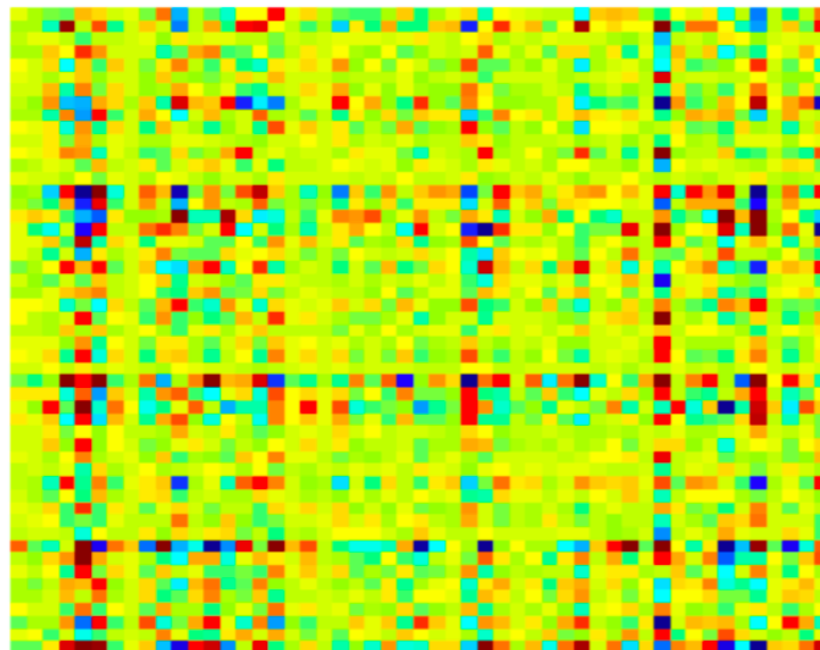
low-rank matrix  $M$



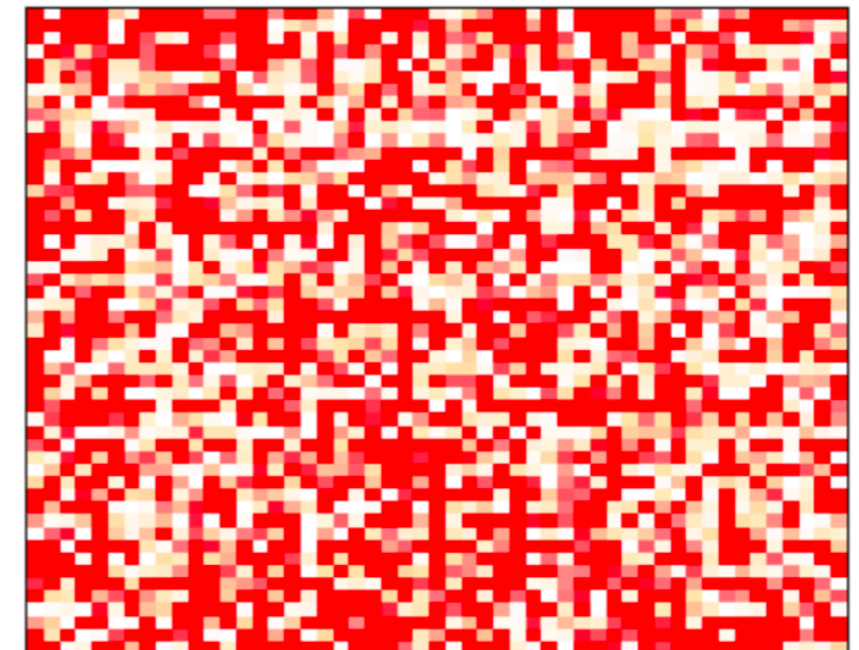
sampled matrix  $M^E$



OPTSPACE output  $\hat{M}$



squared error  $(M - \hat{M})^2$

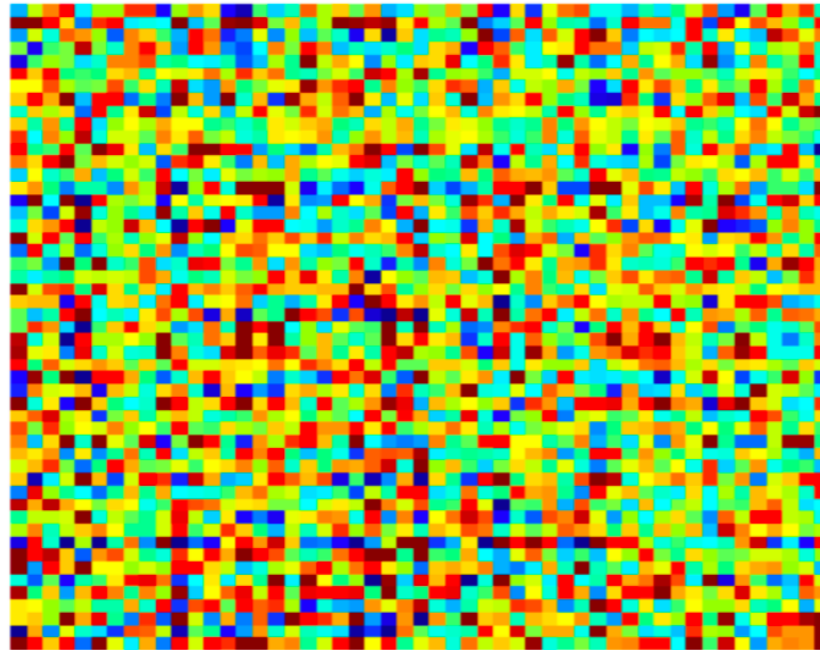


1.00% sampled

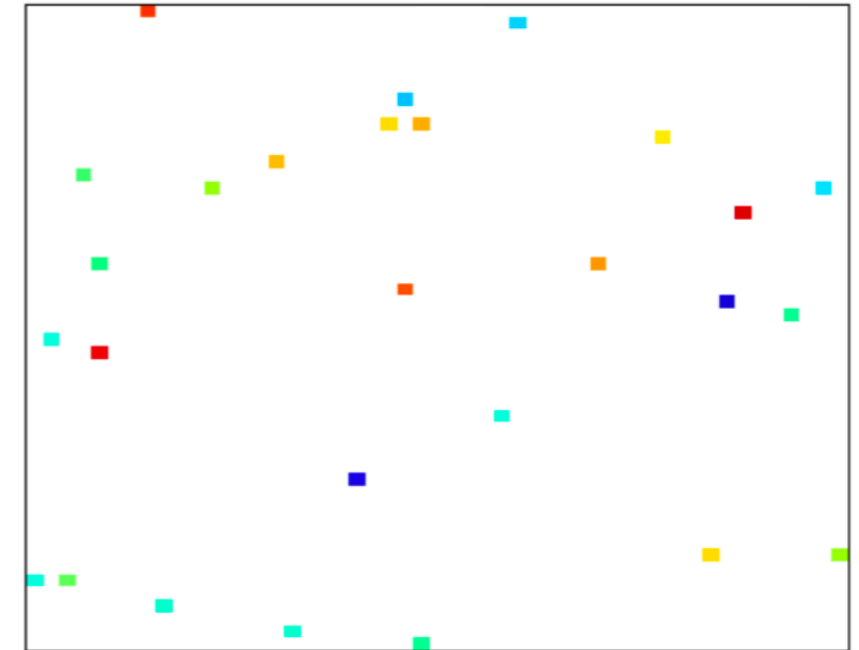


# Example: $2000 \times 2000$ rank-8 random matrix

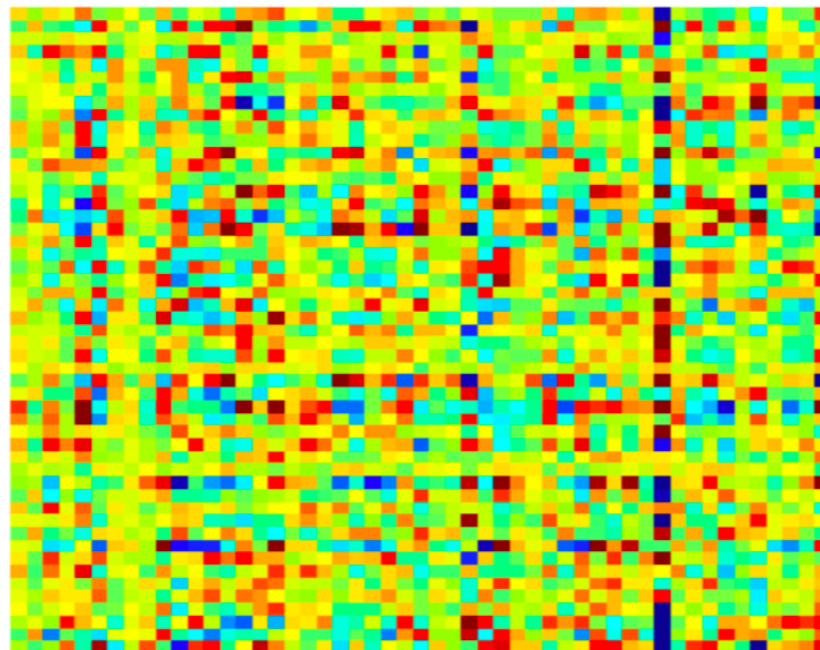
low-rank matrix  $M$



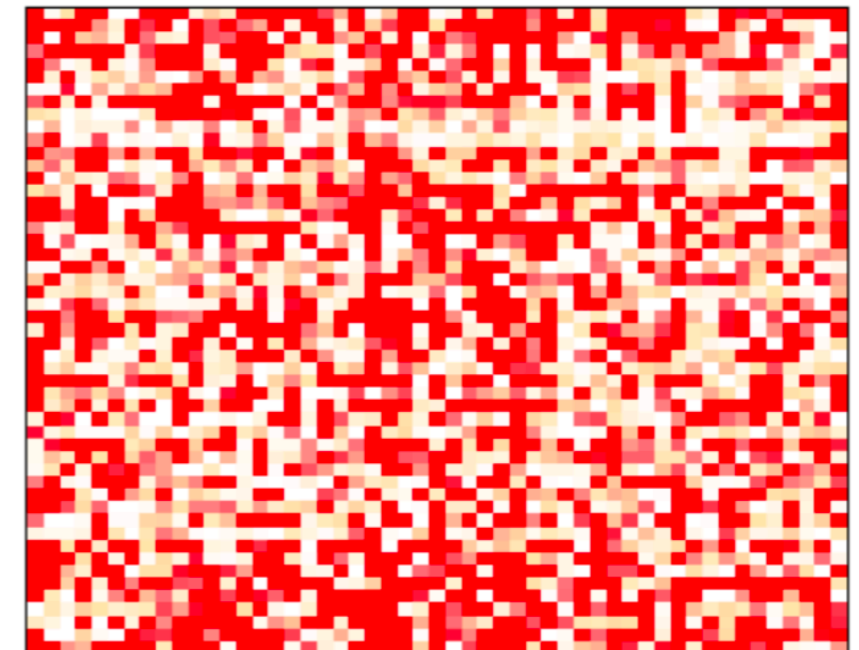
sampled matrix  $M^E$



OPTSPACE output  $\hat{M}$



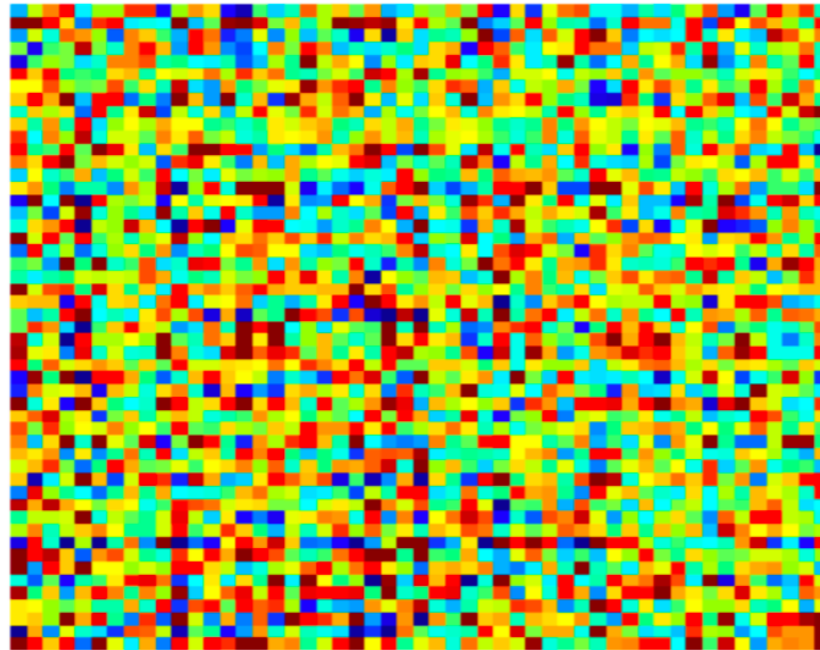
squared error  $(M - \hat{M})^2$



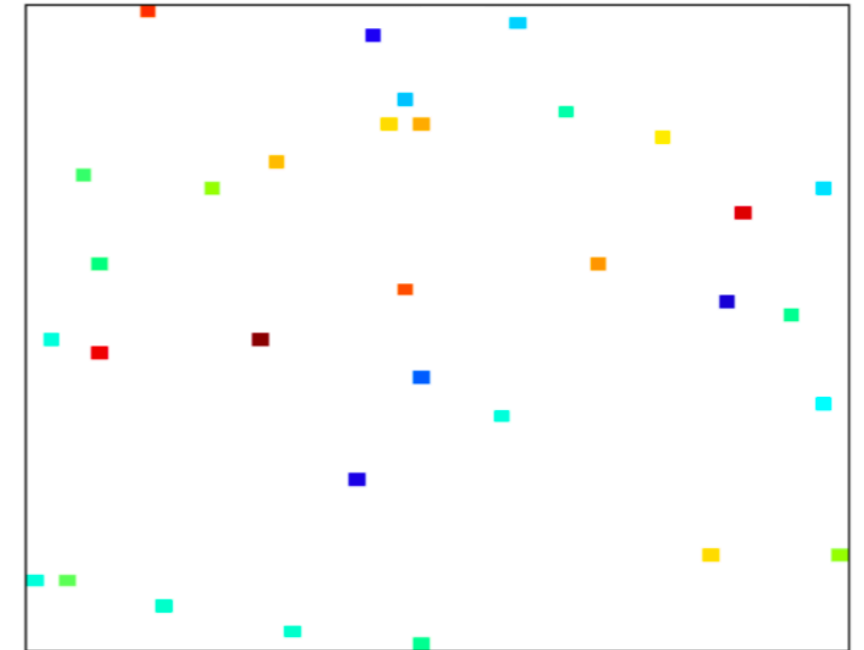
1.25% sampled

# Example: $2000 \times 2000$ rank-8 random matrix

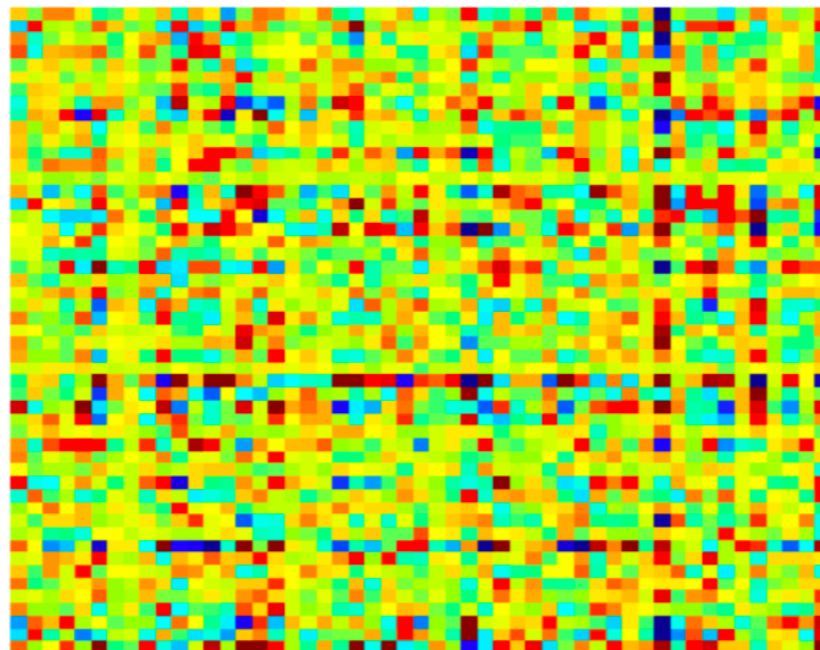
low-rank matrix  $M$



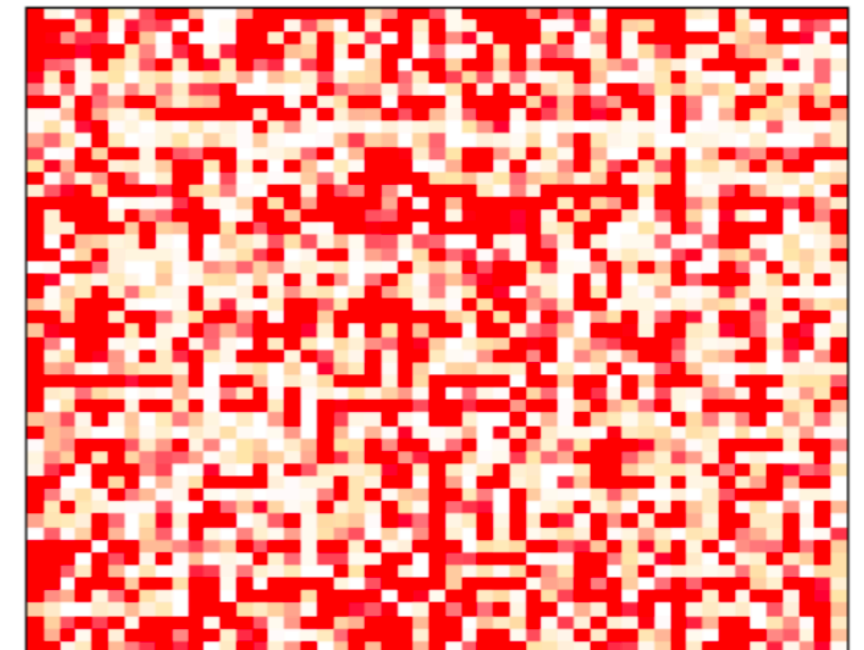
sampled matrix  $M^E$



OPTSPACE output  $\hat{M}$



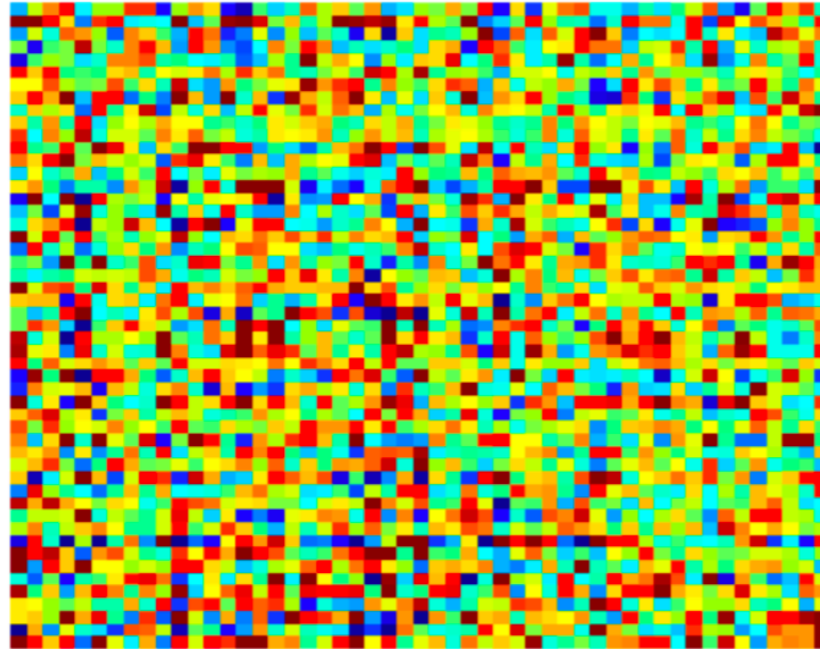
squared error  $(M - \hat{M})^2$



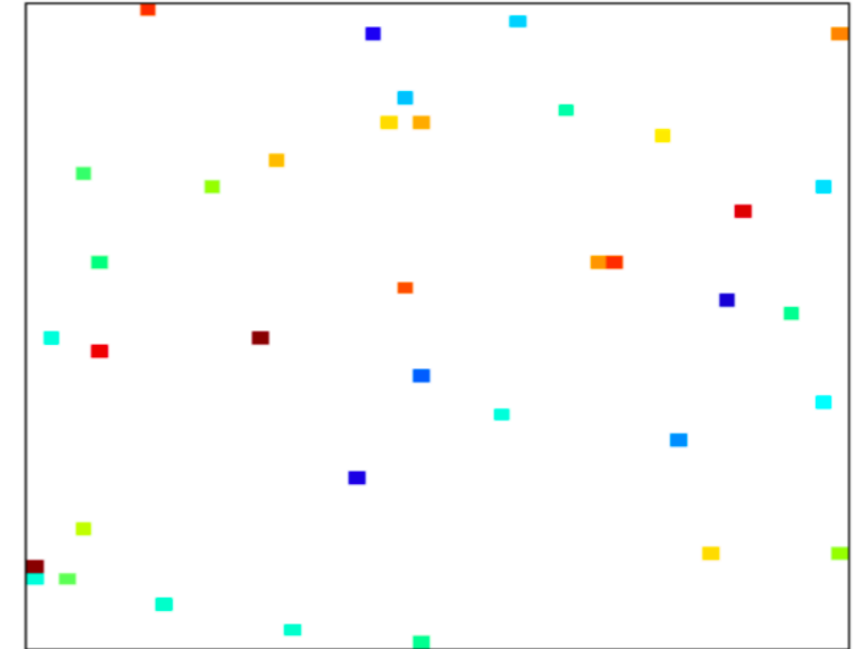
1.50% sampled

# Example: $2000 \times 2000$ rank-8 random matrix

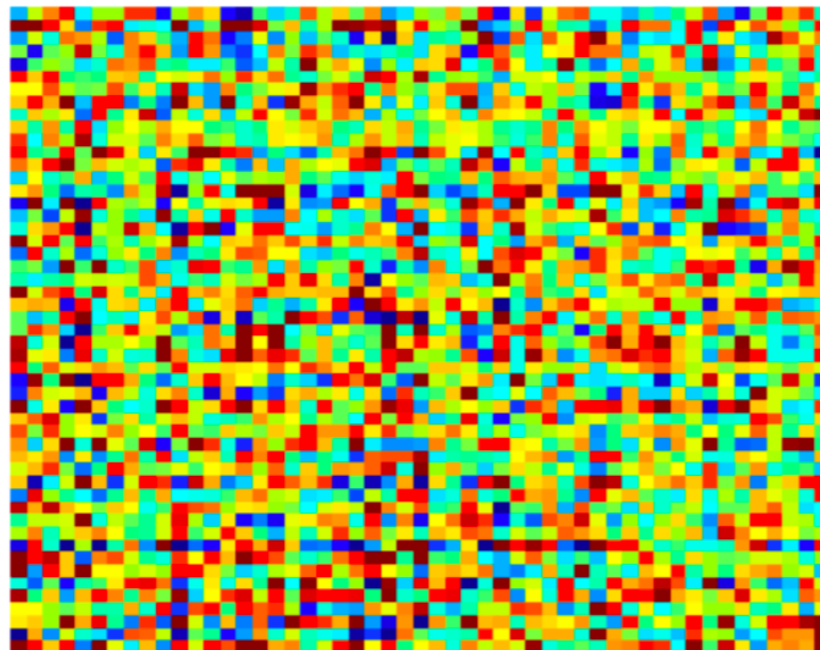
low-rank matrix  $M$



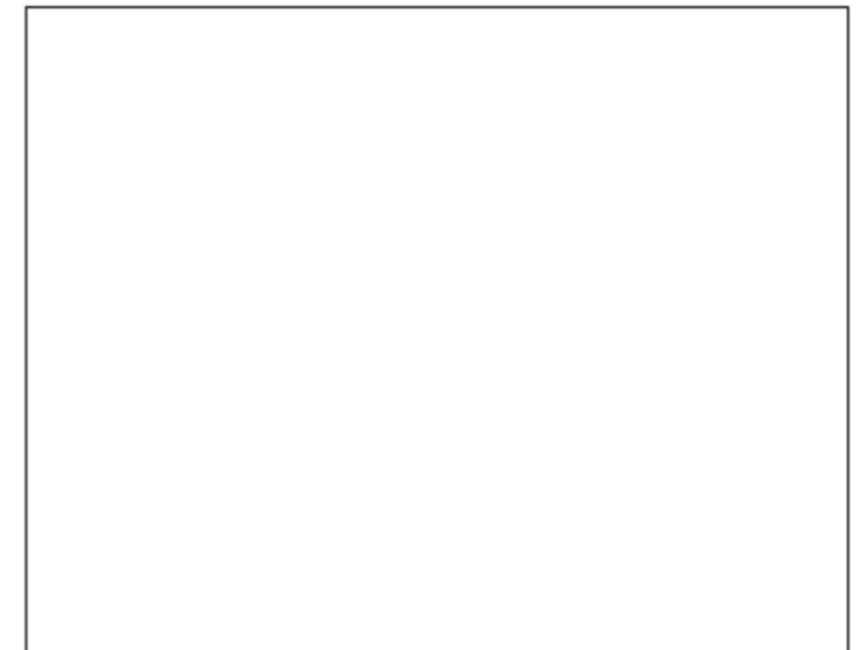
sampled matrix  $M^E$



OPTSPACE output  $\hat{M}$

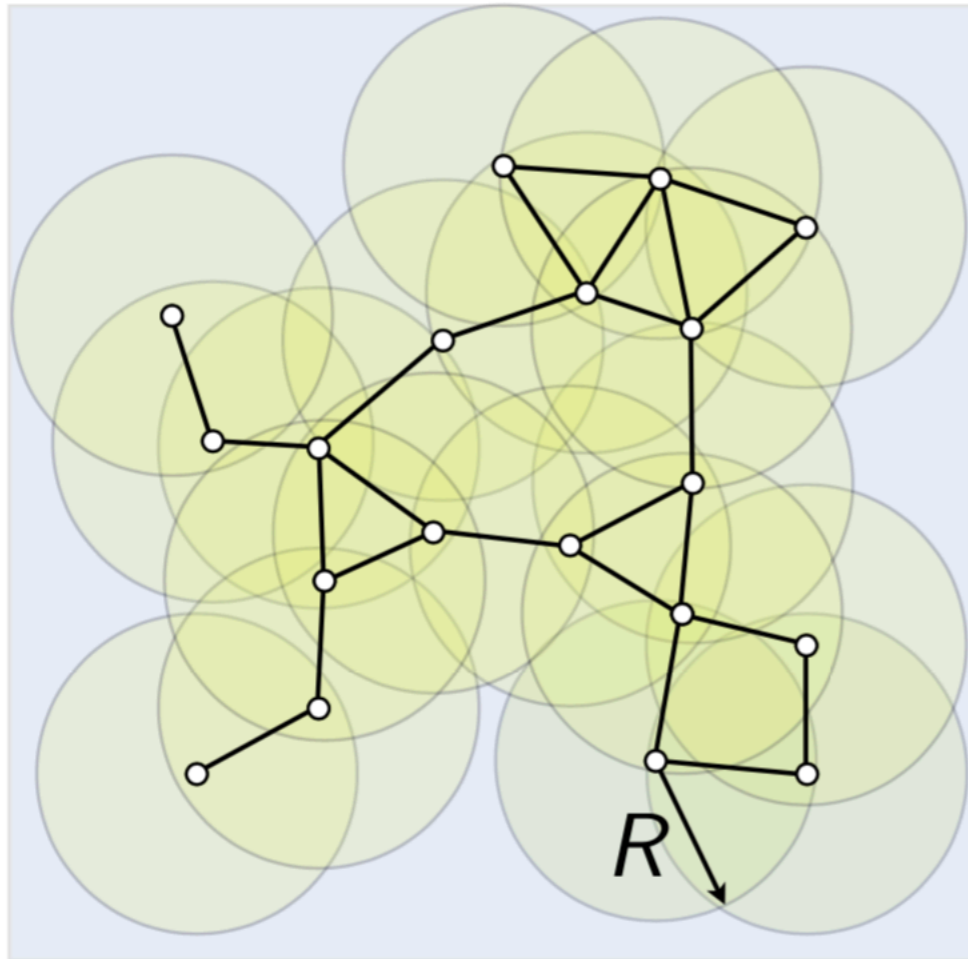


squared error  $(M - \hat{M})^2$

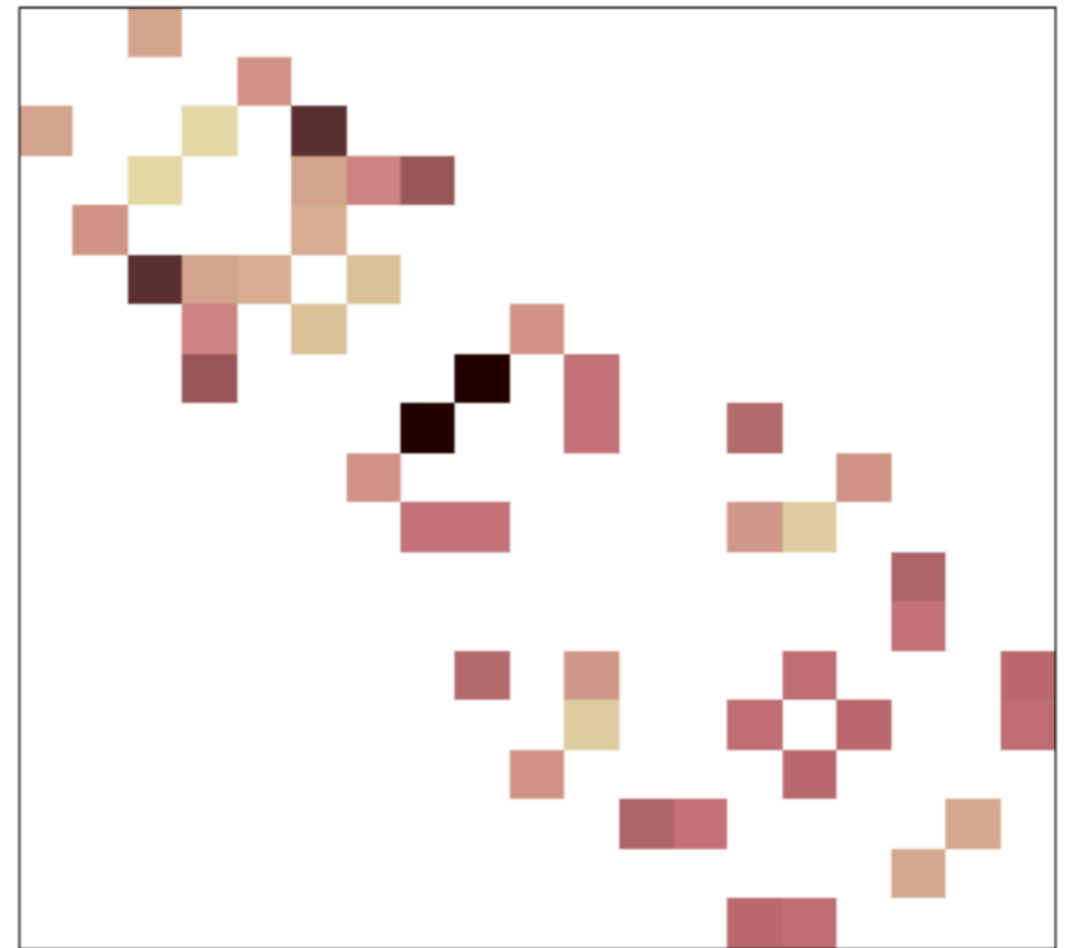


1.75% sampled

# Application: localization

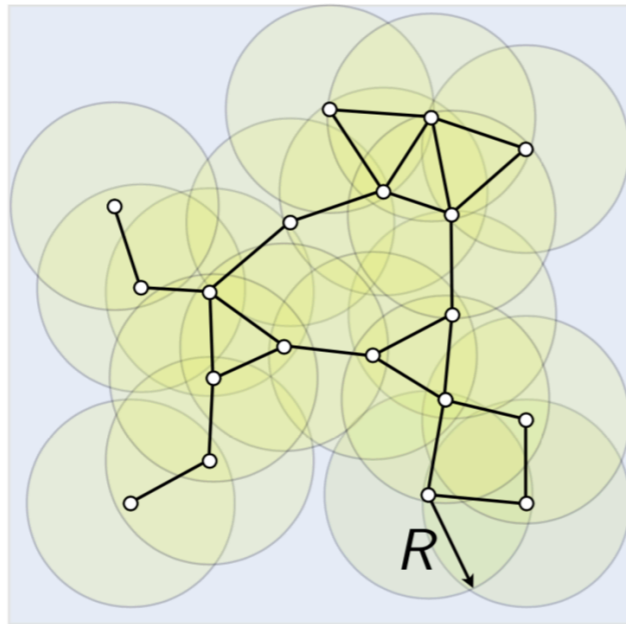


Distance Matrix  $D$

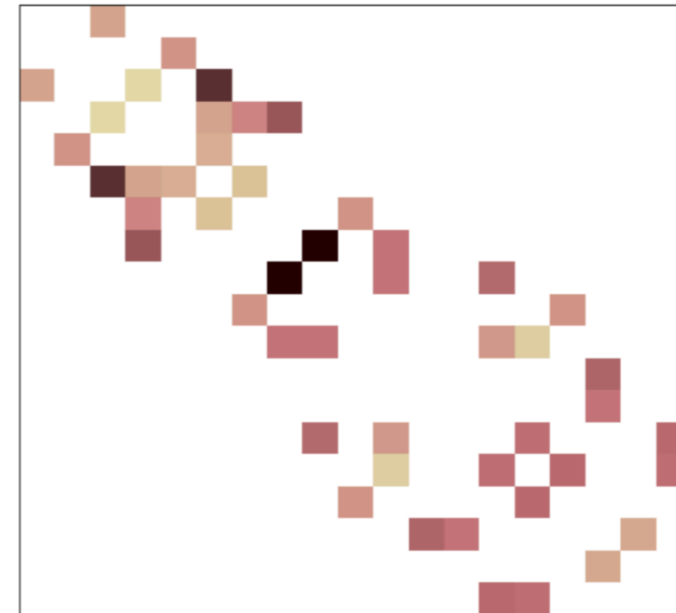


- Wireless sensors deployed in a region
- Each measure distance to the close-by sensors
- Goal: find the distances to all sensors
  - If we have all pairwise distances, then it is easy to find locations of all sensors simultaneously

# Application: localization



Distance Matrix D



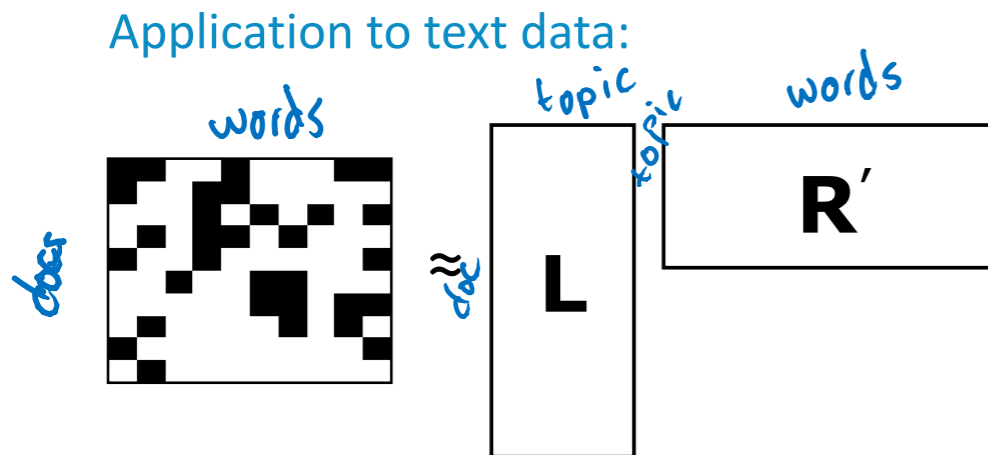
- Why is this a **Matrix Completion** problem?
  - We have missing entries
  - The data is in a matrix form
  - But most importantly, the ground-truth is a low-rank matrix
    - The ambient dimension is 2 or 3, i.e. position is  $(x_u, y_u)$

$$D_{uv} = (x_u - x_v)^2 + (y_u - y_v)^2$$

$$D = \begin{matrix} & \mathbf{1} & L & & \\ \mathbf{1} & x_u^2 + y_u^2 & -\sqrt{2}x_u & -\sqrt{2}y_u & \\ & & & & \\ & & & & \\ & & & & \end{matrix} \quad R^T = \begin{matrix} x_v^2 + y_v^2 & & & \\ \mathbf{1} & & & \\ \sqrt{2}x_v & & & \\ \sqrt{2}y_v & & & \end{matrix}$$

# Application: recommendation systems

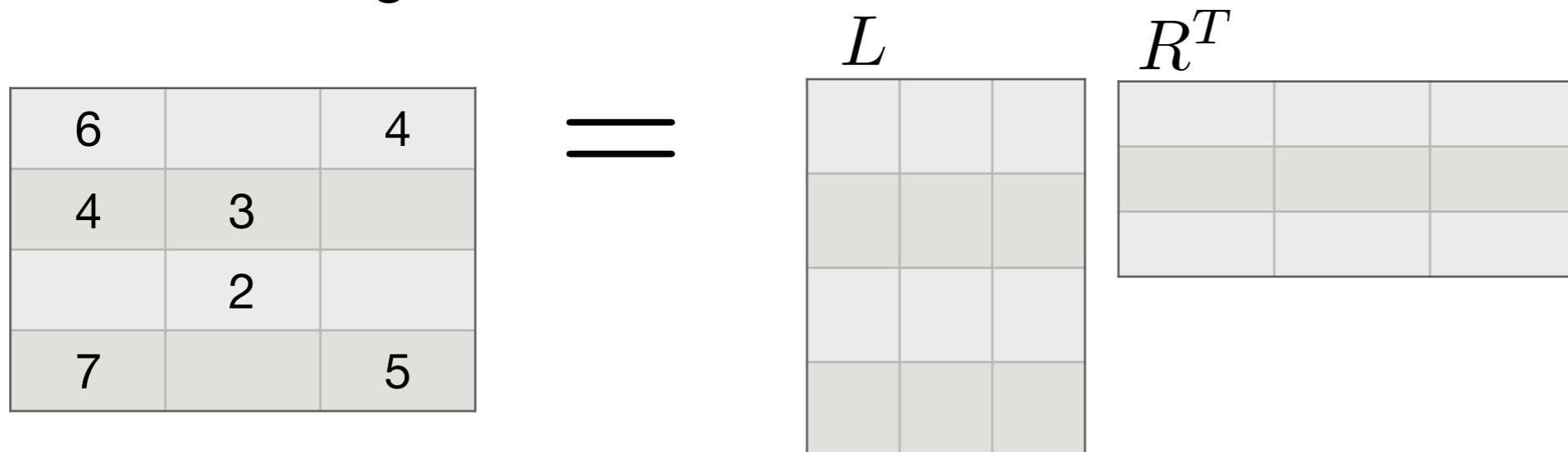
- Given partially observed ratings matrix
- Discover  $k$  topics, and  $k$ -dimensional user features  $L_u$  movie features  $R_v$
- Predict how much a user will like a movie by  $r_{uv} = L_u^T * R_v$
- Make recommendations based on the prediction
- Applied to Wikipedia



party law york county century king engine car war army military  
 government american united roman empire greek design model cars forces battle force british  
 election court city washington john bc ancient emperor ii production built engines command general navy ship  
 president elected texas served virginia kingdom period battle city vehicle class models division ships troops corps  
 council general minister pennsylvania war moved ohio time great war ad early reign kings iii son rule power greece commander infantry attack men  
 political national members florida illinois george james died army centuries dynasty some officer fleet soldiers units officers  
 committee united office federal ottoman years led byzantine defeated version type series motor rear operations unit june august brigade july fire  
 member house parliament vote named jersey born boston south union west company georgia smith began ruled year throne athens capital castle training march battalion april operation  
 public elections democratic held michigan fort years philadelphia white introduced range ford sold fuel drive wheel tank fitted factory machine  
 son died candidate congress senate district seat constitution secret republic campaign mounted early developed based replaced wheels time  
 married family commission supreme votes conservative season team art museum work white red black blue called  
 king daughter john played coach football record teams baseball field year birds small long large animals color will head green gold side  
 death william father born wife royal ireland irish henry house lord including painted architecture york fashion painter life early created sculpture artistic small hand long arms top flag  
 charles sir prince brother children england queen duke thomas years marriage george earl edward english second fm morning host began sports fox air cable city household miles density dog wood body type large  
 school students album band radio station news television age 18 population music musical opera  
 university high college schools education year program student recorded rock bands release live tour video record albums festival orchestra dance  
 campus community programs training center members science national years public academic association courses arts performed jazz piano theatre  
 department teachers colleges classes offers activities universities district engineering learning founded faculty girls symphony composer played performances  
 sports children boys international board teaching academy secondary established singer artists members included early programme day broadcasts moved cbs poverty united village recordings string

# Which is correct about matrix factorization based recommendation systems?

- a) provide personalization
- b) capture context (e.g. time of the day)
- Another weakness of matrix factorization
  - We need to know  $k$ , in some sense
  - If we set  $k = \min\{m, n\}$ , what goes wrong?
    - overfitting

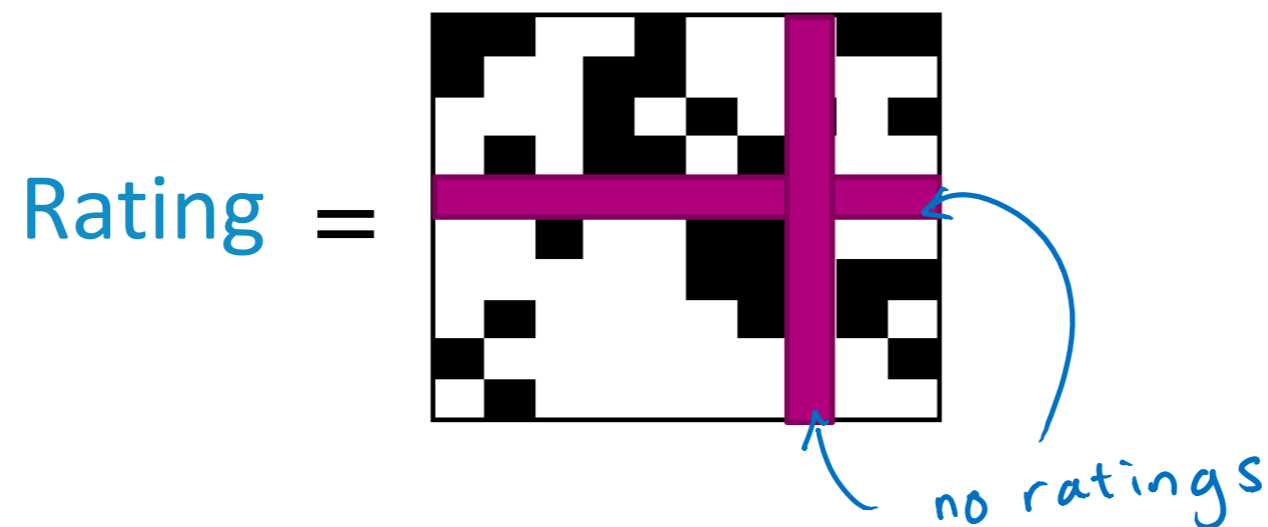


- Solution: regularize

$$\min_{L_u} \sum_{v:r_{uv} \neq ?} (L_u^T R_v - r_{uv})^2 + \lambda \|L_u\|^2$$

# Featured matrix factorization

- Limitations of matrix factorization
  - Cold-start problem
    - This model still cannot handle a new user or movie



- As there is no observation for the entire row/column putting anything in that row has no penalty

$$\text{minimize}_{L,R} \sum_{u,v:r_{uv} \neq ?} \left( \underbrace{(LR^T)_{uv}}_{L_u^T R_v} - r_{uv} \right)^2$$



# Combining features and discovered topics

- Features capture **context**
  - *Time of day, what I just saw, user info, past purchases,...*
- Discovered topics from matrix factorization capture **groups of users** who behave similarly
  - *Women from Seattle who teach and have a baby*
- **Combine** to mitigate cold-start problem
  - Ratings for a new user from **features** only
  - As more information about user is discovered, matrix factorization **topics** become more relevant

# Collaborative filtering with specified features

- Create feature vector for each movie (often have this even for new movies):

$$\phi(v) = (\text{genre, year, director, ...})$$

*( 'action', 1994, Tarantino, ... )*

- Define weights on these features for how much **all users** like each feature

$$w = \text{vector of same length}$$

- Fit linear model:

$$\text{For all users, } r_{uv} \approx w \cdot \phi(v) \quad \text{standard regression model}$$

- Minimize:

$$\min_w \sum_{r_{uv} \neq ?} (w \cdot \phi(v) - r_{uv})^2 + \lambda \|w\| \quad \leftarrow \text{LS, lasso, ridge}$$

# Building in personalization

- Of course, users do *not* have identical preferences
- Include a user-specific deviation from the global set of user weights:

$$r_{uv} \approx (w + w_u) \cdot \phi(v)$$

← personalization to user  $u$  = deviation from crowd vector  $w$

- If we don't have any observations about a user, **use wisdom of the crowd**

$$\text{Initialize } w_u = 0 \Rightarrow r_{uv} \approx w \cdot \phi(v)$$

- As we gain more information about the user, **forget the crowd**

$w_u$  more informed (personalization)

- Can add in user-specific features, and cross-features, too

$$\phi(u) = (\text{age, gender, education} \\ 25, F, MSc, \dots)$$
$$\phi(u, v) = (\dots \phi(u) \dots, \dots \phi(v) \dots, \text{cross features})$$

# Featurized matrix factorization— A combined approach

## Feature-based approach:

- Feature representation of user and movies fixed
- Can address cold-start problem

## Matrix factorization approach:

- Suffers from cold-start problem
- User & movie features are learned from data

## A unified model:

$$r_{uv} \approx L_u \cdot R_v + (w + w_u) \cdot \phi(u, v)$$

Solve via coord. desc., grad. desc., etc.