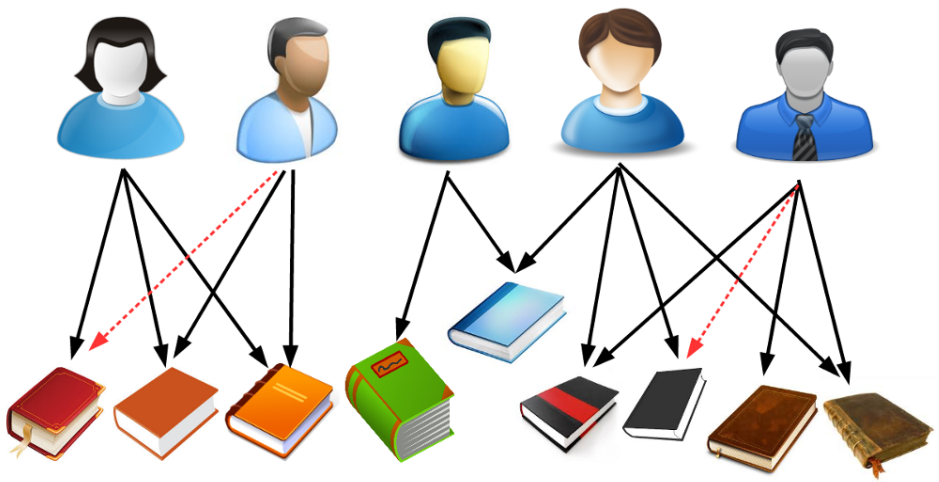
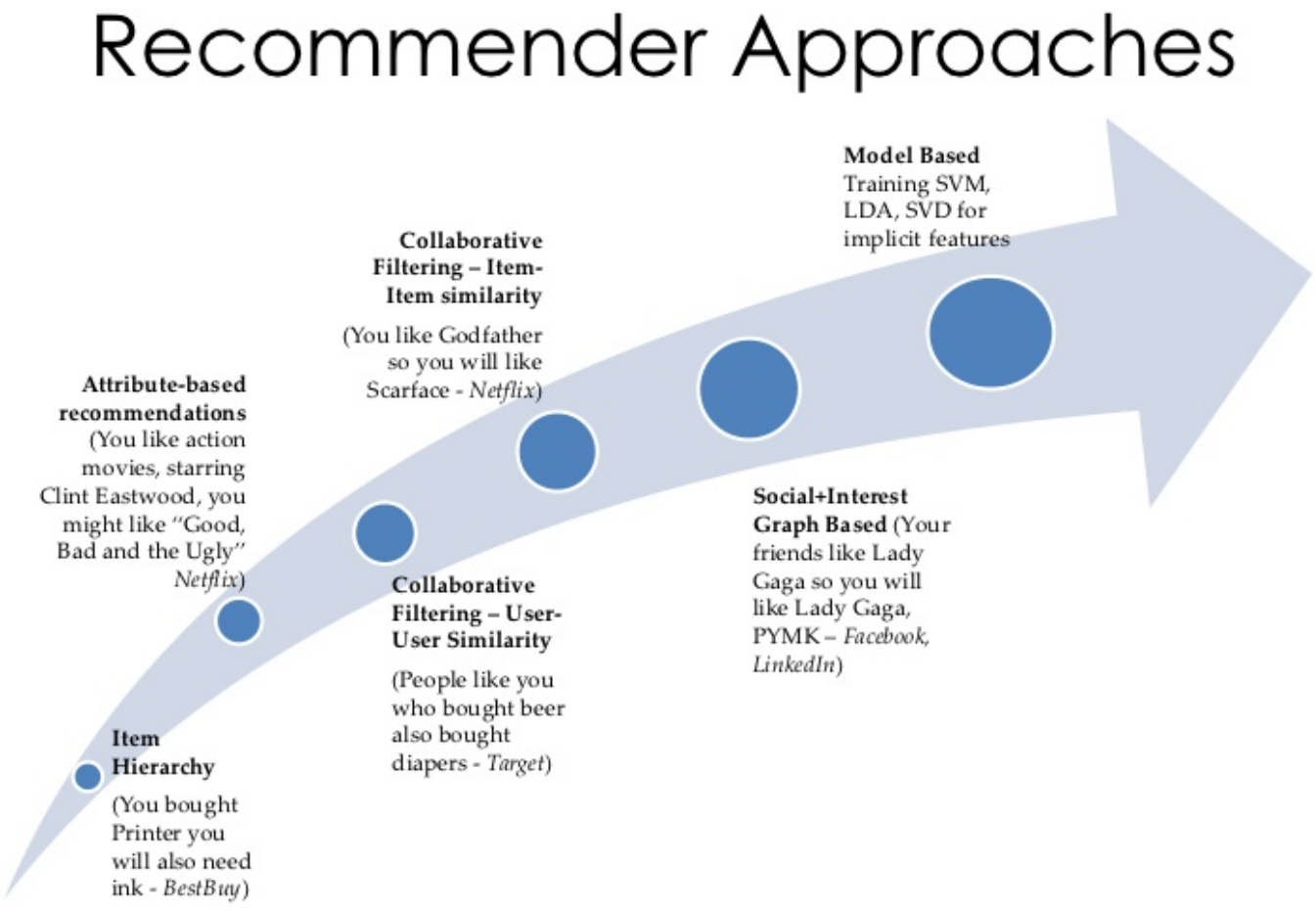
Lecture 8 - Graphs and Machine Learning II

F. d'Alché-Buc, E. Le Pennec

Fall 2016

A set of users, a set of items, some users rate some items. Given a user, predict a score attached to an item.




Recommender Approaches
Item Hierarchy
(You bought Printer you will also need ink - BestBuy)
Attribute-based recommendations
(You like action movies, starring Clint Eastwood, you might like "Good, Bad and the Ugly" Netflix)
Collaborative Filtering – Item-Item similarity
(You like Godfather so you will like Scarface - Netflix)
Collaborative Filtering – User-User Similarity
(People like you who bought beer also bought diapers - Target)
Social+Interest Graph Based (Your friends like Lady Gaga so you will like Lady Gaga, PYMK – Facebook, LinkedIn)
Model Based
Training SVM, LDA, SVD for implicit features

If items could be described by feature vectors x or pairs of some metric based on content, and idem for users profile y. Then, build a function $f : (x, y) \rightarrow r$ from a training set
$\{((x_i, y_i), r_i), i = 1, \ldots, n\}$
Any supervised regression method able to cope with an ordered pair is suitable.
Example: Movies: genre, actors, ... People: age, gender, job, ...

- Appropriate metric or kernel:$k((x, y), (x', y')) = k_i(x, x')k_u(y, y')$: kernel approaches OR k-nearest neighbors
- Tree-based methods on input $z^T = [x^T y^T]$

N.B.: it is closed to a link prediction problem in a bipartite graph

- Pros
  - You learn a function so you can predict for a new pair of item and user
  - Benefit from existing regression methods
- Cons
  - Requires features, maybe easy in the case of items but difficult in the case of users
  - Data are no more iid

What is collaborative filtering? a transductive problem.

Many **users**

Many **items**

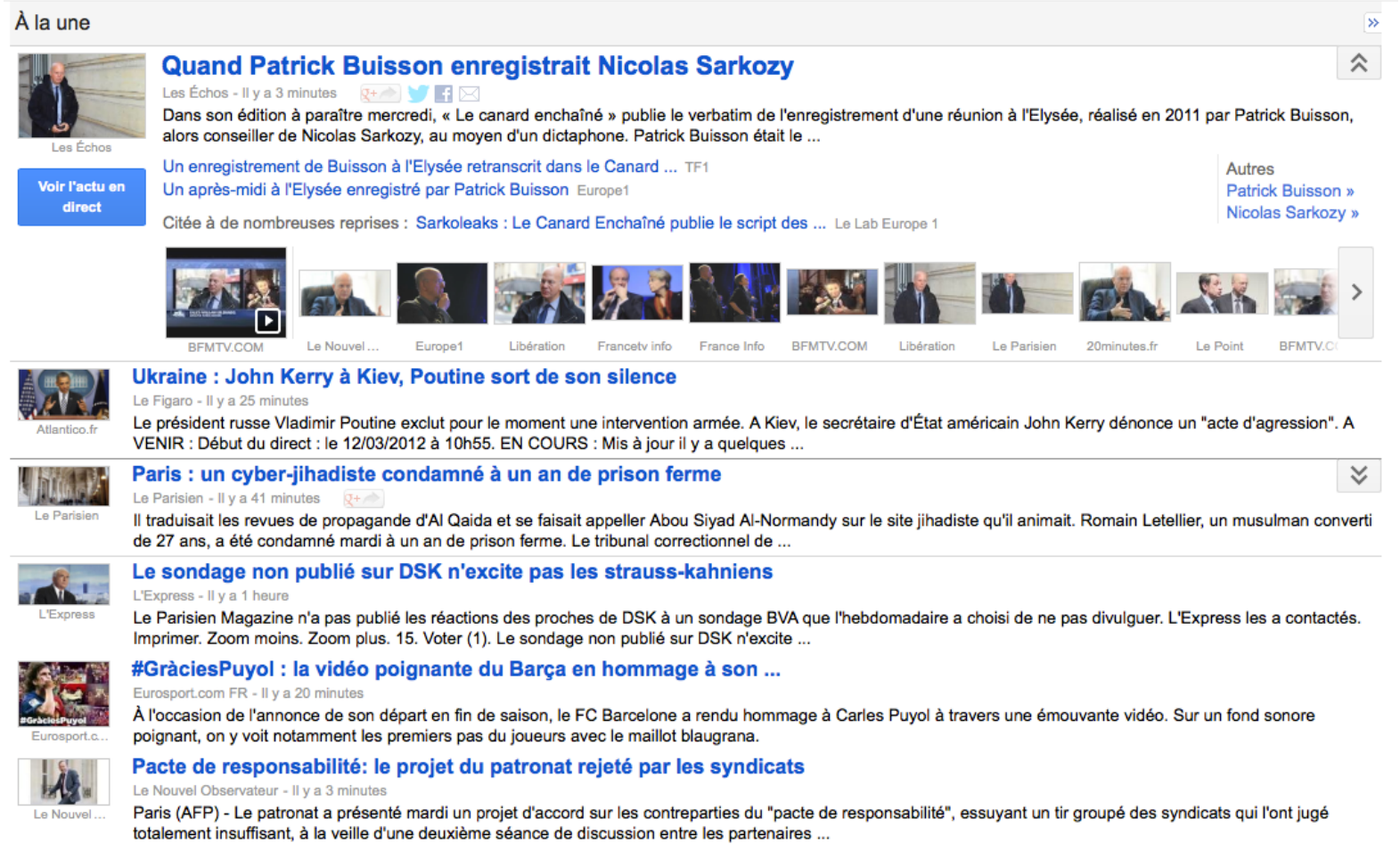
- Based on many observed user-item interactions (rating, purchase, click)
- Predict new interactions

Amazon.com recommends products based on your purchases, browsing history

- but based on the purchases and history of other users too

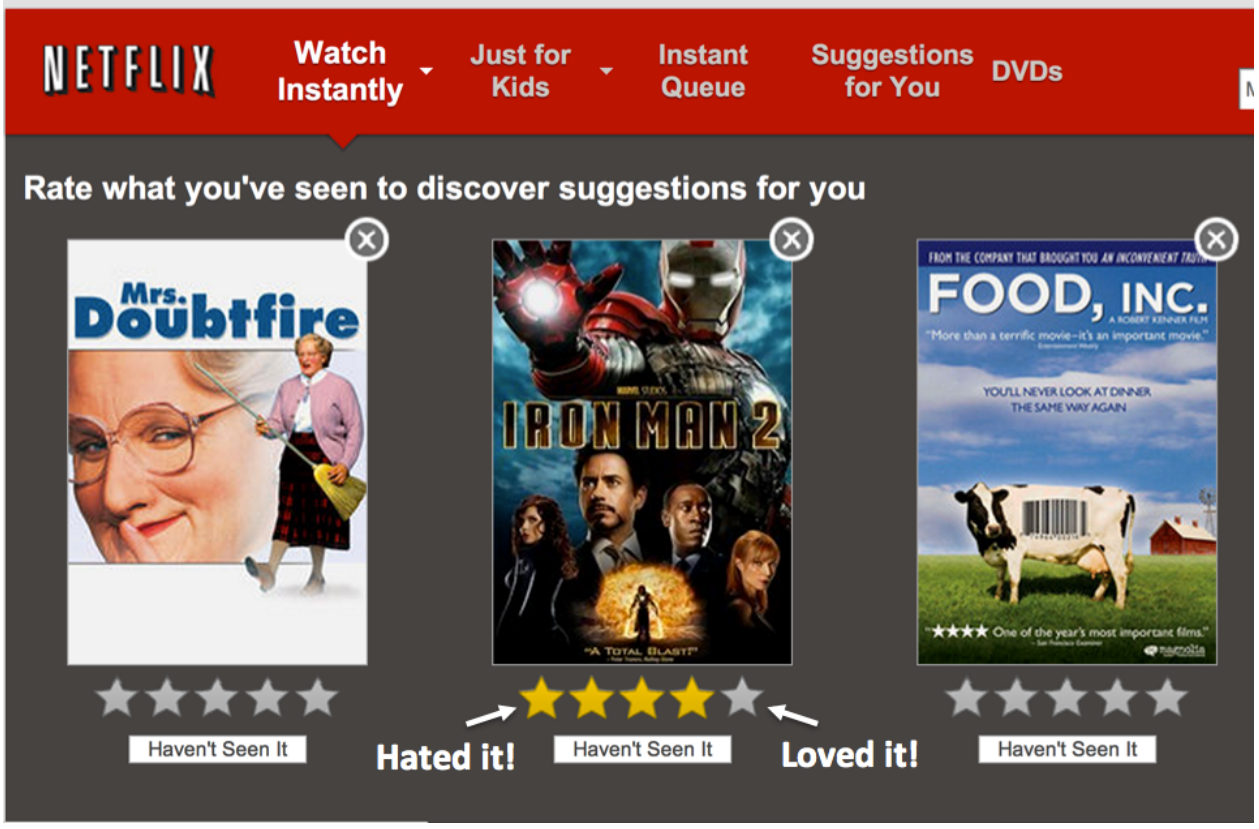
Google News recommends news based on your browsing activity

- but on the browsing activity of other users too



Netflix predicts the rating you'd give to a movie

- using all the ratings given by all users



- 60% of Netflix's DVD rentals due to recommendations

Netflix predicts the rating you'd give to a movie

- using all the ratings given by all users

- 60% of Netflix's DVD rentals due to recommendations

Introduction to recommendation systems

Collaborative Filtering: Netflix Prize


How can we improve Cinematch?
Let's have a contest!
What sho
the pri
How about $1 million?

Netflix Prize

- October 2, 2006
- Dataset: 100 million ratings , n_U =480K users, n_I =18K movies
- **only 1.1%** of the matrix is filled!
- **Goal**: create a computer code that predicts ratings
- $1m grand prize for anyone beating **Cinematch** accuracy by 10%
- 5000 teams over 150 countries participated

Collaborative Filtering, Matrix completion: **fill unobserved entries** of a matrix

- Unknown matrix M has size $n_U \times n_I$
- Observe $m \ll n_U n_I$ entries (100m $\ll$ 8.64M for Netflix)
- Seems impossible!

There is hope:

- Personal preferences are correlated: if Alice likes A and B and Bob likes A, B and C, then Alice is more likely to like C
- There are latent factors that describe the data in a much lower dimensional space. Groups of users, groups of movies, factors that explain the taste of users. High-dimensionality but hidden low-dimensional structure

Collaborative Filtering task:

- discover patterns (low-dimensional hidden structure, latent factors)
- use these patterns for prediction of new user-item interactions
- Do not necessarily use item or user attributes (demographic information, author, genre, cast, plot, etc.)

Given:

- Users $u \in \{1, \ldots, n_U\}$
- Items $i \in \{1, \ldots, n_I\}$
- When u has an interaction with item i, (watches the movie, clicks on a banner, buys a product), he enters a scalar rating $r_{u,i}$ (number of clicks, rating of a movie, etc.)
- Set E of pairs (u, i) of observed ratings $r_{u,i}$

Matrix completion problem

$$M = \begin{bmatrix} ? & ? & 2 & \cdots & 5 \\ 2 & ? & 1 & \cdots & ? \\ ? & 2 & ? & \cdots & 4 \end{bmatrix}$$

Measures of success. Decompose $E = E_{\text{train}} \cup E_{\text{test}}$ into training and testing respectively. $r_{u,i}$ = ground truth and $\hat{r}_{u,i}$ = estimated rating

- Root Mean Square Error

$$\texttt{RMSE} = \sqrt{\frac{1}{|E_{\text{test}}|} \sum_{(u,i) \in E_{\text{test}}} (r_{u,i} - \hat{r}_{u,i})^2}$$

- Mean Absolute Error

$$\texttt{MAE} = \frac{1}{|E_{\text{test}}|} \sum_{(u,i) \in E_{\text{test}}} |r_{u,i} - \hat{r}_{u,i}|$$

- Ranking error: fraction of true top-5 preferences are in my predicted top 5?

- Remove biases from the ratings
- Some users give systematically higher ratings
- Some items gets systematically better rates (old movies...)
- Don't forget that PCA needs centering

Remove bias

$$\widetilde{r}_{u,i} = r_{u,i} - b_{u,i}$$

Let's denote

- $U(i)$ the set of users who rated item i
- $I(u)$ the set of items rated by user u

Let's compute means! We can choose $b_{u,i}$ as one of the following:

- Global mean

$$b = \frac{1}{|E|} \sum_{(u,i) \in E} r_{u,i}$$

- Item's mean rating

$$b_i = \frac{1}{|U(i)|} \sum_{u \in U(i)} r_{u,i}$$

- User's mean rating

$$b_u = \frac{1}{|I(u)|} \sum_{i \in I(u)} r_{u,i}$$

- Item's mean rating + user's mean deviation from item mean

$$b_{u,i} = b_i + \frac{1}{|I(u)|} \sum_{i' \in I(u)} (r_{u,i'} - b_{i'})$$

- Some users have much more ratings than other (1000 $\times$ more!)
- Users with a small numbers of ratings are not as reliable as ones: more noisy. It's hard to trust a mean with only one rating!
- Interpolate between a global estimate and an estimate from user's data

Interpolate to have a better bias estimation:

$$\widetilde{b}_u = \frac{\alpha}{\alpha + |I(u)|} b + \frac{|I(u)|}{\alpha + |I(u)|} b_u$$

where we recall that b is the global mean

$$b = \frac{1}{|E|} \sum_{(u,i) \in E} r_{u,i}$$

and b_u is the user's mean

$$b_u = \frac{1}{|I(u)|} \sum_{i \in I(u)} r_{u,i}$$

K-Nearest Neighbor Method (K-NN) most widely used family of methods

- item-based or user-based
- for product recommendation: **item-based**
- represent each item as a vector of user's ratings with many missing values $r_{\bullet,i} = [2, ?, 1, ?, ?, ?, 2, 1, ?, ?, ?, 5]$

Idea: users rate similar items similarly.

In order to predict a rating $r_{u,i}$ for user u and item i:

- Compute similarity between i and every other items
- Find the K items rated by u most similar to i
- Compute weighted average of these ratings

How to measure similarity between items?

- Cosine similarity

$$d(r_{\bullet,i}, r_{\bullet,i'}) = \frac{\langle r_{\bullet,i}, r_{\bullet,i'} \rangle}{\|r_{\bullet,i}\| \|r_{\bullet,i'}\|}$$

- Pearson correlation coefficient

$$d(r_{\bullet,i}, r_{\bullet,i'}) = \frac{\langle r_{\bullet,i} - \bar{r}_{\bullet,i}, r_{\bullet,i'} - \bar{r}_{\bullet,i'} \rangle}{\|r_{\bullet,i} - \bar{r}_{\bullet,i}\| \|r_{\bullet,i'} - \bar{r}_{\bullet,i'}\|}$$

- Inverse Euclidean distance

$$d(r_{\bullet,i}, r_{\bullet,i'}) = \frac{1}{\|r_{\bullet,i} - r_{\bullet,i'}\|}$$

Vectors $r_{\bullet,i}$ contains many missing values: compute these similarities over subets of users that rated both items i and i'

How to choose the K-nearest neighbors?

- Select the K items with largest similarity score to item i, among the items rated by u, denoted $N(i, u)$
- Prediction given by

$$\hat{r}_{u,i} = b_{u,i} + \sum_{i' \in N(i,u)} w_{i,j}(r_{u,i'} - b_{u,i'})$$

where $w_{i,i'}$ weights

Example of weights:

- Equal weights

$$w_{i,i'} = \frac{1}{N(u,i)}$$

- Similarity weights

$$w_{i,i'} = \frac{d(r_{\bullet,i}, r_{\bullet,i'})}{\sum_{i'' \in N(i,u)} d(r_{\bullet,i}, r_{\bullet,i''})}$$

Event better: user optimized weights.

- Choose weights that best predict other known ratings of i among all users that rated i
- Corresponds to many small linear regression problems
- Needs to store **many** weights $O(n_I^2)$

- Easy to implement
- No training time
- Flexible
- But need to store many parameters (all item, vectors, weights in memory)
- Don't exploit hidden low-dimensional structure

Matrix Factorization (= MF)

- Assume that we know features about the items

$$y_i = [\text{cast}, \text{year}, \text{genre}, \cdots] \in \mathbb{R}^r$$

  for all $i = 1, \ldots, n_I$.
- r features for each item
- We want the users parameters x_u for $u = 1, \ldots, n_U$

Linear regression

$$\hat{x}_u \in \operatorname*{argmin}_{x_u} \sum_{i \in I(u)} (r_{u,i} - \langle x_u, y_i \rangle)^2.$$

Even better: **ridge regression**

$$\hat{x}_u = \operatorname*{argmin}_{x_u} \sum_{i \in I(u)} (r_{u,i} - \langle x_u, y_i \rangle)^2 + \lambda \|x_u\|_2^2$$

- But we don't want to construct ad-hoc features y_i for items
- Not a good idea for building recommendations

- Assume that we know user's features x_u for all $u = 1, \ldots, n_U$.
- r features for each user
- We want the items features y_i for $i = 1, \ldots, n_I$

Once again: **linear regression**

$$\hat{y}_i \in \operatorname*{argmin}_{y_i} \sum_{u \in U(i)} (r_{u,i} - \langle x_u, y_i \rangle)^2.$$

Even better: **ridge regression**

$$\hat{y}_i \in \operatorname*{argmin}_{y_i} \sum_{u \in U(i)} (r_{u,i} - \langle x_u, y_i \rangle)^2 + \lambda \|y_i\|_2^2$$

- But we can't construct ad-hoc features x_u for users
- Still not a good idea for building recommendations

- We don't want to construct ad-hoc features y_i for items
- We can't construct ad-hoc features x_u for users

So let's learn items and users features **at the same time**!

Putting altogether

$$\hat{x}_u = \underset{x_u}{\operatorname{argmin}} \sum_{i \in I(u)} (r_{u,i} - \langle x_u, \hat{y}_i \rangle)^2 + \lambda \|x_u\|_2^2$$

$$\hat{y}_i = \underset{y_i}{\operatorname{argmin}} \sum_{u \in U(i)} (r_{u,i} - \langle \hat{x}_u, y_i \rangle)^2 + \lambda \|y_i\|_2^2$$

for all $u = 1, \ldots, n_U$ and $i = 1, \ldots, n_I$

$$\hat{x}_u = \underset{x_u}{\operatorname{argmin}} \sum_{i \in I(u)} (r_{u,i} - \langle x_u, \hat{y}_i \rangle)^2 + \lambda \|x_u\|_2^2$$

$$\hat{y}_i = \underset{y_i}{\operatorname{argmin}} \sum_{u \in U(i)} (r_{u,i} - \langle \hat{x}_u, y_i \rangle)^2 + \lambda \|y_i\|_2^2$$

for all $u = 1, \ldots, n_U$ and $i = 1, \ldots, n_I$

- **Issue:** the $\hat{x}_u$'s depends on the $\hat{y}_i$'s that depend on the $\hat{x}_u$'s that depends on the $\hat{y}_i$ that... !

Let's rewrite this. Put

$$X^\top = \begin{bmatrix} \vdots & \vdots & \vdots \\ x_1 & \cdots & x_{n_U} \\ \vdots & \vdots & \vdots \end{bmatrix} \quad \text{and} \quad Y^\top = \begin{bmatrix} \vdots & \vdots & \vdots \\ y_1 & \cdots & y_{n_I} \\ \vdots & \vdots & \vdots \end{bmatrix}$$

Then we consider the minimization of

$$F(X, Y) = \sum_{(u,i)\in E} (r_{u,i} - \langle x_u, y_i \rangle)^2 + \lambda \sum_{u=1}^{n_U} \|x_u\|_2^2 + \lambda \sum_{i=1}^{n_I} \|y_i\|_2^2$$

over $X \in \mathbb{R}^{n_U \times r}$ and $Y \in \mathbb{R}^{n_I \times r}$ jointly.

- The penalization terms $\lambda \sum_{u=1}^{n_U} \|x_u\|_2^2 + \lambda \sum_{i=1}^{n_I} \|y_i\|_2^2$ prevent from overfitting

$$F(X,Y) = \sum_{(u,i)\in E} (r_{u,i} - \langle x_u, y_i \rangle)^2 + \lambda \sum_{u=1}^{n_U} \|x_u\|_2^2 + \lambda \sum_{i=1}^{n_I} \|y_i\|_2^2$$

over $X \in \mathbb{R}^{n_U \times r}$ and $Y \in \mathbb{R}^{n_I \times r}$ jointly.
Let's write this with matrices

$$F(X,Y) = \|\mathcal{P}_E(R - XY^\top)\|_F^2 + \lambda \|X\|_F^2 + \lambda \|Y\|_F^2$$

where

$$\|A\|_F = \text{ Frobenius norm of } A = \sqrt{\sum_{j,k} A_{j,k}^2}$$

and

$$\mathcal{P}_E(A)_{u,i} = \begin{cases} A_{u,i} & \text{if } (u,i) \in E \\ 0 & \text{otherwise} \end{cases}$$

Put $\lambda = 0$ and $E = \{1, \ldots, n_U\} \times \{1, \ldots, n_I\}$

$$F(X, Y) = \|\mathcal{P}_E(R - XY^\top)\|_F^2 = \|R - XY^\top\|_F^2$$

Solution is given by the SVD (Singular Value Decomposition)
Recall that

- X of size $n_U \times r$
- Y of size $n_I \times r$

Then

$$\operatorname*{argmin}_{X,Y} \|R - XY^\top\|_F^2$$

is given by thresholded SVD of R

SVD (Singular Value Decomposition)

Any matrix $R \in \mathbb{R}^{n_U \times n_I}$ writes

$$R = U \Sigma V^\top$$

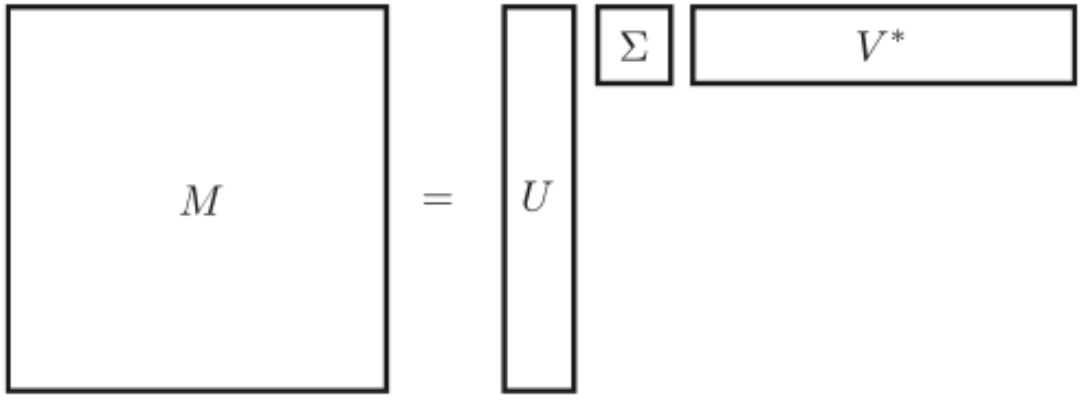
where

- U is the matrix of **left singular vectors** (columns of U are eigenvectors of $RR^\top$, it satisfies $U^\top U = I$
- V is the matrix of **right singular vectors** (eigenvectors of $R^\top R$, it satisfies $V^\top V = I$
- $\Sigma = \text{diag}[\sigma_1, \ldots, \sigma_{n_U \wedge n_I}]$ is the diagonal matrix containing the **singular values**

$$\sigma_1 \geq \cdots \geq \sigma_{n_U \wedge n_I}$$

where

$$\sigma_j = \sqrt{\lambda_j(R^\top R)} = j\text{th eigenvalue of } R^\top R$$



Fundamental result:

$$\underset{M \in \mathbb{R}^{n_U \times n_I} : \text{rank}(X) = r}{\text{argmin}} \|R - M\|_2^2 = U_r \Sigma_r V_r^\top$$

where $R = U \Sigma V^\top$ is the SVD of R and

- $\Sigma_r = \text{diag}[\sigma_1, \ldots \sigma_r]$
- U_r contains the first r columns of U
- V_r contains the first r columns of V
- Don't forget that PCA = SVD of the covariance matrix

Hence a solution of

$$(\hat{X}, \hat{Y}) \in \underset{X,Y}{\operatorname{argmin}} \|R - XY^\top\|_F^2$$

is given by

$$\hat{X} = U_r \Sigma_r \quad \text{and} \quad \hat{Y} = V_r^\top$$

- Matrix completion can be understood as an SVD with missing entries
- With extra regularization to avoid overfitting using ridge penalization

Ok. So how do I solve

$$F(X,Y) = \sum_{(u,i)\in E} (r_{u,i} - \langle x_u, y_i \rangle)^2 + \lambda \sum_{u=1}^{n_U} \|x_u\|_2^2 + \lambda \sum_{i=1}^{n_I} \|y_i\|_2^2$$

or equivalently

$$F(X,Y) = \|\mathcal{P}_E(R - XY^\top)\|_F^2 + \lambda \|X\|_F^2 + \lambda \|Y\|_F^2$$

Idea: if we knew Y we could solve X using ridge regression and vice-versa: alternate between optimizing on X and Y with the other matrix fixed

Alternating least-squares (ALS) algorithm

Repeat until *convergence*:

- For each u solve the linear system:
  $x_u^{\text{new}} \leftarrow$ solution of $\sum_{i \in I(u)} (y_i y_i^\top + \lambda I) x_u = \sum_{i \in I(u)} r_{u,i} y_i$
- For each item i solve
  $y_i^{\text{new}} \leftarrow$ solution of $\sum_{u \in U(i)} (x_u x_u^\top + \lambda I) y_i = \sum_{u \in U(i)} r_{u,i} x_u$
- $x_u \leftarrow x_u^{\text{new}}$, $y_i \leftarrow y_i^{\text{new}}$

- Updates for x_u and y_i can be done in parallel
- Complexity. Space: $O(n_u r + n_I r)$ and time: $O(n_u r^3 + n_I r^3)$ per iteration. $O(r^3)$ for solving the linear systems
- No need to store the complete ratings matrix

Idea: use standard gradient descent

- $\nabla_{x_u} F(X, Y) = \lambda x_u + \sum_{i \in I(u)} (\langle x_u, y_i \rangle - r_{u,i}) y_i$
- $\nabla_{y_i} F(X, Y) = \lambda y_i + \sum_{u \in U(i)} (\langle x_u, y_i \rangle - r_{u,i}) x_u$

Gradient Descent algorithm

Repeat until *convergence*:

- For each u update
  $x_u^{\text{new}} \leftarrow x_u - \eta \left(\lambda x_u + \sum_{i \in I(u)} (\langle x_u, y_i \rangle - r_{u,i}) y_i \right)$
- For each i update
  $y_i^{\text{new}} \leftarrow y_i - \eta \left(\lambda y_i + \sum_{u \in U(i)} (\langle x_u, y_i \rangle - r_{u,i}) x_u \right)$
- $x_u \leftarrow x_u^{\text{new}}$, $y_i \leftarrow y_i^{\text{new}}$

- Updates for x_u and y_i can be done in parallel
- Complexity: $O(n_u r + n_I r)$ no $O(r^3)$ overhead iteration
- No need to store the complete ratings matrix

Stochastic Gradient Descent algorithm

Repeat until *convergence*:

- Choose $(u, i) \in E$ at random
- Update x_u
  $x_u^{\text{new}} \leftarrow x_u - \eta(\lambda x_u + (\langle x_u, y_i \rangle - r_{u,i}) y_i)$
- Update y_i
  $y_i^{\text{new}} \leftarrow y_i - \eta(\lambda y_i + (\langle x_u, y_i \rangle - r_{u,i}) x_u)$
- $x_u \leftarrow x_u^{\text{new}}$, $y_i \leftarrow y_i^{\text{new}}$

- Complexity: $O(n_u r + n_I r)$ no $O(r^3)$ overhead iteration
- No need to store the complete ratings matrix

Parameters to tune:

- step-size, or learning rate η. Must be decreasing for SGD
- Regularization parameter $\lambda > 0$ and number of latent factors r. Tuned using cross-validation

There is a big problem:

$$F(X, Y) = \sum_{(u,i)\in E} (r_{u,i} - \langle x_u, y_i \rangle)^2 + \lambda \sum_{u=1}^{n_U} \|x_u\|_2^2 + \lambda \sum_{i=1}^{n_I} \|y_i\|_2^2$$

is **not** a convex problem

- Local minimum, initialization is important
- No guarantees towards a good minimum
- Mostly heuristics

- Unknown matrix R of size $n_U \times n_I$
- If R has rank r, its degrees of freedom are $r(n_U + n_I - r)$
- $E \subset \{1, \dots, n_U\} \times \{1, \dots, n_I\}$ of observed entries of R
- We need $|E| \geq r(n_U + n_I - r)$ (otherwise no hope to recover R
- We observe only $\mathcal{P}_E(R)$

We assume that the rank of R is small. So let's penalize the rank

- Tempting to consider

$$\hat{R} \in \operatorname*{argmin}_{M \in \mathbb{R}^{n_U \times n_I}} \left\{ \frac{1}{2} \|\mathcal{P}_E(M - R)\|_F^2 + \lambda \operatorname{rank}(M) \right\}$$

- Too hard
- For Lasso we've found that a convex relaxation of ℓ_0 is ℓ_1
- Can't we do the same for the rank?

Yes!

Convex relaxation for the rank

$$\text{rank}(M) = \sum_{k=1}^{n_U \wedge n_I} \mathbf{1}_{\sigma_j(M)>0} = \|\sigma(M)\|_0$$

Use nuclear norm:

$$\|M\|_* = \sum_{j=1}^{n_I \wedge n_U} \sigma_j(M)$$

Hence tempting to consider

$$\hat{R} \in \underset{M \in \mathbb{R}^{n_I \times n_U}}{\text{argmin}} \left\{ \frac{1}{2} \|\mathcal{P}_E(M - R)\|_2^2 + \lambda \|M\|_* \right\}$$

for a regularization parameter $\lambda > 0$. This is a **convex** problem!

Proximal gradient descent for the CF problem

Repeat until *convergence*:

- $M \leftarrow S_{\lambda\eta_k}(M - \eta_k(\mathcal{P}_E(M - R)))$

where S_λ is the spectral soft-thresholding operator: if $M = U\Sigma V^\top$ SVD of M, then

$$S_\lambda(M) = U\,\mathrm{diag}[(\sigma_1 - \lambda)_+, \ldots, (\sigma_{n_1 \wedge n_2} - \lambda)_+]V^\top$$

Thresholding of the singular value: the solution will be of low rank. Many other algorithms, more memory efficient and faster

- Convex problem: convex optimization and convergence guarantees to a minimum
- Bottleneck: a SVD is required at **each iteration**! Complexity of a SVD $O((n_U \vee n_I)(n_U \wedge n_I)^2)$
- Can be reduced using partial SVD (compute only k top singular values and vectors). Complexity is (best case) $O(n_1 n_2 k)$ [keyword: **Lanczos** algorithms]
- Compute an approximate solution, given some tolerance
- Remedy for large SVD is the **divide and conquer** principle

Sketch of application: image inpainting

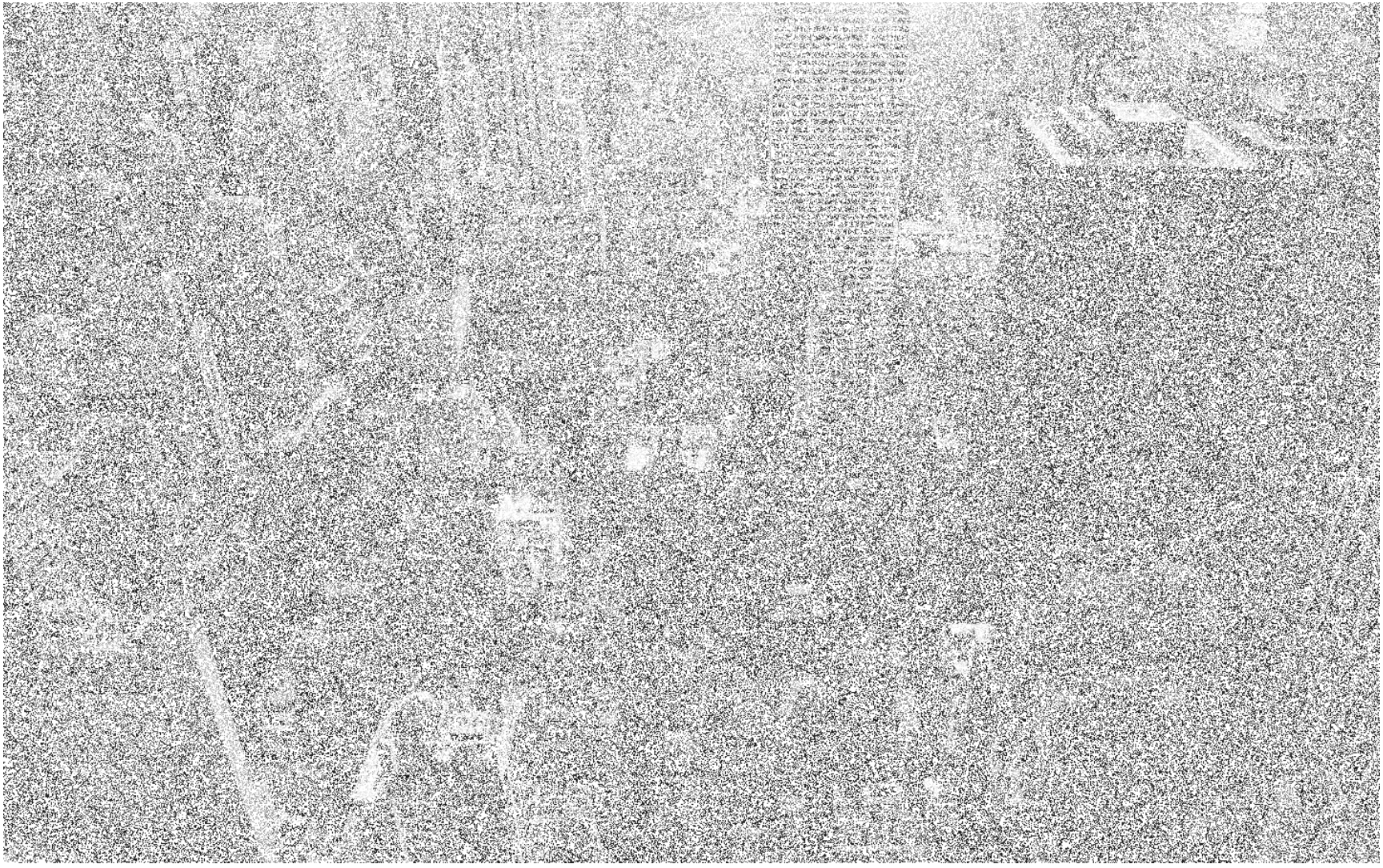
Sketch of application: image inpainting

Sketch of application: image inpainting

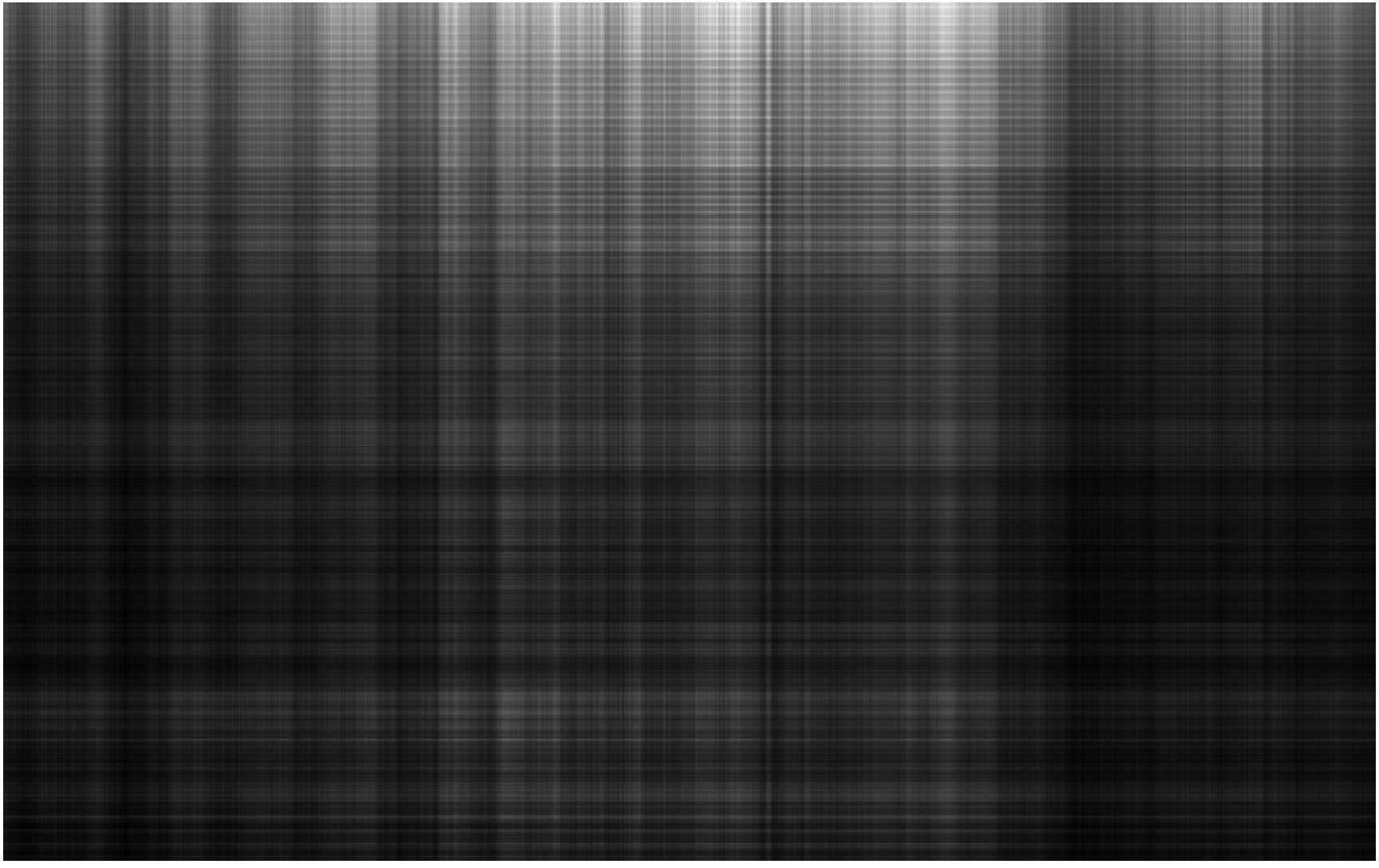
Sketch of application: image inpainting

Sketch of application: matrix completion

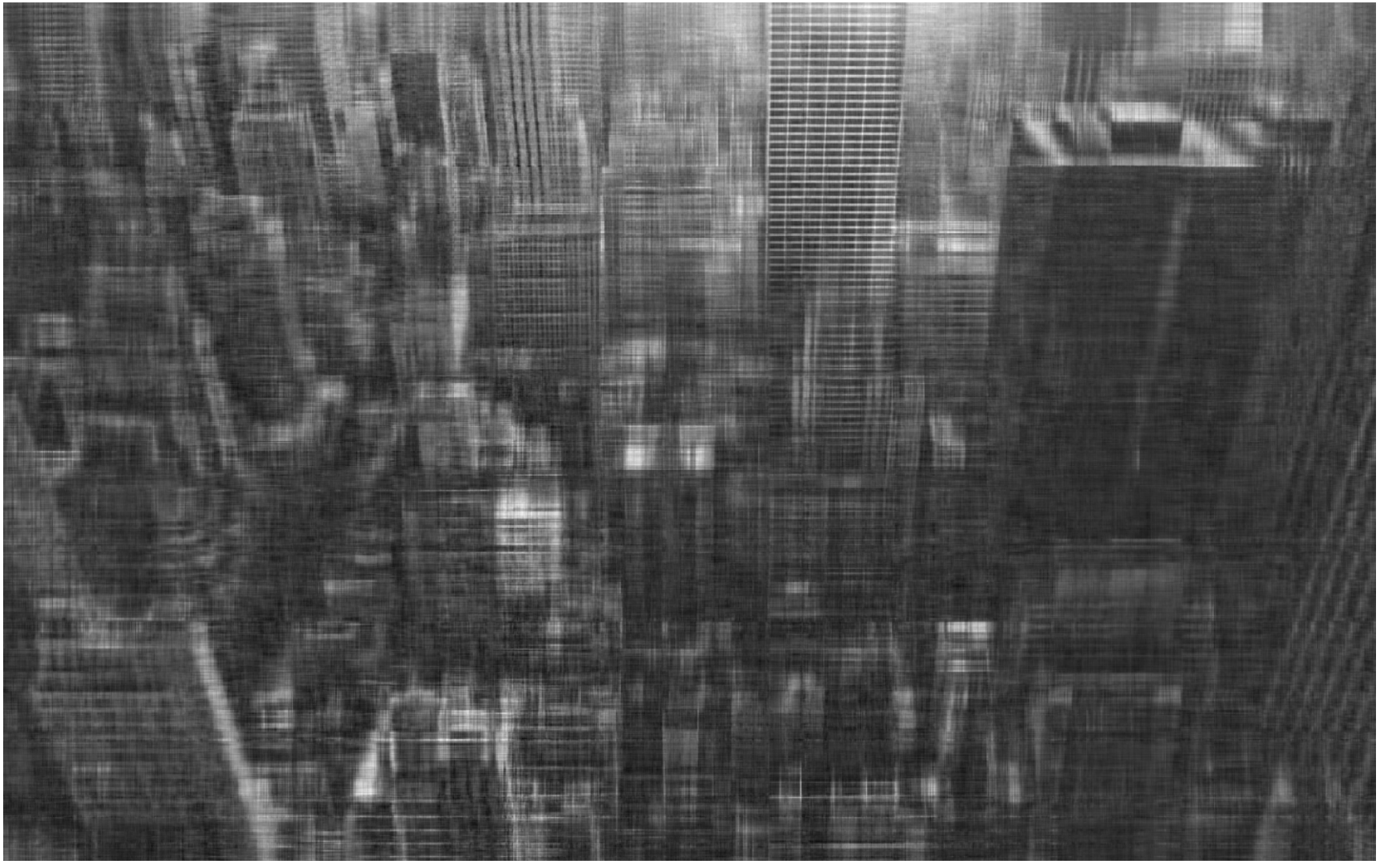
We only observe 35% of the picture



Solutions of increasing rank 2



Solutions of increasing rank 49



Solutions of increasing rank 200.

Exact reconstruction (no noise)

$$\hat{R} \in \text{argmin}\{\|M\|_* \text{ such that } \mathcal{P}_E(M) = \mathcal{P}_E(R)\}$$

Assume $n = n_U = n_I$ for short and put $m = |E|$. Then under some assumptions

- No method can succeed if $m \leq crn \log n$. Namely, need **at least**

$$m \geq crn \log n$$

  observed entries to recover M and $r = \text{rank}$
- If $m \geq crn(\log n)^2$ then reconstruction is **exact**! with a large probability
- In this setting, convex relaxation is **exact**: **no loss** when relaxing rank by $\|\cdot\|_*$
- Gives the exact same solution as the one constrained by rank!
- Convex programming incredibly powerful in this case
- Compressed sensing theory (See Emmanuel Candès's work)

Blend multiple variations of MF approaches and k-NN approaches through Gradient boosting decision trees including Matrix Factorization which behaves the best among base methods.

- The Belkor solution to the netflix grand prize, Yehuda Korenhttp://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf
- Matrix factorization techniques, Koren, Bell, Volinski, IEEE Computer 2009 https://datajobs.com/data-science-repo/Recommender-Systems-

- Recommender systems: slides/tutorial Amatriain
- The Belkor solution to the netflix grand prize, Yehuda Korenhttp://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf
- Matrix factorization techniques, Koren, Bell, Volinski, IEEE Computer 2009 https://datajobs.com/data-science-repo/Recommender-Systems-
- Beyond MF, Compressed sensing: http://dsp.rice.edu/cs

- 4 students
- A non trivial problem (no pure classification, no pure regression): quantile regression, active learning, ranking, recommendation, time series modeling, churn,...
- A short presentation 2/3 pages of the problem and state of the art
- Description/discussion of at least 2 methods , referenced in papers or of your own
- Description of evaluation metrics, model selection protocole
- toy dataset + two real datasets : analysis of performance of at least 2 methods
- Conclusion, perspectives

- before Dec 15, 2016: submit the problem to solve (on the website) - you receive a go-nogo answer.
- before Jan 6, 2017: (reportv1) short presentation of the problem and state of the art
- before Jan 31, 2017: (reportv2) Full report + code-ready-for-demo, dataset available