CS5284 : Graph Machine Learning

Lecture 5 : Recommendation on Graphs

Semester 1 2025/26

Xavier Bresson

https://x.com/xbresson

Department of Computer Science
National University of Singapore (NUS)

Course lectures

- Introduction to Graph Machine Learning
- Part 1: GML without feature learning (before 2014)
  - Introduction to Graph Science
  - Graph Analysis Techniques without Feature Learning
    - Graph clustering
    - Graph SVM
    - Recommendation on graphs
    - Graph-based visualization
- Part 2 : GML with shallow feature learning (2014-2016)
  - Shallow graph feature learning
- Part 3 : GML with deep feature learning, a.k.a. GNNs (after 2016)
  - Graph Convolutional Networks (spectral and spatial)
  - Weisfeiler-Lehman GNNs
  - Graph Transformer & Graph ViT
  - Graph generation & molecular science
  - Material design
  - Integrating GNNs and LLMs
  - Deep Learning for Combinatorial optimization

Outline

- Recommendation systems
- Google PageRank
- Collaborative recommendation
- Content recommendation
- Hybrid systems
- Conclusion

Outline

- **Recommendation systems**
- Google PageRank
- Collaborative recommendation
- Content recommendation
- Hybrid systems
- Conclusion

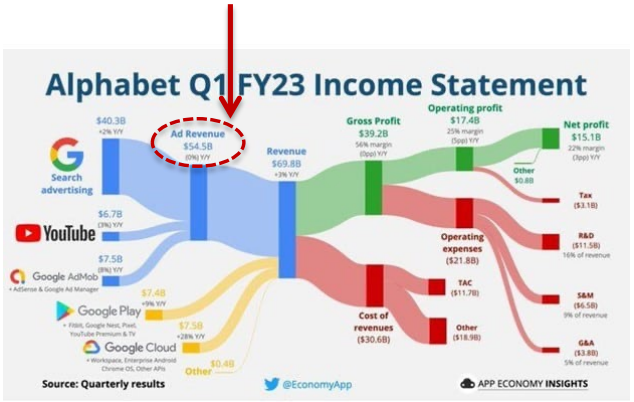
Recommender systems

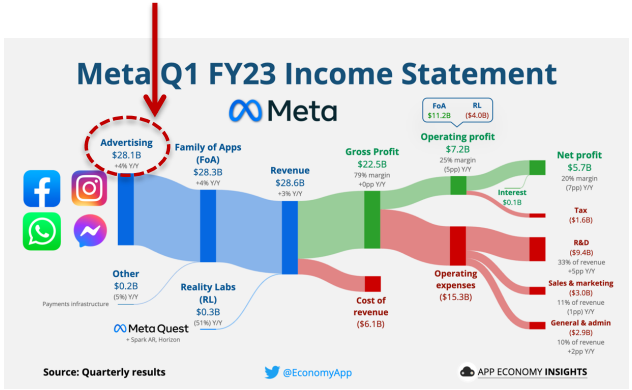
- Recommendation is the task of providing the most relevant items to a user.
- It is one of the most common tasks in machine learning.
- Arguably, the most essential task generating revenues for IT companies :
  - Google : Recommends webpages (e.g. PageRank) and products (Google Ads).
  - Meta/Facebook : Recommends products to users (Facebook Ads).
  - Amazon : Product recommendation to customers.
  - Netflix : Movie recommendations to encourage subscription renewals.

Google Ads

facebook Ads

amazon advertising





Recommender systems

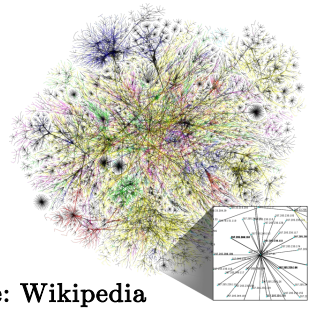
- Recommender systems make use of the following elements :
  - Product features
  - User features
  - Historical and possibly dynamic ratings
  - Relationships (i.e. graphs) between products, users and pairs of product-user
- In this lecture, we will cover recommendation techniques that are defined by :
  - Graphs exclusively : PageRank and content recommendation
  - Features exclusively : Low-rank recommendation, a.k.a. collaborative recommendation
  - Both features and graphs simultaneously : Hybrid systems

Outline

- Recommendation systems
- **Google PageRank**
- Collaborative recommendation
- Content recommendation
- Hybrid systems
- Conclusion

PageRank

- Google PageRank[1] is a ranking algorithm that scores the nodes of an arbitrary graph depending on their popularity, which is determined by the number of incoming edges.
- PageRank was applied to Internet in 1998, where the nodes represent webpages.
- This algorithm revolutionized webpage recommendations on the Internet.
- Previously, recommendations were made by humans using arbitrary and often biased criteria.
- Additionally, human-based recommendations are not scalable for large-scale graphs, s.a. Internet :
  - In 1998, the Internet had approximately 2.4 million webpages.
  - In 2023, the number of webpages is estimated to be around 30 billion !



Source: Wikipedia

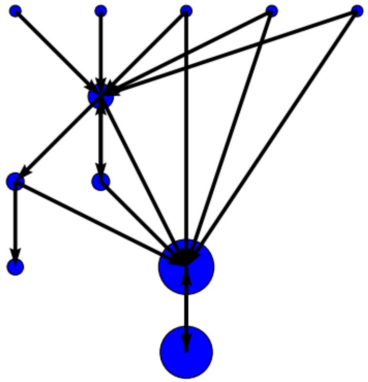
Source: Google

Larry Page and Sergey Brin, 1998

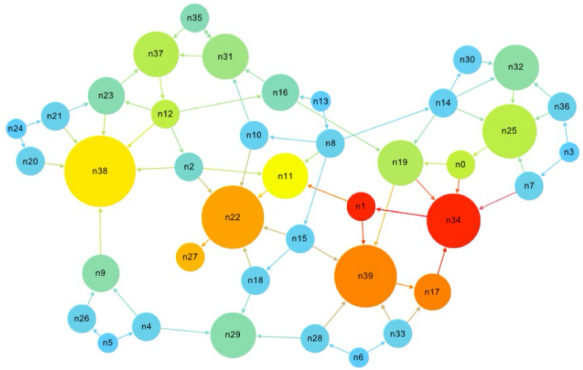
[1] Page, Brin, Motwani, Winograd, The PageRank citation ranking: Bringing order to the web, 1999 (20,000 citations)

PageRank

- The popularity of a node is simply defined by the number of incoming edges.
- There is no need to define complex and arbitrary features explicitly. If a page is popular for some reasons, several other pages will link to it.
- It turns out that ranking the nodes of a graph based on the number of incoming edges is a mathematically well-defined task.
- The solution is given by the stationary state of a directed graph G, which encodes the importance of the nodes in terms of their connectivity influence.



Directed graph and PageRank solution in bleu with the ball size related to the PR value.



Directed graph and the ball size related to the PageRank value.

Stationary state

- **The stationary state and the modes of vibration of a graph G are determined by the eigenvectors of its adjacency matrix A, which are computed through eigenvalue decomposition (EVD) :**

$$A = U\Lambda U^T \in \mathbb{R}^{n\times n} \text{ with}$$

$$U = [u_1, ..., u_n] \in \mathbb{R}^{n\times n} \quad (n \text{ eigenvectors in } \mathbb{R}^n)$$

$$U^T U = I_n, \text{ i.e. } \langle u_k, u_{k'} \rangle = \begin{cases} 1 & k = k' \\ 0 & \text{otherwise} \end{cases},$$

$$\Lambda = \text{diag}(\lambda_1, ..., \lambda_n) \in \mathbb{R}^{n\times n} \quad (\text{eigenvalues})$$

$$\lambda_{\max} = \lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$$

$$Au_k = \lambda_k u_k \in \mathbb{R}^n, \ 1 \leq k \leq n$$

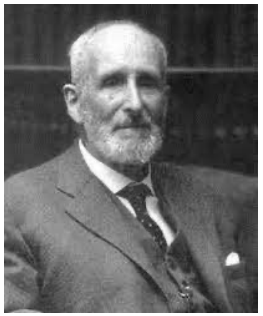
We are interested into the largest eigenvalue :

$$Au_{\max} = \lambda_{\max} u_{\max} \in \mathbb{R}^n$$

∵ A is by construction, always square

Perron-Frobenius theorem

- PF theorem[1,2] :
  - For a real square matrix A with positive entries, the largest eigenvector exists, is unique (up to multiplicity), and has strictly positive elements.
- PF theorem when applied to stochastic matrices :
  - Given a real square matrix A that is row stochastic (each row sums to 1), the PF theorem asserts that the largest eigenvector is unique, has an eigenvalue of 1, and contains strictly positive elements.



Oskar Perron
1880-1975

Georg Frobenius
1849-1917

[1] Perron, Zur Theorie der Matrices, 1907
[2] Frobenius, Ueber Matrizen aus nicht negativen Elementen, 1912

Stationary equation

- PF theorem when applied to stochastic and irreducible matrices (defined in next slide) :
  - For a directed graph G = (V,E,A) defined by a stochastic and irreducible adjacency matrix A, the PF theorem states that the largest left eigenvector is unique, has an eigenvalue of 1, and contains strictly positive elements :

$$u_{\max}^T A = u_{\max}^T \lambda_{\max} = u_{\max}^T \in \mathbb{R}^{1\times n}$$

  - Since the largest left eigenvector is unique, it can also be computed by the stationary equation :

$$u^T A = u^T \in \mathbb{R}^{1\times n}$$

or using the right eigenvector equation :

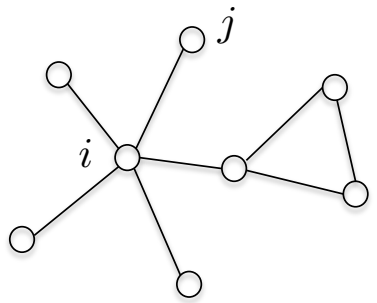
$$A^T u = u \in \mathbb{R}^n \quad (\text{with } (u^T A)^T = (u^T)^T)$$

  - The solution to the stationary state equation is the PageRank function.

Stochastic matrix

- A stochastic matrix is a matrix where each row is normalized to form a probability density function :

$$\sum_{j \in V} A_{ij} = 1 \ \forall i \ \Leftrightarrow \ A 1_n = 1_n \in \mathbb{R}^n$$



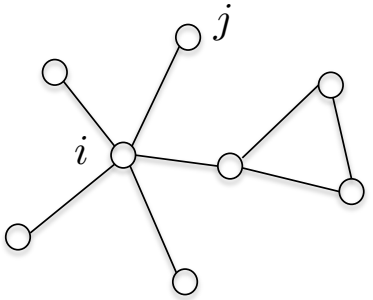
- To convert an adjacency matrix A into a stochastic matrix, we can normalize A using the degree matrix D :

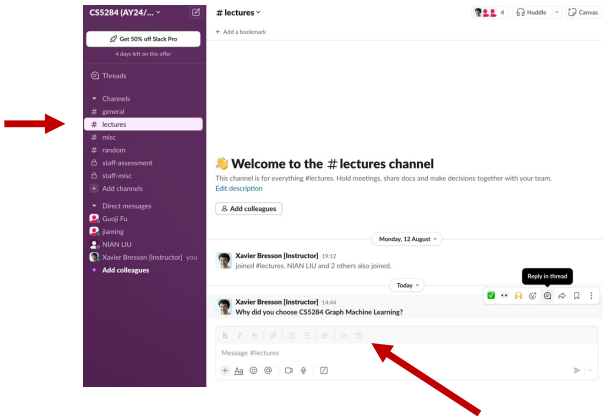
$$A \leftarrow D^{-1} A \ \text{ with } \ D^{-1}_{ii} = \begin{cases} (\sum_j A_{ij})^{-1} & \text{if } i\text{th row} \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{1}{deg(i)} = \frac{1}{\sum_j A_{ij}}$$

In-lecture question

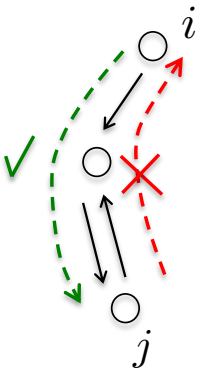
- How can you interpret the stochastic matrix A? What does A_{ij} represent, and how can you design a random walk algorithm from A? (A random walk is a path that consists of a succession of random steps on some mathematical space.)
- In Slack #lectures
  - Identify the question and Reply in thread with a short response
- Answer :





Irreducible matrix $\forall i,j \quad i \to \cdots \to j$

- A matrix is considered irreducible if it represents a graph where every node is reachable from every other node. This property is known as strong connectivity in graph theory.
  - A graph is strongly connected if, for any pair of nodes (i,j), there exists a directed path from node i to node j and a directed path from node j to node i.
  - To ensure that an adjacency matrix A is both stochastic (i.e., each row sums to 1) and irreducible, we can proceed as follows :



$$A \leftarrow \alpha D^{-1} A + (1-\alpha)\frac{\mathrm{I}_n}{n}$$

How does this ensure irreducibility ? why not $1_n 1_n^\top$?

with

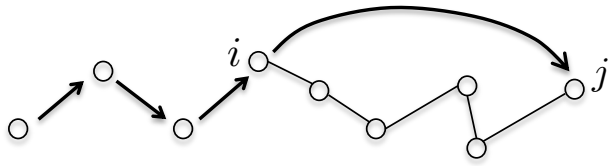
$A \in \mathbb{R}^{n \times n}$ is the original (sparse) adjacency matrix of the graph

$\mathrm{I}_n \in \mathbb{R}^{n \times n}$ is the (full) identity matrix

$\alpha \in [0,1]$, original choice is $\alpha = 0.85$[1]

[1] Page, Brin, Motwani, Winograd, The PageRank citation ranking: Bringing order to the web, 1999

Personalized PageRank



- Vanilla PageRank :
  - The term $(1\text{-}\alpha)I_n/n$ is equivalent to a user who has the freedom to jump to any webpage within the network, without specific preferences or constraints.
  - The terms $\alpha D^{-1}A + (1\text{-}\alpha)I_n/n$ models a user who follows the internet structure α% of the time (mirroring the typical browsing pattern where users click on links provided on a webpage) and for the remaining $(1\text{-}\alpha)$% of the time, decides to click to a webpage that has no connection to the previous page (due to curiosity for a different topic).
- Personalized PageRank :
  - It is feasible to further refine this model by incorporating a user's preferences or biases, such as those influenced by advertisements or personalized recommendations :

$$A \leftarrow \alpha D^{-1} A + (1-\alpha)\, p_{\text{Prior}} \in \mathbb{R}^{n \times n}$$

with

$p_{\text{Prior}} \in \mathbb{R}^{n \times n}$ is a prior probability designed by Google

$p_{\text{Prior}\,i,j} \in [0,1]$ is the prior probability of a user moving from page i to page j

Spectral algorithm

- PageRank function is given by the stationary equation $A^T u = u$, where A is the fully stochastic and irreducible matrix derived from the original (sparse) adjacency matrix of graph G.
- The most direct technique to solve $A^T u = u$ is to compute the eigenvector corresponding to the largest eigenvalue, which is known to be 1[1,2].
- Advantages of Eigenvalue Decomposition (EVD) :
  - EVD guarantees finding the exact solution.
  - The solution is independent of the initial condition.
- Limitations :
  - EVD is computationally intensive with $O(n^2)$ speed complexity and memory consuming $O(n^2)$.
  - EVD cannot be parallelizable because eigenvectors are solutions of a global linear system.
  - EVD does not scale well to large networks, s.a. internet, which contains billions of nodes.

[1] Perron, Zur Theorie der Matrices, 1907
[2] Frobenius, Ueber Matrizen aus nicht negativen Elementen, 1912

Power method

- An alternative solution is the Power method[1,2,3] :
  - The stationary state equation $A^T u = u$ can be solved using a fixed-point iterative scheme, with convergence guaranteed since the eigenvalues of A are less than or equal to 1 :

$$u^{k+1} = Au^k \in \mathbb{R}^n, \ k = 0, 1, 2, ...$$

At convergence, we have :

$$u^\star = Au^\star$$

where $u^\star \in \mathbb{R}^n$ is the PageRank function.

[1] Mises, Pollaczek-Geiringer, Praktische Verfahren der Gleichungsauflösung, 1929
[2] Page, Brin, Motwani, Winograd, The PageRank citation ranking: Bringing order to the web, 1999
[3] Nesterov, Nemirovski, Finding the stationary states of Markov chains by iterative methods, 2015

Power method

- **The power method is an iterative algorithm[1] :**

$$\text{Initialization (uniform distribution): } u^{k=0} = \frac{1_n}{n} \in \mathbb{R}^n$$

$$\text{Iterate until convergence: } \quad u^{k+1} = \alpha D^{-1} A^T u^k + (1-\alpha)\frac{1_n}{n} \in \mathbb{R}^n$$

- **Speed complexity is O(EK), where E is the number of edges in the graph and K is the number of iterations to converge to a precision ε as follows :**

$$\|x^{k+1} - x^k\|_1 \leq \varepsilon \quad \text{for} \quad K = \frac{\log_{10}\varepsilon}{\log_{10}\alpha} = \begin{cases} 85 & \text{for } \alpha = 0.85 \\ 1833 & \text{for } \alpha = 0.99 \end{cases}, \text{ and } \varepsilon = 10^{-6}$$

- **The Power Method can be parallelized due to the operation of matrix-vector multiplication.**
- **It is the optimal technique for computing the largest or smallest eigenvector of a linear operator.**

[1] Page, Brin, Motwani, Winograd, The PageRank citation ranking: Bringing order to the web, 1999

CS425 Lect 10

Power Iteration Method

- **Given a web graph with N nodes, where the nodes are pages and edges are hyperlinks**

Power iteration: a simple iterative scheme

- Suppose there are *N* web pages
- Initialize: $\mathbf{r}^{(0)} = [1/N,\ldots,1/N]^T$ ← all first start with equal importance
- Iterate: $\mathbf{r}^{(t+1)} = \mathbf{M} \cdot \mathbf{r}^{(t)}$
- Stop when $|\mathbf{r}^{(t+1)} - \mathbf{r}^{(t)}|_1 < \varepsilon$ stopping condition

Equivalent!

$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{d_i}$$

d_i …. out-degree of node i

$|x|_1 = \sum_{1 \le i \le N} |x_i|$ is the L_1 norm

$$\sum_{i=1}^{N} \left| r_i^{(t+1)} - r_i^{(t)} \right|$$

Can use any other vector norm, e.g., Euclidean

$$\sum_{i=1}^{N} \left(r_i^{(t+1)} - r_i^{(t)} \right)^2$$

- **Intuitive interpretation of power iteration**: each node starts with equal importance (of 1/N). During each step, each node passes its current importance along its outgoing edges, to its neighbors.

Spectral vs Power techniques

- **Memory complexity**

  EVD : $A^T u = u$ is $O(n^2)$ as A is full (irreducible property)

  Power method : $u^{k+1} = \alpha D^{-1} A^T u^k + (1-\alpha)\frac{1_n}{n}$ is $O(E)$ as A is sparse (for most real-world graphs)

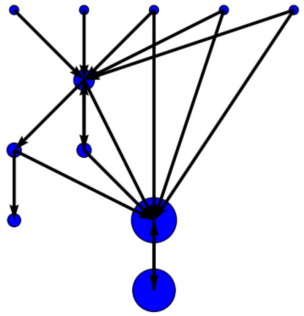
- **Speed complexity**

  EVD : $O(n^2)$ for the eigenvector with the largest eigenvalue

  Power method : $O(EK)$ (K is the number of iterations)

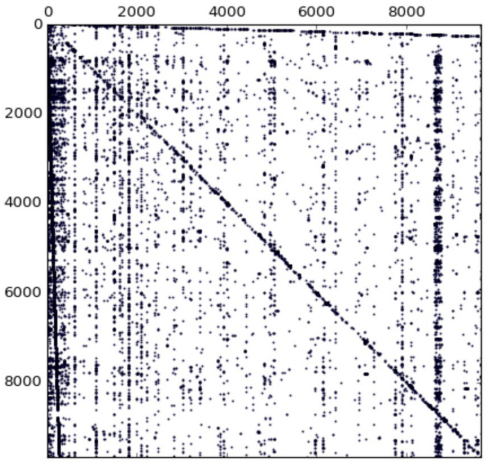
Lab 1 : PageRank

- Run code01.ipynb and conduct the following actions :
  - Compare the computational speed between the EVD and the Power method.
  - Visualize the PageRank function of a basic network.

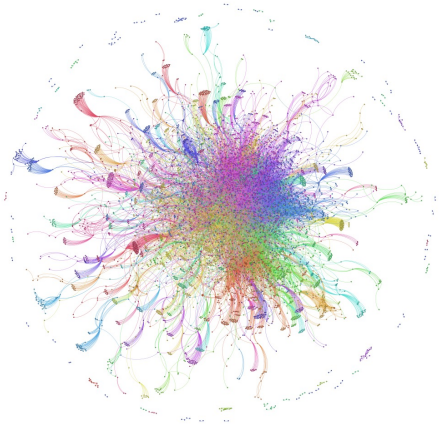


PageRank function of Artificial graph

```
# Power Method

# Initialization
x = e
diffx = 1e10
k = 0

# Iterative scheme
start = time.time()
alpha = 0.85
while (k<1000) & (diffx>1e-6):

    # Update iteration
    k += 1

    # Update x
    xold = x
    x = alpha* P.dot(x) + e.dot( alpha* a.T.dot(x) + (1.0-alpha) )

    # Stopping condition
    diffx = np.linalg.norm(x-xold,1)
```



Adjacency matric of California graph



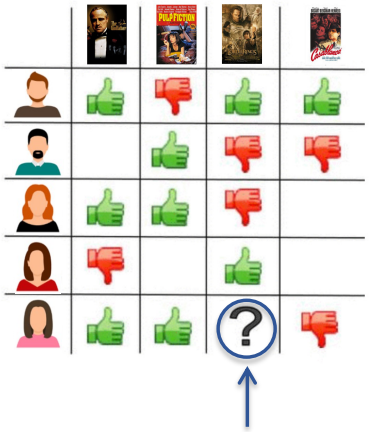
Visualization of California graph

Outline

- Recommendation systems
- Google PageRank
- **Collaborative recommendation**
- Content recommendation
- Hybrid systems
- Conclusion

Recommendation task

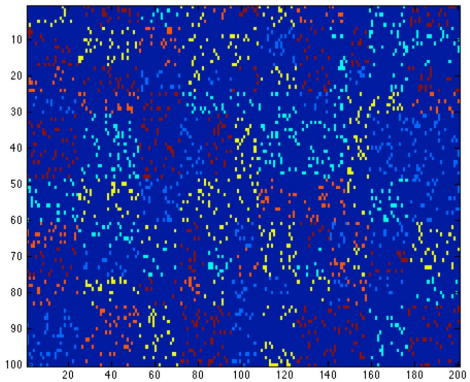
- A prominent example is the 1M$ Netflix Prize[1,2] in 2009.
- The Netflix competition aimed to develop the best algorithm for predicting user ratings based on a sparse set of previously rated movies.
- The dataset statistics
  - 480,189 users, 17,770 movies, 100,480,507 ratings ⇒ Only 0.011% of all possible ratings !
- Interestingly, the winning algorithm was never implemented by Netflix due to its complexity and high engineering costs.



[1] Bennett, Lanning, The netflix prize, 2007 (3,200 citations)
[2] https://en.wikipedia.org/wiki/Netflix_Prize

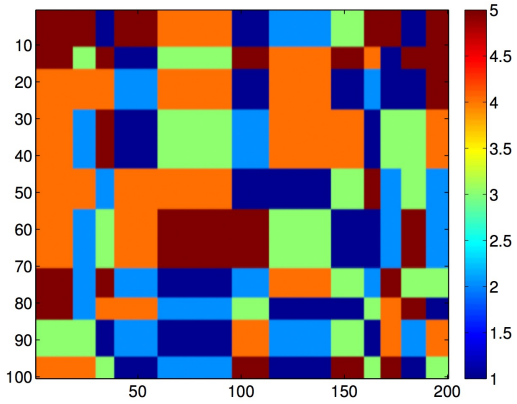
Task formalization

- Problem : Given a (small) set of ratings/observations M_{ij} ($\in [1,5]$) for movie j and user i, estimate a matrix X that best fits the given ratings.
- This problem is known as the matrix completion task.



Given sparse ratings
$M \in \mathbb{R}^{n \times m}$

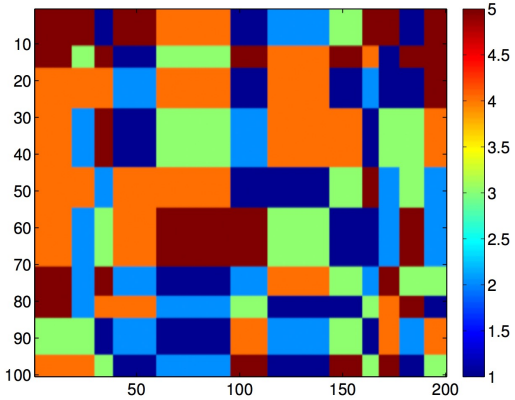
Recommendation
=
Matrix completion



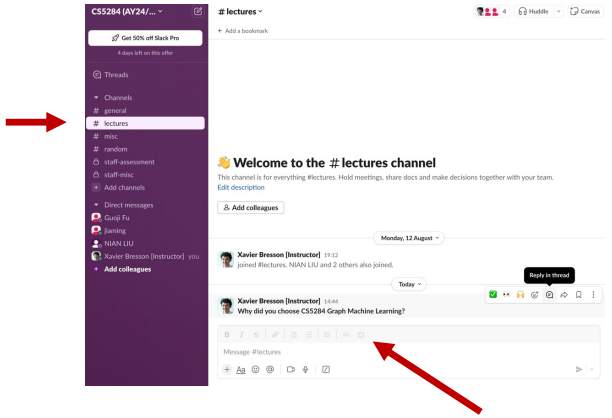
Fully estimated ratings
$X \in \mathbb{R}^{n \times m}$

In-lecture question

- When examining the ground truth matrix of ratings, what specific properties can you identify? Additionally, what kind of invariance is required when designing a recommender system?
- In Slack #lectures
  - Identify the question and Reply in thread with a short response
- Answer :

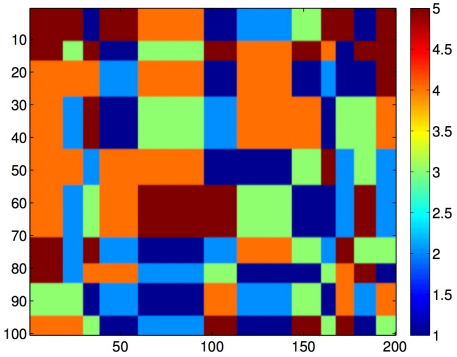


Ground truth ratings



Collaborative recommendation

- We assume that the rating matrix is low-rank.
- The rank of a matrix is defined as the number of its independent rows or columns.
- The low-rank hypothesis has proven valid for several real-world datasets.
  - For example, Netflix :
    - There exist communities of users who rate movies similarly.
    - There are groups of movies that receive the same ratings.
  - Same assumptions for Amazon (users, products), LinkedIn (users, jobs), Facebook (users, ads).



Low-rank matrix
$X \in \mathbb{R}^{100 \times 200}$

linearly independent rows = 13
linearly independent cols = 15
$\Rightarrow$ rank(X) = max(13,15) = 15

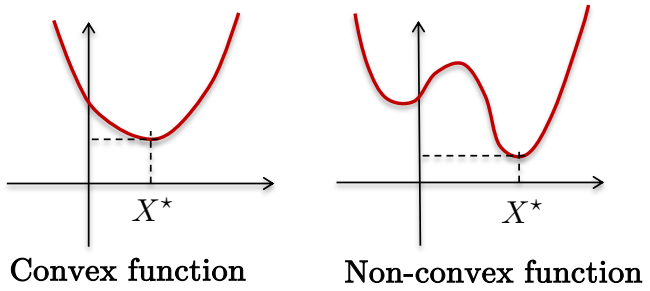
Low-rank modeling

- **Approximating a low-rank matrix from the original sparse observation matrix is typically achieved through optimization techniques :**

$$\min_{X \in \mathbb{R}^{n \times m}} \operatorname{rank}(X) \quad \text{s.t.} \quad \begin{cases} X_{ij} = M_{ij} & \forall ij \in \Omega_{\text{obs}} \text{ (noiseless case)} \\ & \text{(observations are clean)} \\ X_{ij} = M_{ij} + n_{ij} & \forall ij \in \Omega_{\text{obs}} \text{ (noisy case)} \\ & \text{(observations are corrupted)} \end{cases}$$

This combinatorial NP-hard problem can be relaxed using two continuous approaches :

1) Convex relaxation with nuclear norm[1].
2) Non-convex relaxation with matrix factorization[2,3,4].



Convex function | Non-convex function

[1] Candes, Recht, Exact matrix completion via convex optimization, 2009
[2] Luo, Zhou, Xia, Zhu, An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems, 2014
[3] Sarwar, Karypis, Konstan, Riedl, Application of dimensionality reduction in recommender system-a case study, 2000
[4] Seung, Algorithms for non-negative matrix factorization, 2000

Unconstrained low-rank optimization

- **We further relax the original constrained optimization problem with an unconstrained problem :**

$$\min_{X \in \mathbb{R}^{n \times m}} \ \text{rank}(X) \ \text{ s.t. } X_{ij} = M_{ij} + n_{ij} \ \forall ij \in \Omega_{\text{obs}}$$

which is equivalent to

$$\min_{X \in \mathbb{R}^{n \times m}} \ \text{rank}(X) + \frac{\lambda}{2} \|\text{Ind}_{\text{obs}} \circ (X - M)\|_F^2 \ \text{ where } (\text{Ind}_{\text{obs}})_{ij} = \begin{cases} 1 & \text{if } ij \in \Omega_{\text{obs}} \\ 0 & \text{otherwise} \end{cases}$$

where

$\min_X \ \text{rank}(X)$ promotes the low-rank property.

$\|\text{Ind}_{\text{obs}} \circ (X - M)\|_F^2$ enforces data fidelity and robustness to perturbation.

Operator $\circ$ is the Hadamard point-wise product.

Constant $\lambda > 0$ controls the trade-off low-rank vs. fidelity.

There exists a closed-form formula between λ and n for the Gaussian noise (Lagrange multiplier[1]).

[1] Boyd, Vandenberghe, Convex optimization, 2004 (75,000 citations)

Convex relaxation

- **The unconstrained NP-hard combinatorial problem is intractable and in its original form and therefore must be relaxed :**

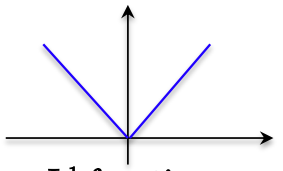
$$\min_{X \in \mathbb{R}^{n \times m}} \operatorname{rank}(X) + \frac{\lambda}{2} \|\mathrm{Ind}_{\mathrm{obs}} \circ (X - M)\|_F^2 \quad \text{where } (\mathrm{Ind}_{\mathrm{obs}})_{ij} = \begin{cases} 1 & \text{if } ij \in \Omega_{\mathrm{obs}} \\ 0 & \text{otherwise} \end{cases}$$

An exact continuous, non-smooth and convex relaxation was proposed within the compressed sensing framework[1,2,3] :

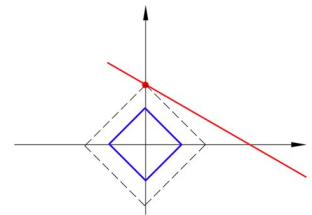
$$\min_{X \in \mathbb{R}^{n \times m}} \|X\|_\star + \frac{\lambda}{2} \|\mathrm{Ind}_{\mathrm{obs}} \circ (X - M)\|_F^2 \quad \text{where } \|\cdot\|_\star \text{ is the nuclear norm defined as}$$

$$\|X\|_\star = \|\Sigma\|_1 = \sum_{k=1}^{p} |\sigma_k(X)|, \text{where } \sigma_k(X) \text{ are the singular values of } X \text{ s.t.}$$

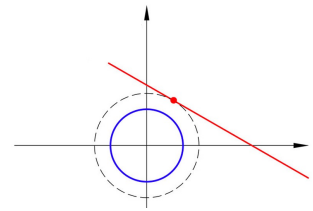
$$X = U \Sigma V^T, \; U \in \mathbb{R}^{n \times p}, \Sigma = \mathrm{diag}(\sigma_1, ..., \sigma_p) \in \mathbb{R}^{p \times p}, V \in \mathbb{R}^{p \times m}, p = \min(n, m)$$



L^1 function

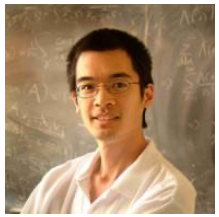


min L^1 s.t. X = M



min L^2 s.t. X = M

[1] Candes, Romberg, Tao, Exact signal reconstruction, 2006 (19,000 citations)
[2] Donoho, Compressed sensing, 2006 (35,000 citations)
[3] Candes, Recht, Exact matrix completion via convex optimization, 2009



Terrence Tao

Emmanuel Candes

David Donoho

Primal-dual optimization

- **We can design an algorithm that is guaranteed to converge with a rate of O(1/k)[1,4], where k is the number of iterations :**

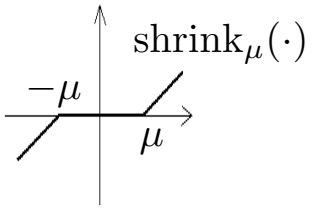
$$\text{Initialization : } X^{k=0} = M \in \mathbb{R}^{n \times m}, Y^{k=0} = 0 \in \mathbb{R}^{n \times m}$$

$$\text{Iterate until convergence, i.e. k=0,1,..}$$

$$Y^{k+1} = Y^k - \sigma^k U \text{shrink}_{1/\sigma^k}(\Sigma) V^T \in \mathbb{R}^{n \times m} \text{ (dual variable)}$$

$$\text{with SVD : } Y^k + \sigma^k X^k = U \Sigma V^T \in \mathbb{R}^{n \times m}$$

$$X^{k+1} = \frac{X^k - \tau^k Y^{k+1} + \tau^k \lambda M}{1 + \tau^k \lambda \text{Ind}_{\text{obs}}} \in \mathbb{R}^{n \times m} \text{ (primal variable)}$$



Shrinkage operator[2]

[1] Candes, Recht, Exact matrix completion via convex optimization, 2009 (7,000 citations)
[2] Donoho, De-noising by soft-thresholding, 1995 (15,000 citations)
[3] Liu, Vandenberghe, Interior-point method for nuclear norm approximation with application to system identification, 2010
[4] Chambolle, Pock, A first-order primal-dual algorithm for convex problems with applications to imaging, 2010 (5,000 citations)

Algorithmic properties

- Advantages
  - The solution is unique, exact, and stable, regardless of the initialization.

- Limitations
  - Time complexity is dominated by Singular Value Decomposition (SVD), which is $O(n^3)$.
  - The memory requirement is $O(n^2)$.
  - SVD-based algorithms cannot scale to large datasets, i.e. for n significantly large.

Non-convex relaxation

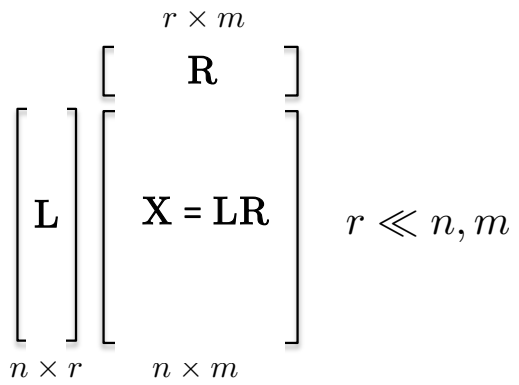
- **The unconstrained NP-hard combinatorial problem is intractable and in its original form and therefore must be relaxed :**

$$\min_{X \in \mathbb{R}^{n \times m}} \operatorname{rank}(X) + \frac{\lambda}{2} \|\mathrm{Ind}_{\mathrm{obs}} \circ (X - M)\|_F^2 \quad \text{where } (\mathrm{Ind}_{\mathrm{obs}})_{ij} = \begin{cases} 1 & \text{if } ij \in \Omega_{\mathrm{obs}} \\ 0 & \text{otherwise} \end{cases}$$

Approximate, continuous and smooth relaxations were proposed within matrix factorization techniques s.a. NMF[1,2,3] (non-negative matrix factorization) :

$$\min_{L \in \mathbb{R}^{n \times r}, R \in \mathbb{R}^{r \times m}} \frac{1}{2}\|L\|_F^2 + \frac{1}{2}\|R\|_F^2 + \frac{\lambda}{2}\|\mathrm{Ind}_{\mathrm{obs}} \circ (LR - M)\|_F^2$$

$$\text{where } (\mathrm{Ind}_{\mathrm{obs}})_{ij} = \begin{cases} 1 & \text{if } ij \in \Omega_{\mathrm{obs}} \\ 0 & \text{otherwise} \end{cases} \quad \text{and } r \ll n, m.$$



[1] Luo, Zhou, Xia, Zhu, An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems, 2014
[2] Sarwar, Karypis, Konstan, Riedl, Application of dimensionality reduction in recommender system-a case study, 2000
[3] Lee, Seung, Algorithms for non-negative matrix factorization, 2000 (12,000 citations)

Algorithmic properties

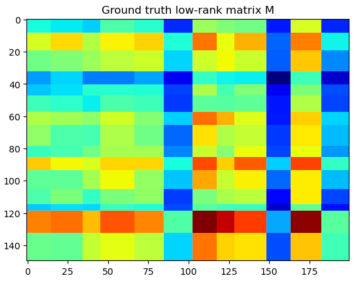
- Advantages
  - Optimization problem is smooth and quadratic : Enables fast optimization techniques with e.g. conjugate gradient[1], Newton's method, etc.
  - The objective is differentiable : Facilitates the application of stochastic gradient descent (SGD) technique for large-scale recommender systems[2].
  - Monotonicity property : Factorization algorithms s.a. NMF[3,4] ensure a monotonic decrease in loss over iterations, $E^{k+1} \leq E^k \ \forall k$.
- Limitations
  - Non-convex optimization : Prone to getting trapped in local minimizers, making good initialization critical.
  - Introduction of a new hyper-parameter : The rank value r requires to be evaluated.

[1] Boyd, Vandenberghe, Convex optimization, 2004
[2] Gemulla, Nijkamp, Haas, Sismanis, Large-scale matrix factorization with distributed stochastic gradient descent, 2011
[3] Sarwar, Karypis, Konstan, Riedl, Application of dimensionality reduction in recommender system-a case study, 2000
[4] Lee, Seung, Algorithms for non-negative matrix factorization, 2000

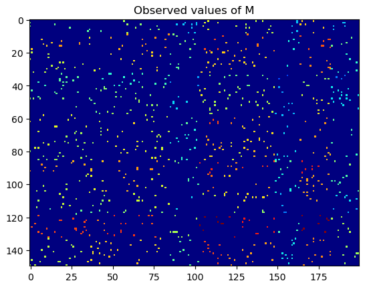
Lab 2 : Collaborative filtering

- Execute code02.ipynb and perform the following tasks:
  - Compute the low-rank solution.
  - Visualize reconstruction process.

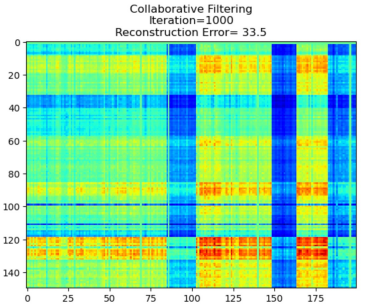
```
# Collaborative filtering / low-rank approximation by nuclear norm

# Indentify zero columns and zero rows in the data matrix X
idx_zero_cols = np.where(np.sum(Otraining,axis=0)<1e-9)[0]
idx_zero_rows = np.where(np.sum(Otraining,axis=1)<1e-9)[0]
nb_zero_cols = len(idx_zero_cols)
nb_zero_rows = len(idx_zero_rows)

# Regularization parameter
OM = O*M
normOM = np.linalg.norm(OM,2)
lambdaNuc = normOM/4.
lambdaDF = 1e3

# Initialization
X = M; Xb = X;
Y = np.zeros([n,m])
normA = 1.
sigma = 1./normA
tau = 1./normA
diffX = 1e10
min_nm = np.min([n,m])
k = 0
while (k<1000) & (diffX>1e-2):

    # Update iteration
    k += 1

    # Update dual variable y
    Y = Y + sigma* Xb
    U,S,V = np.linalg.svd(Y/sigma)
    Sdiag = shrink( S , lambdaNuc/ sigma )
    I = np.array(range(min_nm))
    Sshrink = np.zeros([n,m])
    Sshrink[I,I] = Sdiag
    Y = Y - sigma* U.dot(Sshrink.dot(V))

    # Update primal variable x
    Xold = X
    X = X - tau* Y
    X = ( X + tau* lambdaDF* O* M)/ (1 + tau* lambdaDF* O)
    # Fix issue with no observations along some rows and columns
    r,c = np.where(X>0.0); median = np.median(X[r,c])
    if nb_zero_cols>0: X[:,idx_zero_cols] = median
    if nb_zero_rows>0: X[nb_zero_rows,:] = median

    # Update primal variable xb
    Xb = 2.* X - Xold
```



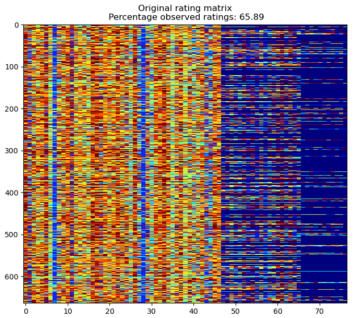
Ground-truth ratings
Artificial dataset



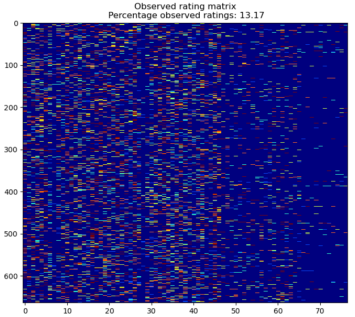
Available ratings



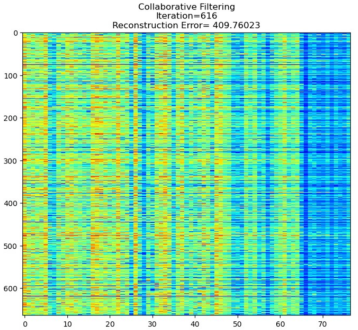
Low-rank solution



Ground-truth ratings
Real-world dataset



Available ratings



Low-rank solution

Outline

- Recommendation systems
- Google PageRank
- Collaborative recommendation
- **Content recommendation**
- Hybrid systems
- Conclusion

Content recommendation

- Content filtering recommendation[1] leverage similarities between users and between items to predict ratings.
- Task formulation
  - Given a (small) set of ratings/observations M_{ij} for movie j and user i, along with a set of user features and product features, estimate a matrix X that best fits the provided ratings while maintaining similarity among users and products.
- Examples
  - User features/attributes : Gender, age, job, occupation, interests, etc.
  - Product features/attributes : Domain, price, release date, size, etc.

Users with their features

Products with their features

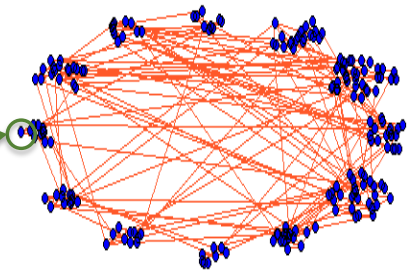
[1] Pazzani, Billsus, Content-based recommendation systems, 2007 (4,000 citations)

Encoding similarity

- **Graphs serve as a universal representation of similarity relationships between users and between products :**

Network of users

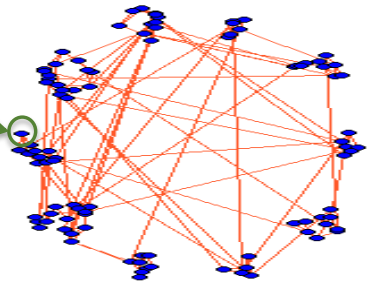
=



$G_r = (V_r, E_r, A_r)$
Row/user graph with
A_r the adjacency matrix of rows

Network of products

=

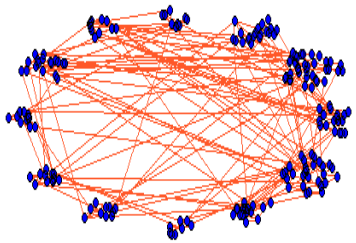


$G_c = (V_c, E_c, A_c)$
Column/product graph with
A_c the adjacency matrix of columns

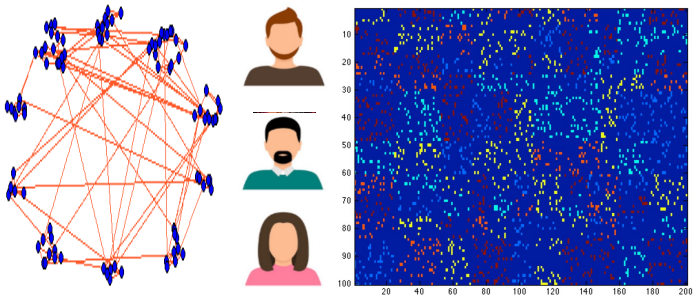
Task formalization

- **Graph-based content filtering[1] :**

$G_c = (V_c, E_c, A_c)$
Column/product graph

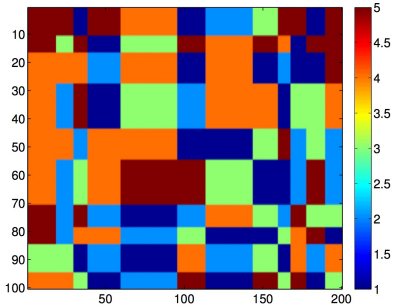


$G_r = (V_r, E_r, A_r)$
Row/user graph



Given sparse ratings
$M \in \mathbb{R}^{n \times m}$

Collaborative Recommendation = Graph diffusion



Fully estimated ratings
$X \in \mathbb{R}^{n \times m}$

[1] Huang, Chung, Ong, Chen, A graph-based recommender system for digital library, 2002

Graph recommendation

- **We can fill out the sparse matrix M with the user and product graphs by diffusing/smoothing the available ratings across these graphs.**
- **Diffusion on graphs can be formulated as a smooth optimization problem :**

$$\min_{X \in \mathbb{R}^{n \times m}} \|X\|_{\mathrm{G}_r}^{\mathrm{diffusion}} + \|X\|_{\mathrm{G}_c}^{\mathrm{diffusion}} + \frac{\lambda}{2} \|\mathrm{Ind}_{\mathrm{obs}} \circ (X - M)\|_F^2$$

where

$$(\mathrm{Ind}_{\mathrm{obs}})_{ij} = \begin{cases} 1 & \text{if } ij \in \Omega_{\mathrm{obs}} \\ 0 & \text{otherwise} \end{cases}$$

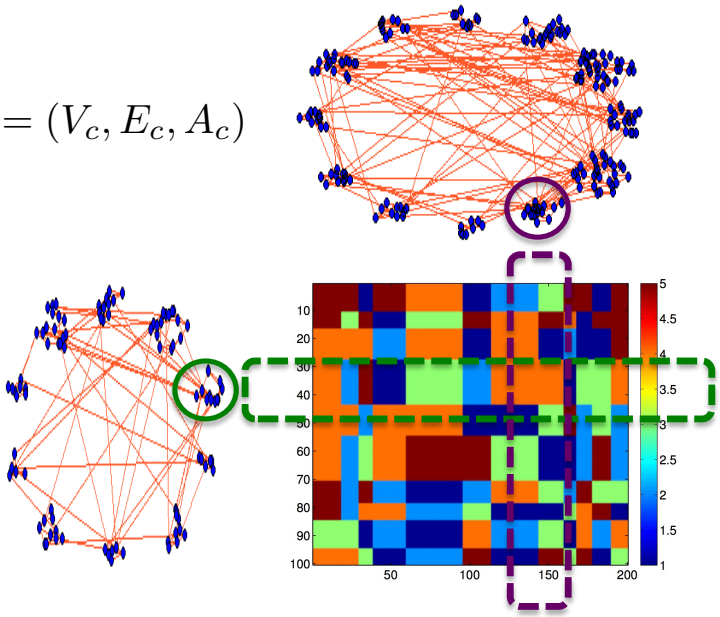
$\|X\|_{\mathrm{G}}^{\mathrm{diffusion}}$ is a diffusion loss on a graph G.

Modeling diffusion on graphs

- Assumption : When a user i is closely connected to another user i' on the graph $\mathrm{G}_{\mathrm{users}}$ (i.e. G_{r}), it suggests that these two users share similarities, likely having similar interests and product preferences. Consequently, it can be anticipated that these users will rate products similarly.
- This implies that the row_i and the $\mathrm{row}_{i'}$ in matrix X are expected to be similar.

$G_c = (V_c, E_c, A_c)$

$G_r = (V_r, E_r, A_r)$



Modeling diffusion on graphs

- The diffusion loss should enforce smoothness of observations/ratings across the graph.
- The Dirichlet norm[1] is widely favored as the default measure of graph smoothness :

$$\|X\|_{\mathrm{G}}^{\mathrm{diffusion}} = \|X\|_{\mathrm{G}}^{\mathrm{Dir}} = \mathrm{tr}(X^T L X) \in \mathbb{R}_+, \ X \in \mathbb{R}^{n \times m}$$

where

$$L = \mathrm{I}_n - D^{-1/2} A D^{-1/2} \in \mathbb{R}^{n \times n} \quad \text{(graph Laplacian)}$$

$$\begin{aligned}
\mathrm{tr}(X^T L X) &= \sum_{k=1}^{m} x_k^T L x_k, \ x_k \in \mathbb{R}^n \\
&= \sum_{k=1}^{m} \sum_{i,j \in V} (x_k)_i L_{ij} (x_k)_j \\
&= \sum_{k=1}^{m} \sum_{i,j \in V} A_{ij} ((x_k)_i - (x_k)_j)^2
\end{aligned}$$

Johann Dirichlet
1805-1859

[1] Chung, Spectral graph theory, 1997

Least squares optimization

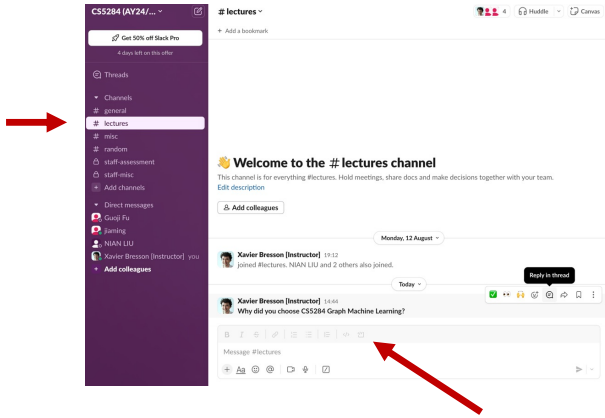
- **Least squares problem with graph regularization :**

$$\min_{X \in \mathbb{R}^{n \times m}} \|X\|_{\mathrm{G}_r}^{\mathrm{Dir}} + \|X\|_{\mathrm{G}_c}^{\mathrm{Dir}} + \frac{\lambda}{2} \|\mathrm{Ind}_{\mathrm{obs}} \circ (X - M)\|_F^2$$

$$\min_{X \in \mathbb{R}^{n \times m}} \mathrm{tr}(X^T L_r X) + \mathrm{tr}(X L_c X^T) + \frac{\lambda}{2} \|\mathrm{Ind}_{\mathrm{obs}} \circ (X - M)\|_F^2$$

In-lecture question

- How can you solve a standard least squares optimization problem, $\min_x || Bx - y ||_2^2$? If matrix B is ill-conditioned (i.e., B is not invertible, making the solution unstable), how can you compute the least squares solution?
- In Slack #lectures
  - Identify the question and Reply in thread with a short response
- Answer :



Least squares optimization

- **Least squares problem with graph regularization :**

$$\min_{X \in \mathbb{R}^{n \times m}} \|X\|_{\mathrm{G}_r}^{\mathrm{Dir}} + \|X\|_{\mathrm{G}_c}^{\mathrm{Dir}} + \frac{\lambda}{2}\|\mathrm{Ind}_{\mathrm{obs}} \circ (X - M)\|_F^2$$

$$\min_{X \in \mathbb{R}^{n \times m}} \mathrm{tr}(X^T L_r X) + \mathrm{tr}(X L_c X^T) + \frac{\lambda}{2}\|\mathrm{Ind}_{\mathrm{obs}} \circ (X - M)\|_F^2$$

- **The optimization problem is smooth and quadratic.**
- **It can be reduced to a linear system of equations, represented as Ax = b.**
- **Techniques s.a. conjugate gradient or stochastic gradient descent are applicable :**

$$(\mathrm{I}_m \otimes L_r + L_c \otimes \mathrm{I}_n + \lambda \mathrm{O}_{mn})x = \lambda m$$

with $x = \mathrm{reshape}(X)$, $m = \mathrm{reshape}(M) \in \mathbb{R}^{nm}$, $\otimes$ is the tensor product

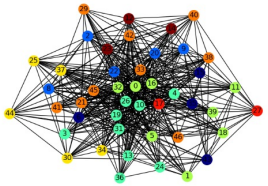
$$Ax = b$$

$$x = A^{-1}b \in \mathbb{R}^{nm}$$

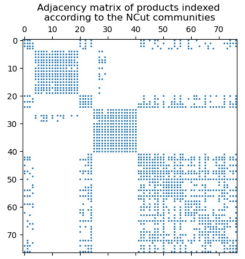
$$X = \mathrm{reshape}(x) \in \mathbb{R}^{n \times m}$$

Lab 3 : Content filtering

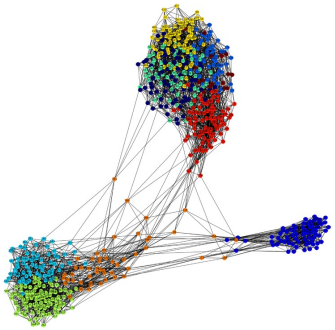
- Run code03.ipynb and perform the following tasks :
  - Compute the graph-based solution.
  - Visualize the solution.



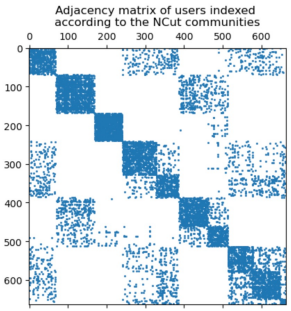
Graph of products

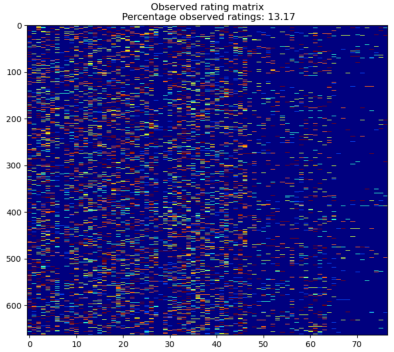



```
# Content Filtering / Graph Regularization by Dirichlet Energy

# Compute Graph Laplacians
Lr = graph_laplacian(Wrow)
Lc = graph_laplacian(Wcol)
I = scipy.sparse.identity(m, dtype=Lr.dtype)
Lr = scipy.sparse.kron( I, Lr )
Lr = scipy.sparse.csr_matrix(Lr)
I = scipy.sparse.identity(n, dtype=Lc.dtype)
Lc = scipy.sparse.kron( Lc, I )
Lc = scipy.sparse.csr_matrix(Lc)

# Regularization parameter
lambdaDir = 1e-2*8
lambdaDF = 1e3
alpha = 0.1

# Pre-processing
L = alpha* Lc + (1.-alpha)* Lr
vec0 = np.reshape(0.T,[-1])
vec0 = scipy.sparse.diags(vec0, 0, shape=(n*m, n*m) ,dtype=L.dtype)
vec0 = scipy.sparse.csr_matrix(vec0)
At = lambdaDir* L + lambdaDF* vec0
vecM = np.reshape(M.T,[-1])
bt = lambdaDF* scipy.sparse.csr_matrix( vecM ).T
bt = np.array(bt.todense()).squeeze()

# Solve by linear system
x,_ = scipy.sparse.linalg.cg(At, bt, x0=bt, tol=1e-9, maxiter=100)
X = np.reshape(x,[m,n]).T

# Reconstruction error
err_test = np.linalg.norm(Otest*(X-Mgt))
```

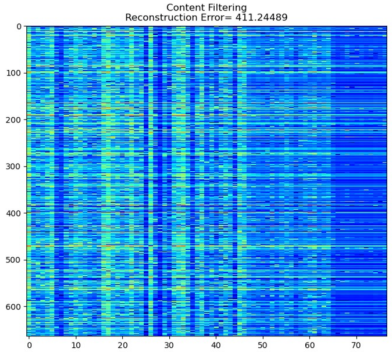


Graph of users





Available ratings
Real-world dataset



Graph-based
smooth solution

Outline

- Recommendation systems
- Google PageRank
- Collaborative recommendation
- Content recommendation
- **Hybrid systems**
- Conclusion

Hybrid system

- It is possible to integrate collaborative and content recommendation techniques[1].
- Task formulation : Given a limited set of ratings/observations M_{ij} for item j and user i, along with user features and item features, design a recommender system that effectively combines :
  - Collaborative filtering to capture the low-rank property of ratings.
  - Content filtering to diffuse ratings across user and product graphs, ensuring alignment with user and product similarities.

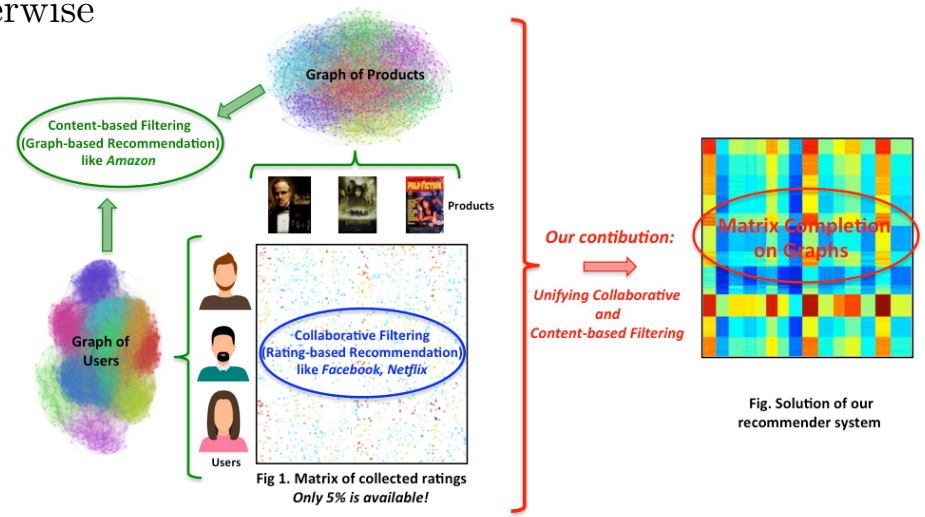
[1] Ma, Zhou, Liu, Lyu, King, Recommender systems with social regularization, 2011

Matrix completion on graphs

- A method called matrix completion on graphs[1] effectively combines these two key aspects :
  - Capturing the low-rank structure of ratings using the nuclear norm.
  - Diffusing ratings across user and product graphs by leveraging the graph Dirichlet norm.

$$\min_{X \in \mathbb{R}^{n \times m}} \|X\|_\star + \frac{\lambda_G}{2}\mathrm{tr}(X^T L_r X) + \frac{\lambda_G}{2}\mathrm{tr}(X L_c X^T) + \frac{\lambda}{2}\|\mathrm{Ind}_{\mathrm{obs}} \circ (X - M)\|_F^2$$

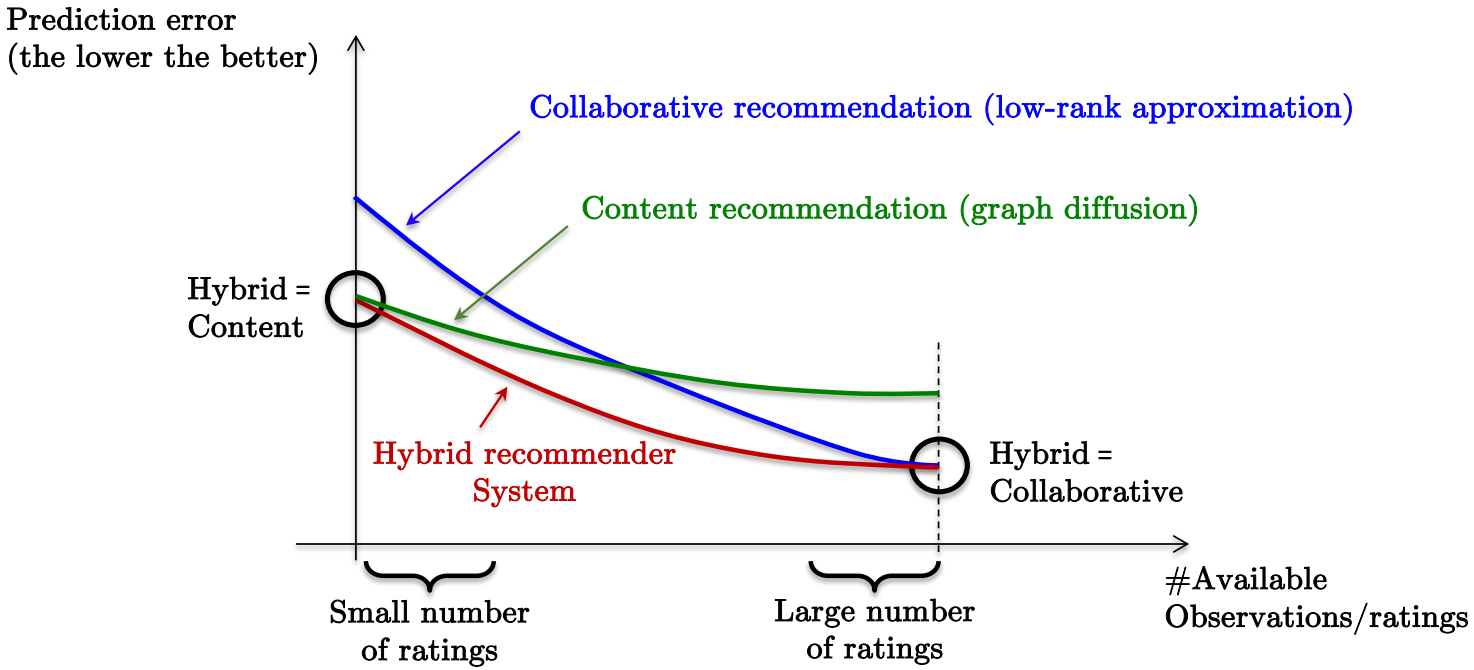
$$\text{where } (\mathrm{Ind}_{\mathrm{obs}})_{ij} = \begin{cases} 1 & \text{if } ij \in \Omega_{\mathrm{obs}} \\ 0 & \text{otherwise} \end{cases}$$



[1] Kalofolias, Bresson, Bronstein, Vandergheynst, Matrix completion on graphs, 2014

Sample complexity vs performance

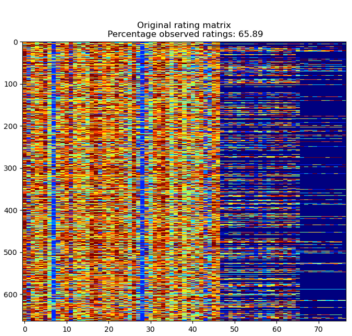
- Building a recommender system :
  - In the absence of sufficient ratings, prioritize the collection of data features.
  - Once an adequate number of ratings have collected, reduce the need on (hand-crafted) features.



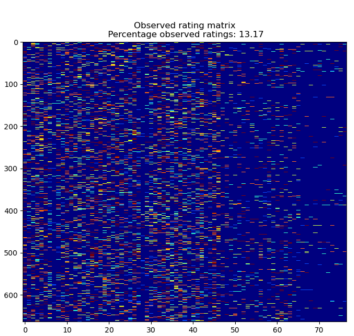
Lab 4 : Hybrid recommendation

- Run code04.ipynb and perform the following tasks :
  - Compute the hybrid solution.
  - Visualize the obtained solution.

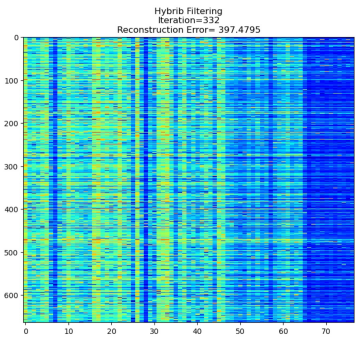
```
# Hybrid system : Matrix Completion on graphs

#Compute Graph Laplacians
Lr = graph_laplacian(Wrow)
Lc = graph_laplacian(Wcol)
I = scipy.sparse.identity(m, dtype=Lr.dtype)
Lr = scipy.sparse.kron( I, Lr )
Lr = scipy.sparse.csr_matrix(Lr)
I = scipy.sparse.identity(n, dtype=Lc.dtype)
Lc = scipy.sparse.kron( Lc, I )
Lc = scipy.sparse.csr_matrix(Lc)

# Indentify zero columns and zero rows in the data matrix X
idx_zero_cols = np.where(np.sum(Otraining,axis=0)<1e-9)[0]
idx_zero_rows = np.where(np.sum(Otraining,axis=1)<1e-9)[0]
nb_zero_cols = len(idx_zero_cols)
nb_zero_rows = len(idx_zero_rows)

# Regularization parameter
OM = O*M
normOM = np.linalg.norm(OM,2)
lambdaNuc = normOM/4.

# Regularization parameter
lambdaDir = 1e-2*8
lambdaDF = 1e3
alpha = 0.1

# Pre-processing
L = alpha* Lc + (1.-alpha)* Lr
vecO = np.reshape(O.T,[-1])
vecO = scipy.sparse.diags(vecO, 0, shape=(n*m, n*m) ,dtype=L.dtype)
vecO = scipy.sparse.csr_matrix(vecO)
At = lambdaDir* L + lambdaDF* vecO
vecM = np.reshape(M.T,[-1])
bt = lambdaDF* scipy.sparse.csr_matrix( vecM ).T
bt = np.array(bt.todense()).squeeze()
Id = scipy.sparse.identity(n*m)
Id = scipy.sparse.csr_matrix(Id)

# Initialization
X = M; Xb = X;
Y = np.zeros([n,m])
normA = 1.
```



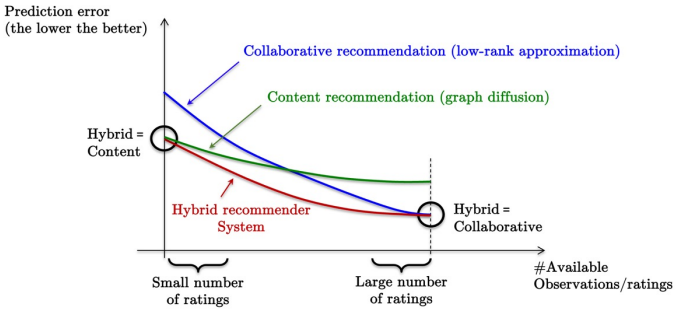
Ground-truth ratings
Real-world dataset



Available ratings



Hybrid solution



Scenario 1 : Low number of ratings (1.3%)
- low-rank/collab sol : 800.25
- graph/content sol : 399.89
- hybrid sol : 402.71 (≈ graph sol)

Scenario 2 : Good number of ratings (13%)
- low-rank/collab sol : 409.76
- graph/content sol : 411.24
- hybrid sol : 397.47

Scenario 3 : High number of ratings (52%)
- low-rank/collab sol : 698.97
- graph/content sol : 748.52
- hybrid sol : 695.00 (≈ low-rank sol)

Outline

- Recommendation systems
- Google PageRank
- Collaborative recommendation
- Content recommendation
- Hybrid systems
- **Conclusion**

Conclusion

- Recommender systems operate on the premise that customers naturally form clusters based on shared interests and preferences.
- The effectiveness of recommendations typically increases with the accumulation of ratings.
- However, obtaining ratings poses challenges, as soliciting customers to rate numerous products is impractical.
- To address this, user and product features are leveraged to enhance predictions, ensuring that similar users and products receive comparable ratings.
- Deep learning techniques can enrich raw features, augmenting the system's predictive capabilities.
- Additionally, in scenarios involving temporal changes in ratings -- such as when movies transition from box office disappointments to cult classics -- capturing these dynamics is critical.
- Time series and Markov models are used to model the temporal nature of ratings, with deep learning architectures s.a. RNNs or Transformers serving as powerful tools for learning dynamic systems[1].

[1] Monti, Bronstein, Bresson, Geometric matrix completion with recurrent multi-graph neural networks, 2017

Questions?