Approximate SDP solvers, Matrix Factorizations, the Netflix Prize, and PageRank

Mittagseminar

Martin Jaggi, Oct 6 2009

Sparse Approximation

f( ) convex

The Problem

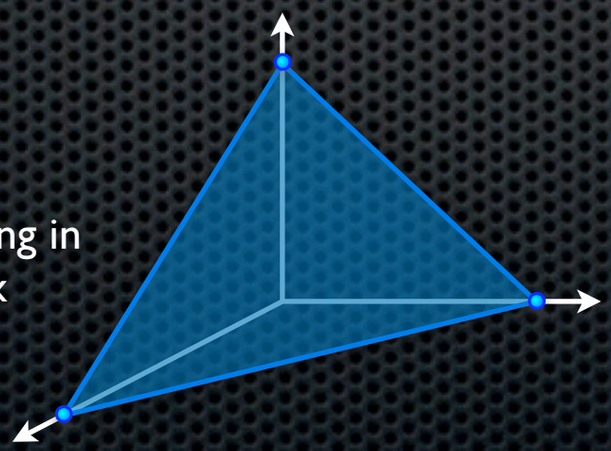
$$\min f(x)$$

$$x \in \mathbb{R}^n$$

$$x \geq 0$$

$$\mathbf{1}^T x = 1$$

vectors living in the simplex



$$\min f(X)$$

$$X \in \mathbb{S}^{n \times n}$$

$$X \succeq 0$$

$$Tr(X) = 1$$

symetric matrices living in the spectahedron

The Problem

$$\min f(x)$$

$$x \in \mathbb{R}^n$$
$$x \geq 0$$
$$\mathbf{1}^T x = 1$$

$$\min f(X)$$

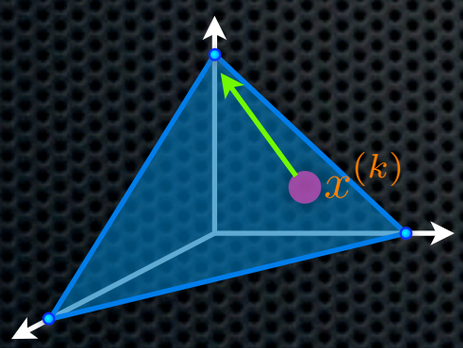
$$X \in \mathbb{S}^{n \times n}$$
$$X \succeq 0$$
$$Tr(X) = 1$$

The Algorithm

$$x^{(k+1)} := (1-\lambda)\, x^{(k)} + \lambda\, e_i$$

$$i := \arg\max_i \; -\nabla f(x^{(k)})_i$$

„Coordinate Descent“



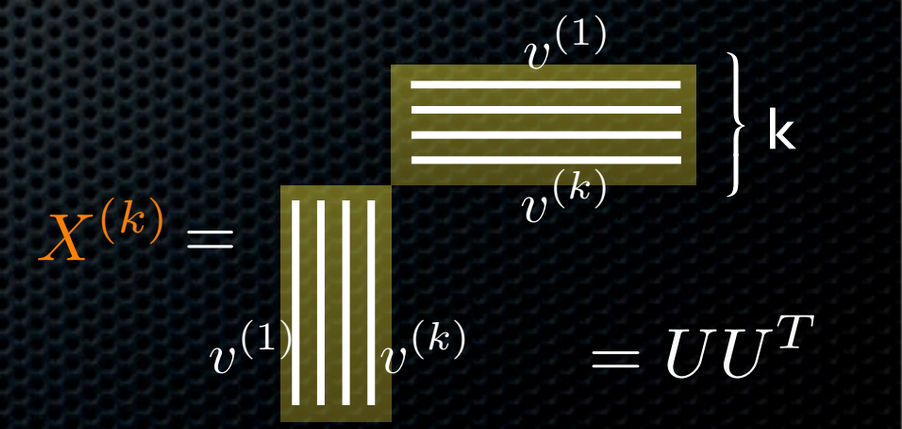
Sparsity = k

$$X^{(k+1)} := (1-\lambda)\, X^{(k)} + \lambda\, vv^T$$

$$v := \arg\max_{\|v\|=1} \; v^T \left(-\nabla f(X^{(k)})\right) v$$

‚largest‘ Eigenvector

$$\lambda = 1/k$$



Rank = k

The Algorithm

$$x^{(k+1)} := (1-\lambda)\, x^{(k)} + \lambda\, e_i$$

$$i := \arg\max_i \; -\nabla f(x^{(k)})_i$$

$$X^{(k+1)} := (1-\lambda)\, X^{(k)} + \lambda\, vv^T$$

$$v := \arg\max_{\|v\|=1} \; v^T\, (-\nabla f(X^{(k)}))\, v$$

The Convergence

After $O\left(\frac{1}{\epsilon}\right)$ steps the primal-dual error is $\leq \epsilon$.

[Clarkson SODA '08]

After $O\left(\frac{1}{\epsilon}\right)$ steps the primal-dual error is $\leq \epsilon$.

[Hazan LATIN '08]

Approximate Eigenvector computation

Instead of

$$v := \arg\max_{\|v\|=1} \; v^T M v$$

$$M := -\nabla f(X^{(k)})$$

it is enough to work with

$$v: \; v^T M v \geq \lambda_{\max} - \epsilon^2$$

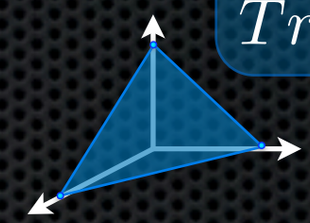
$$\|v\| = 1$$

Such a v can be found by doing $O\left(\frac{1}{\epsilon}\right)$ Lanzcos steps. Alternative: Power method

$\min f(X)$

How to solve general Semidefinite Programs?

$X \in \mathbb{S}^{n \times n}$
$X \succeq 0$
$Tr(X) = 1$



Optimization Version:

$$\min \; Tr(CX)$$
$$Tr(A_i X) \leq b_i \qquad i \in [m]$$
$$X \in \mathbb{S}^{n \times n}$$
$$X \succeq 0$$

Feasibility Version:

Find X s.t.

$$Tr(A_i X) \leq b_i \qquad i \in [m+1]$$
$$X \in \mathbb{S}^{n \times n}$$
$$X \succeq 0$$
$$Tr(X) = 1$$

$$f(X) := \frac{1}{M} \log \left(\sum_{i=1}^{m+1} e^{M(Tr(A_i X) - b_i)} \right)$$

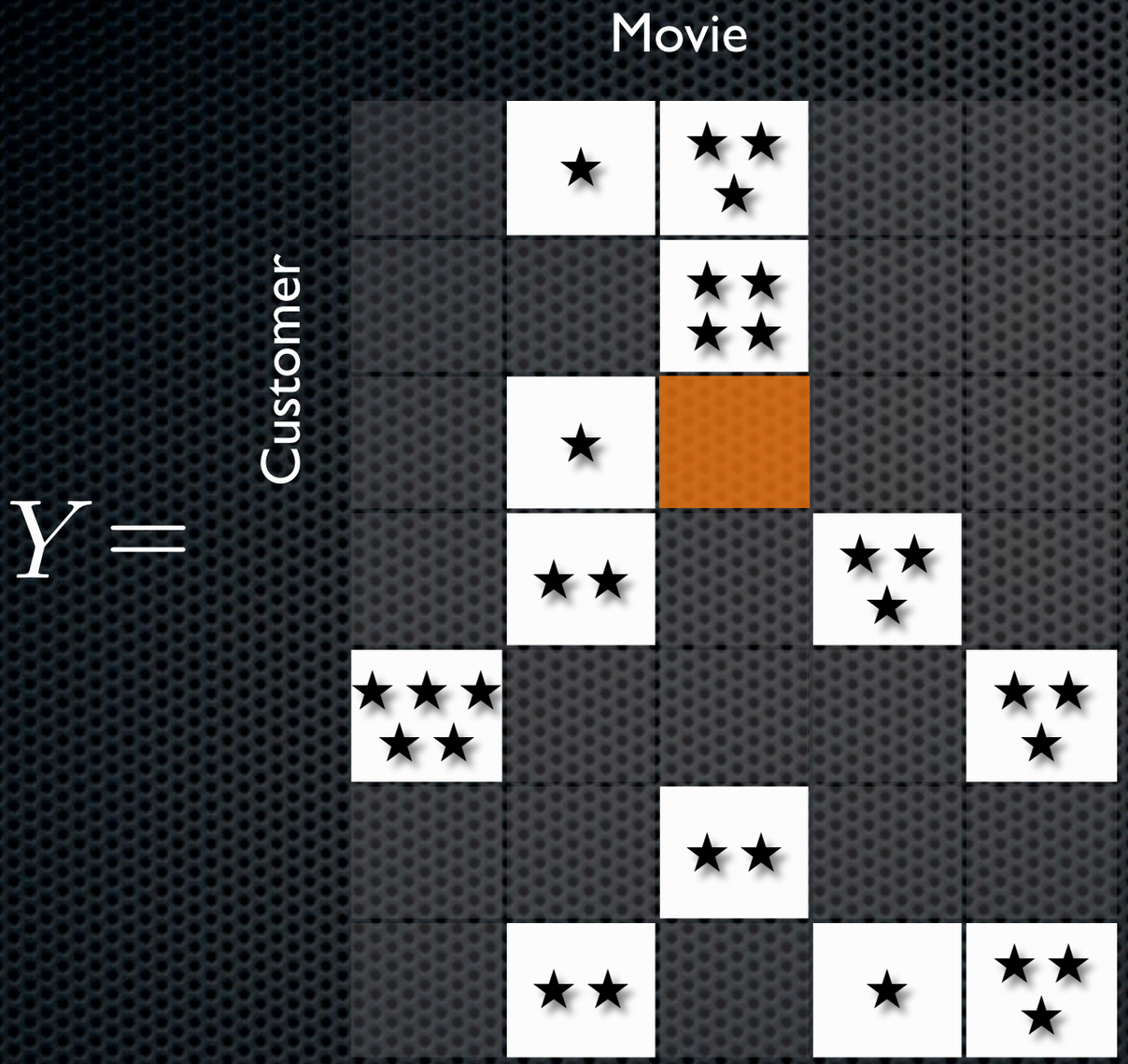
„Soft Max“

By this trick, Hazan‘s algorithm is able to satisfy all constraints up to an error $\leq \epsilon$.

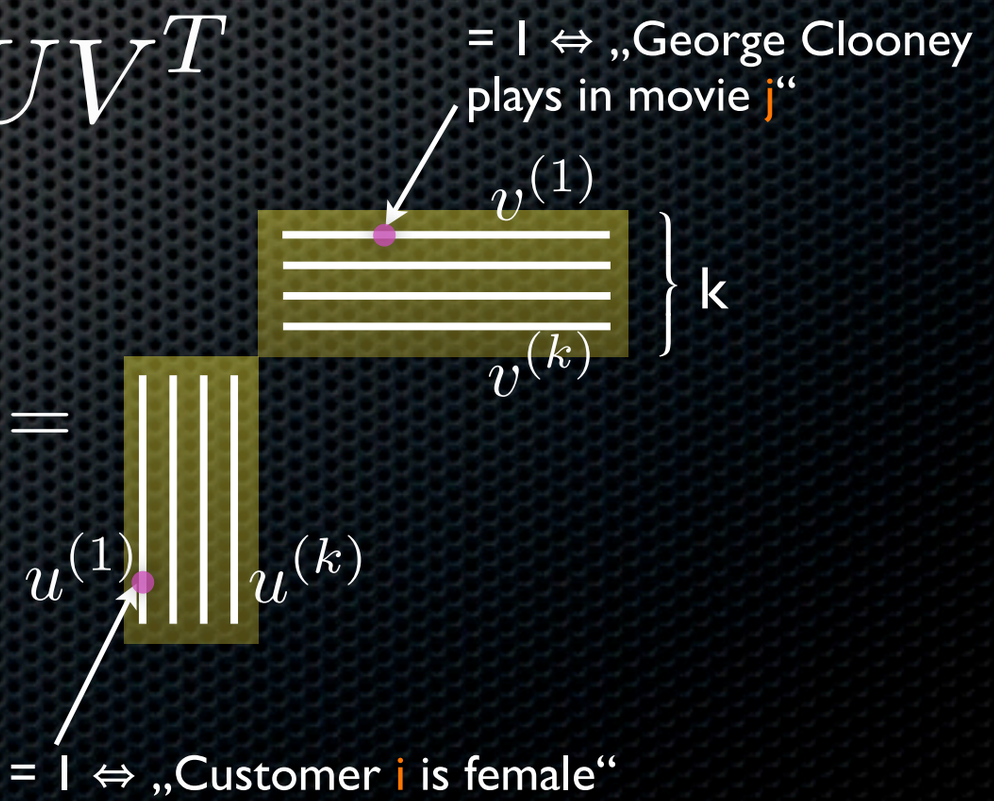
Matrix Factorizations

and machine learning

The Netflix Prize:
17‘000 Movies
500‘000 Customers
100‘000‘000 Ratings
(Observed Entries ≈ 1%)

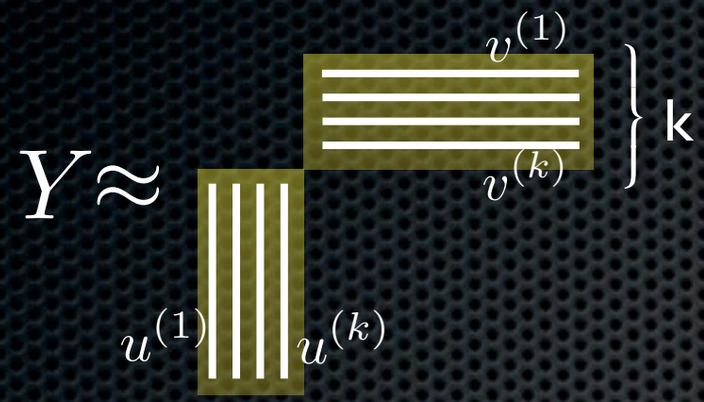


$\approx UV^T$



Matrix Factorizations

for recommender systems



[Short IEEE article,

Wikipedia: Netflix Prize]

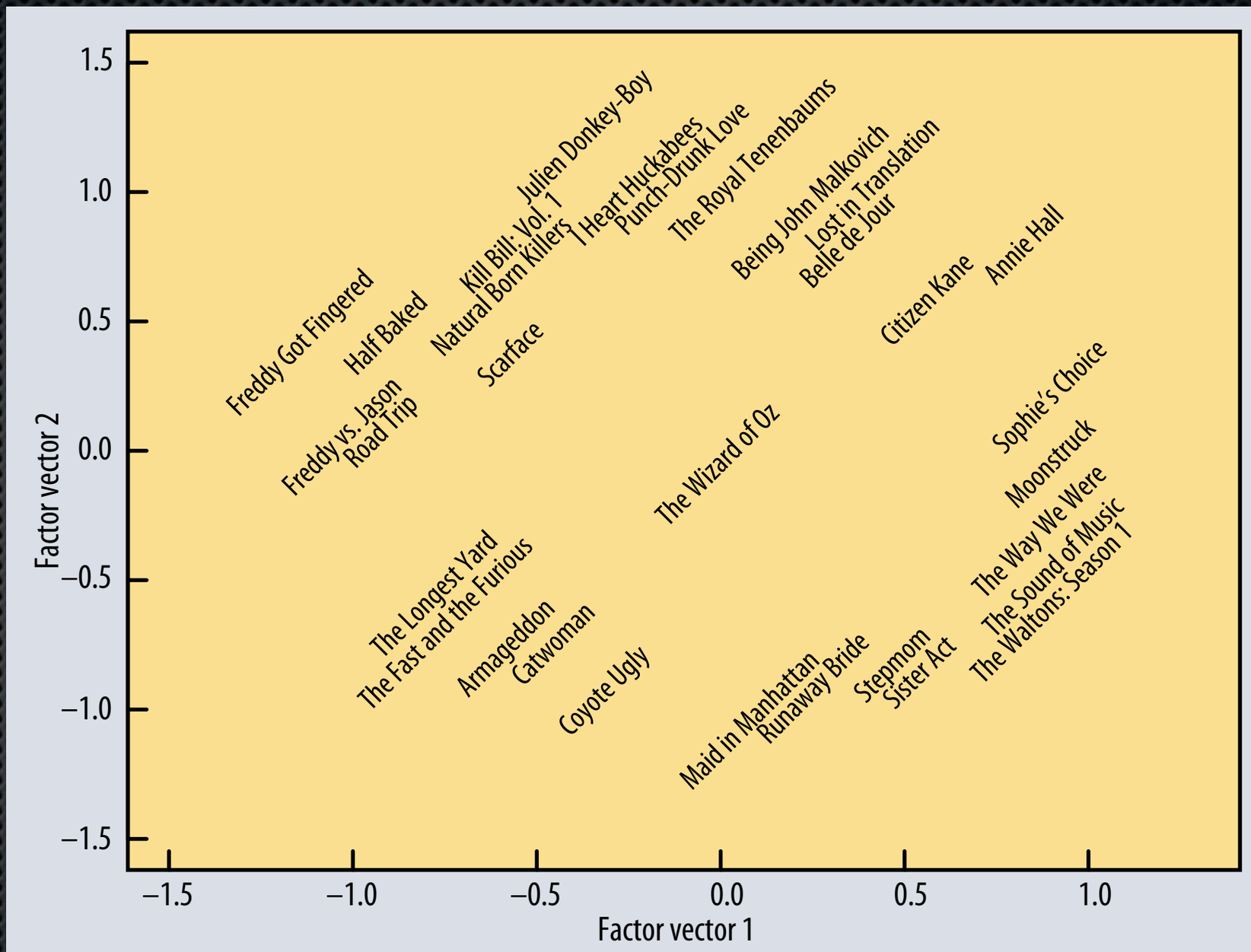


Figure 3. The first two vectors from a matrix decomposition of the Netflix Prize data. Selected movies are placed at the appropriate spot based on their factor vectors in two dimensions. The plot reveals distinct genres, including clusters of movies with strong female leads, fraternity humor, and quirky independent films.

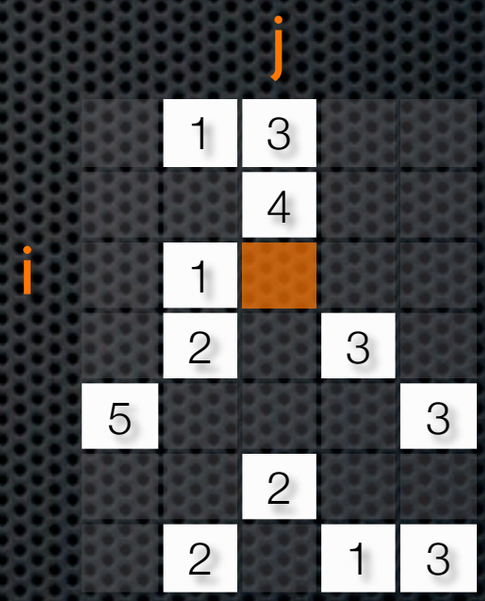
Matrix Factorizations

and machine learning

Applications:

- Customer i ↔ Product j

(Amazon, Netflix, Migros Cumulus etc...)



- Customer i → Customer j

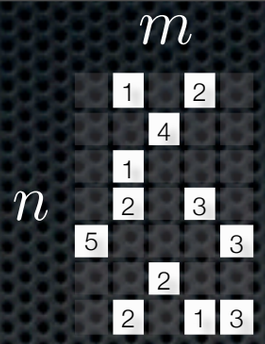
(Symmetry?, k=?)

- Word i ↔ Document j

(Search engines, Latent Semantic Analysis)

- many other applications
  (e.g. dimensionality reduction, clustering)

Accuracy vs Model complexity



$$Y \approx UV^T =: X$$

Error	Model complexity	
$\min_{U,V} f(UV^T)$	s.t. $rank(UV^T) = k$	Low Rank $= rank(X)$
	s.t. $\|U\|_{\text{Fro}}^2 + \|V\|_{\text{Fro}}^2 = t$	Low Norm $= 2\|X\|_*$

[Srebro NIPS '05]

$$f(X) := \sum_{ij \in S} (X - Y)_{ij}{}^2$$

Low Norm Matrix Factorization

$$f(X) := \sum_{ij \in S} (X - Y)_{ij}^2$$

$$\min_{U,V} \quad f(UV^T)$$

$$\text{s.t.} \quad \|U\|_{\text{Fro}}^2 + \|V\|_{\text{Fro}}^2 = t$$

$$= Tr(UU^T) + Tr(VV^T)$$

$$= Tr(Z)$$

$$\begin{pmatrix} UU^T & UV^T \\ VU^T & VV^T \end{pmatrix} =: Z$$

$$= \begin{pmatrix} U \\ V \end{pmatrix} (U^T \; V^T)$$

is equivalent to

$$\min_{Z} \; f(Z) \qquad Z \in \mathbb{S}^{(n+m)\times(n+m)}$$
$$Z \succeq 0$$
$$Tr(Z) = t$$

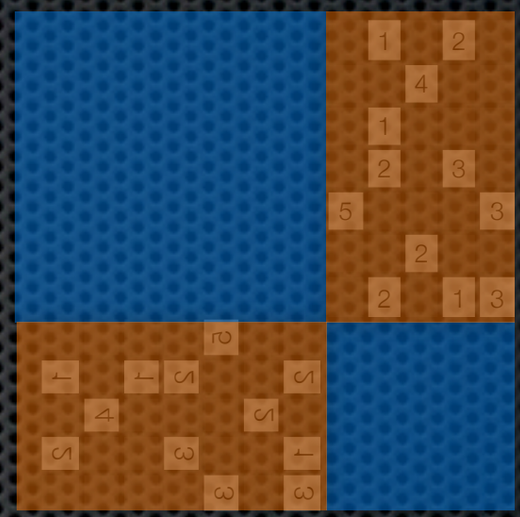
Perfectly fits for Hazan's Algorithm.

Low Norm Matrix Factorization

$$f(Z) := \sum_{ij \in S'} (Z - Y)_{ij}^2$$

Have to be careful, principal EV is not always the largest EV! Add a constant to the diagonal in that case.

We need the largest Eigenvector of $M := -\nabla f(Z^{(k)})$



$=: Z$

Largest Eigenvector of the bipartite weighted graph with adjacency matrix M.

- Hazan's new approximate SDP solver applies to *Low Norm Matrix Factorization*
- Easy to parallelize (Power method)
- Algorithm maintains sparsity structure of the given matrix, needs no additional memory
- Speed is comparable to existing methods, and much better than generic SDP solvers

Thanks