Dictionary Learning for Massive Matrix Factorization

Arthur Mensch, Julien Mairal
Gaël Varoquaux, Bertrand Thirion

Inria Parietal, Inria Thoth

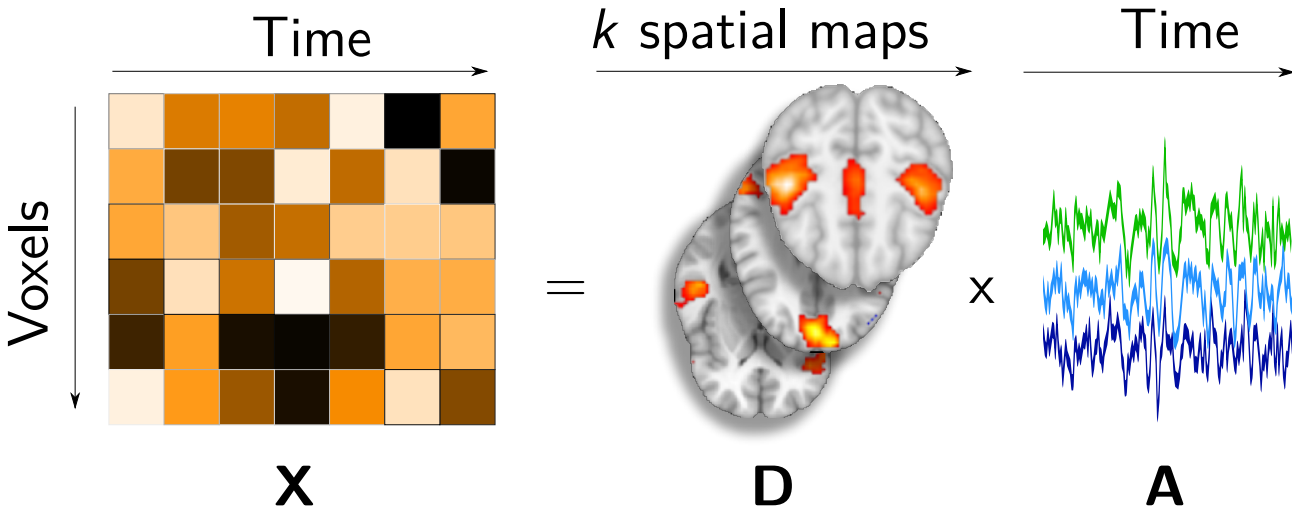
October 6, 2016

PARIETAL

Introduction

Why am I here ?

- **Inria Parietal**: machine learning for neuro-imaging (fMRI data)
- **Matrix factorization**: major ingredient in fMRI analysis
- Very large datasets (**2 TB**): we designed faster algorithms
- These algorithms can be used in **collaborative filtering**



- Work presented at ICML 2016

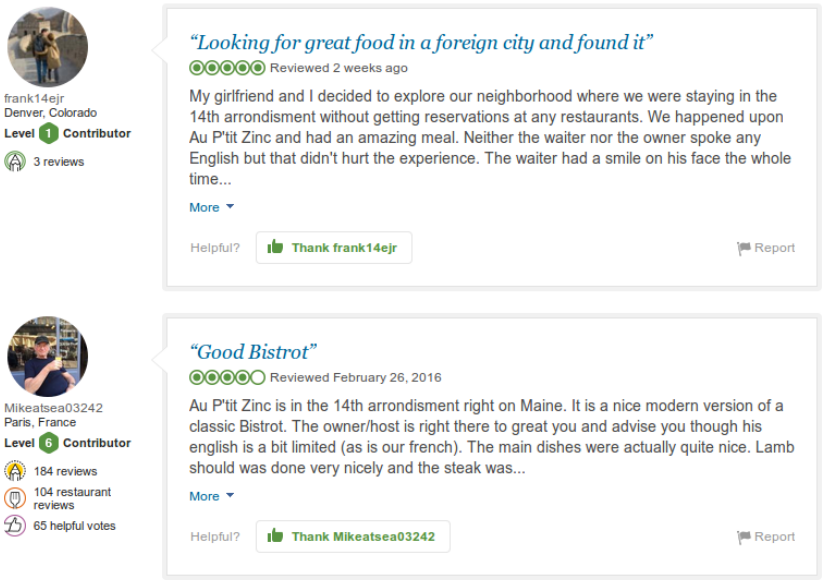
Déroulé

1. Matrix factorization for recommender systems
   - Collaborative filtering
   - Matrix factorization formulation
   - Existing methods
2. Subsampled online dictionary learning
   - Dictionary learning – existing methods
   - Handling missing values efficiently
   - New algorithm
3. Results
   - Setting
   - Benchmarks
   - Parameter setting

Déroulé

1. Matrix factorization for recommender systems
   - Collaborative filtering
   - Matrix factorization formulation
   - Existing methods
2. Subsampled online dictionary learning
   - Dictionary learning – existing methods
   - Handling missing values efficiently
   - New algorithm
3. Results
   - Setting
   - Benchmarks
   - Parameter setting

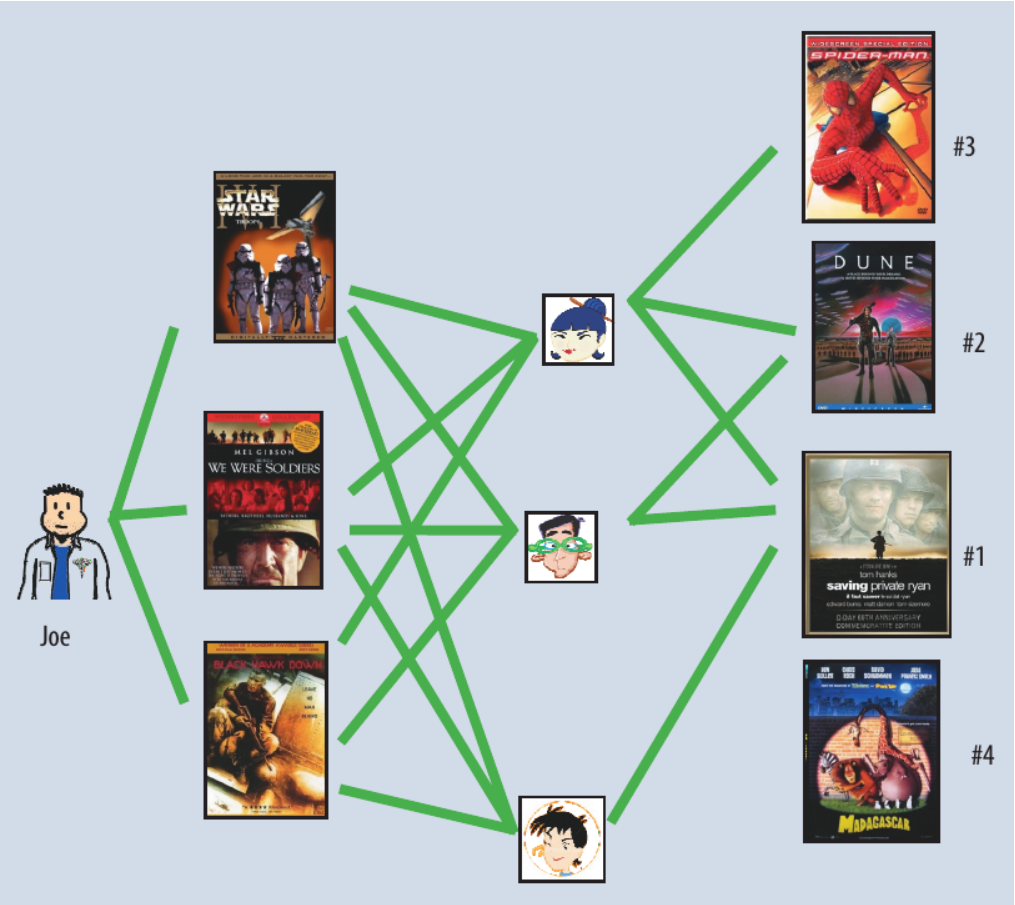
Collaborative filtering



- Collaborative platform
- n users rate a fraction of p items
- *e.g* movies, restaurants
- Estimate ratings for recommendation

Use the ratings of other users for recommendation

How to predict ratings ?



Credit: [Bell and Koren, 2007]

- Joe like *We were soldiers, Black Hawk down*.
- Bob and Alice like the same films, and also like *Saving private Ryan*.
- Joe should watch *Saving private Ryan*, because all of them indeed likes *war films*.

Need to uncover *topics* in items

Predicting rate with scalar products

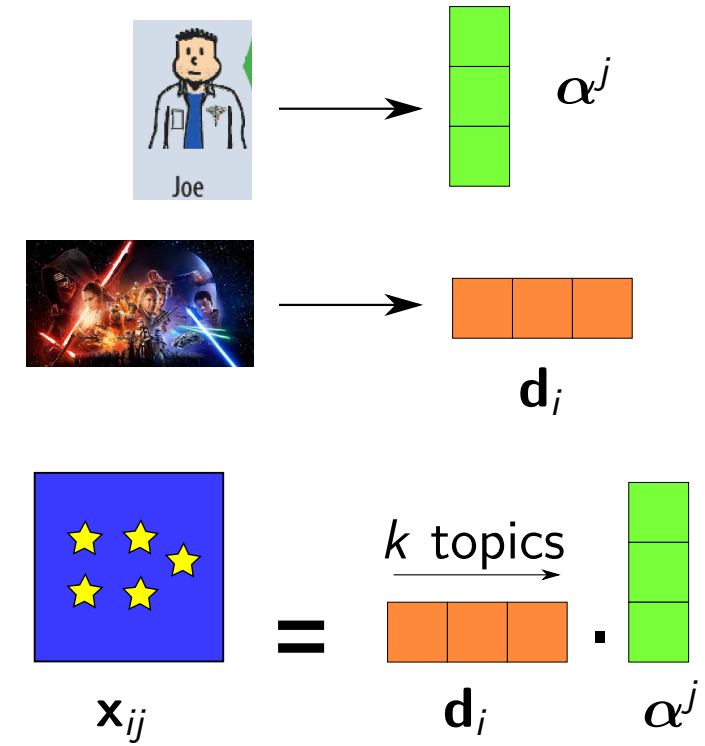
Embeddings to model the existence of *genre/category/topics*

- Representative vectors for users and items:

$$(\alpha^j)_{1\leq j\leq n}, (\mathbf{d}_i)_{1\leq i\leq p} \in \mathbb{R}^k$$

- q-th coefficient of $\mathbf{d}_i$, α^j

$=$ affinity with the "topic" q



Predicting rate with scalar products

Embeddings to model the existence of *genre/category/topics*

- Representative vectors for users and items:

$$(\alpha^j)_{1\leq j\leq n}, (\mathbf{d}_i)_{1\leq i\leq p} \in \mathbb{R}^k$$

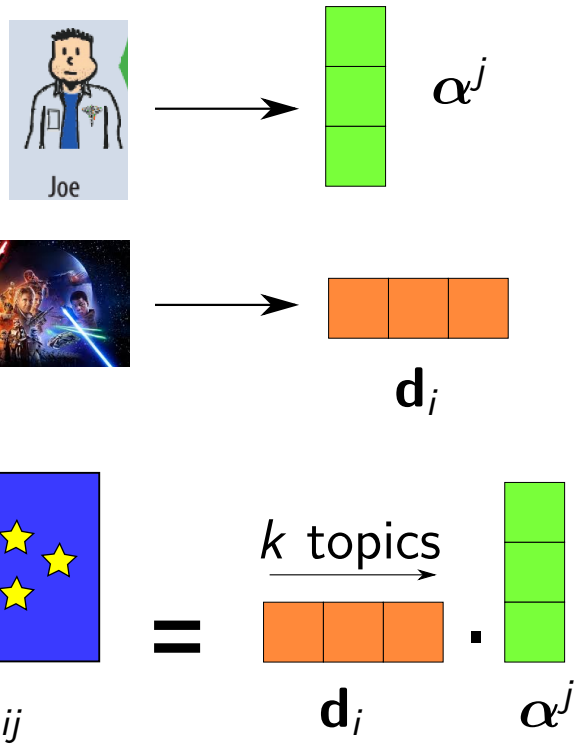
- q-th coefficient of $\mathbf{d}_i$, α^j

$=$ affinity with the "topic" q

- Ratings $\mathbf{x}_{ij}$ (item i, user j):

$$\mathbf{x}_{ij} = \mathbf{d}_i^\top \alpha^j \quad (+ \text{ biases})$$

$=$ *Common affinity for topics*



Predicting rate with scalar products

Embeddings to model the existence of *genre/category/topics*

- Representative vectors for users and items:

$$(\alpha^j)_{1\leq j\leq n}, (\mathbf{d}_i)_{1\leq i\leq p} \in \mathbb{R}^k$$

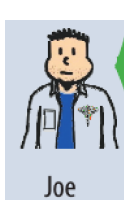
- q-th coefficient of $\mathbf{d}_i$, α^j

= affinity with the "topic" q

- Ratings $\mathbf{x}_{ij}$ (item i, user j):

$$\mathbf{x}_{ij} = \mathbf{d}_i^\top \alpha^j \quad (+ \text{ biases})$$

= *Common affinity for topics*



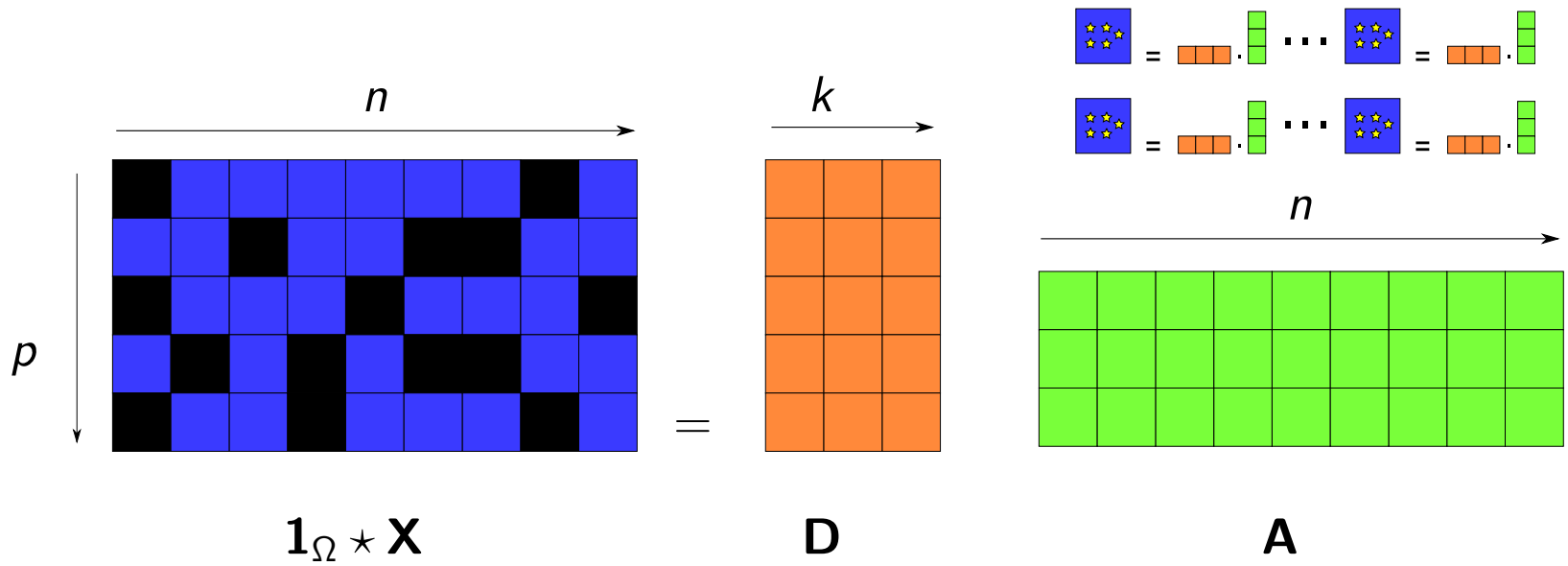
α^j

$\mathbf{d}_i$

$\mathbf{x}_{ij}$ = k topics $\mathbf{d}_i$ · α^j

Learning problem: estimate **D** and **A** with known ratings

Matrix factorization



- $\mathbf{X} \in \mathbb{R}^{p \times n} \approx \mathbf{DA} \in \mathbb{R}^{p \times k} \times \mathbb{R}^{k \times n}$
- Constraints / penalty on factors $\mathbf{D}$ and $\mathbf{A}$
- We only observe $\mathbf{1}_\Omega \star \mathbf{X}$ — Ω set of ratings provided by users

Recommender systems : **millions** of users, **millions** of items

How to scale matrix factorization to very large datasets ?

Formalism

Finding representation in $\mathbb{R}^k$ for items and users:

$$\min_{\substack{\mathbf{D}\in\mathbb{R}^{p\times k}\\ \mathbf{A}\in\mathbb{R}^{k\times n}}} \sum_{(i,j)\in\Omega} (\mathbf{x}_{ij} - \mathbf{d}_i^\top \boldsymbol{\alpha}^j)^2 + \lambda(\|\mathbf{D}\|_F^2 + \|\mathbf{A}\|_F^2)$$

$$= \|\mathbf{1}_\Omega \star (\mathbf{X} - \mathbf{DA})\|_2^2 + \lambda(\|\mathbf{D}\|_F^2 + \|\mathbf{A}\|_F^2) \quad \mathbf{1}_\Omega \text{ set of known ratings}$$

ℓ_2 reconstruction loss — ℓ_2 penalty for generalization

Existing methods

- Alternated minimization
- Stochastic gradient descent

Existing methods

Alternated minimization

- Minimize over $\mathbf{A}$, $\mathbf{D}$: alternate between

$$\mathbf{D} = \min_{\mathbf{D} \in \mathbb{R}^{p \times k}} \sum_{(i,j) \in \Omega} (\mathbf{x}_{ij} - \mathbf{d}_i^\top \boldsymbol{\alpha}^j)^2 + \lambda \|\mathbf{D}\|_F^2$$

$$\mathbf{A} = \min_{\mathbf{A} \in \mathbb{R}^{k \times n}} \sum_{(i,j) \in \Omega} (\mathbf{x}_{ij} - \mathbf{d}_i^\top \boldsymbol{\alpha}^j)^2 + \lambda \|\mathbf{A}\|_F^2$$

- No hyperparameters
- Slow and memory expensive: use all ratings at each iteration

a.k.a. coordinate descent (variation in parameter update order)

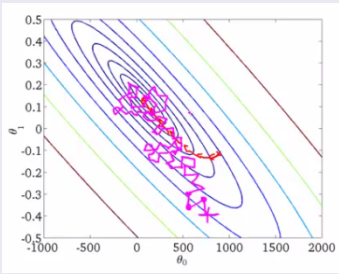
Existing methods

Stochastic gradient descent

$$\min_{\mathbf{A},\mathbf{D}} \sum_{(i,j)\in\Omega} f_{ij}(\mathbf{A},\mathbf{B}) \stackrel{\text{def}}{=} (\mathbf{x}_{ij} - \mathbf{d}_i^\top \boldsymbol{\alpha}^j)^2 + \frac{1}{c_j}\lambda\|\boldsymbol{\alpha}^j\|_2^2 + \frac{1}{c_i}\lambda\|\mathbf{d}_i\|_2^2$$

Gradient step for each rating:

$$(\mathbf{A}_t, \mathbf{D}_t) \leftarrow (\mathbf{A}_{t-1}, \mathbf{D}_{t-1}) - \frac{1}{c_t}\nabla_{(\mathbf{A},\mathbf{D})} f_{ij}(\mathbf{A}_{t-1}, \mathbf{D}_{t-1})$$



- Fast and memory efficient – won the Netflix prize
- Very sensitive to step sizes (c_t) – need to cross-validate

Towards a new algorithm

Best of both worlds ?

- Fast and memory efficient algorithm
- Little sensitive to hyperparameter setting

Subsampled online dictionary learning

- Builds upon the online *dictionary learning* algorithm
  - popular in computer vision and interpretable learning (fMRI)
- Adapt it to handle missing values efficiently

Déroulé

1. Matrix factorization for recommender systems
   - Collaborative filtering
   - Matrix factorization formulation
   - Existing methods
2. Subsampled online dictionary learning
   - Dictionary learning – existing methods
   - Handling missing values efficiently
   - New algorithm
3. Results
   - Setting
   - Benchmarks
   - Parameter setting

Dictionary learning

Recall: recommender system formalism

- Non-masked matrix factorization with ℓ_2 penalty:

$$\min_{\substack{\mathbf{D}\in\mathbb{R}^{p\times k}\\ \mathbf{A}\in\mathbb{R}^{k\times n}}} \sum_{j=1}^{n} (\mathbf{x}^j - \mathbf{D}^\top \boldsymbol{\alpha}^j)^2 + \lambda(\|\mathbf{D}\|_F^2 + \|\mathbf{A}\|_F^2)$$

- Penalties can be richer, and made into constraints

Dictionary learning

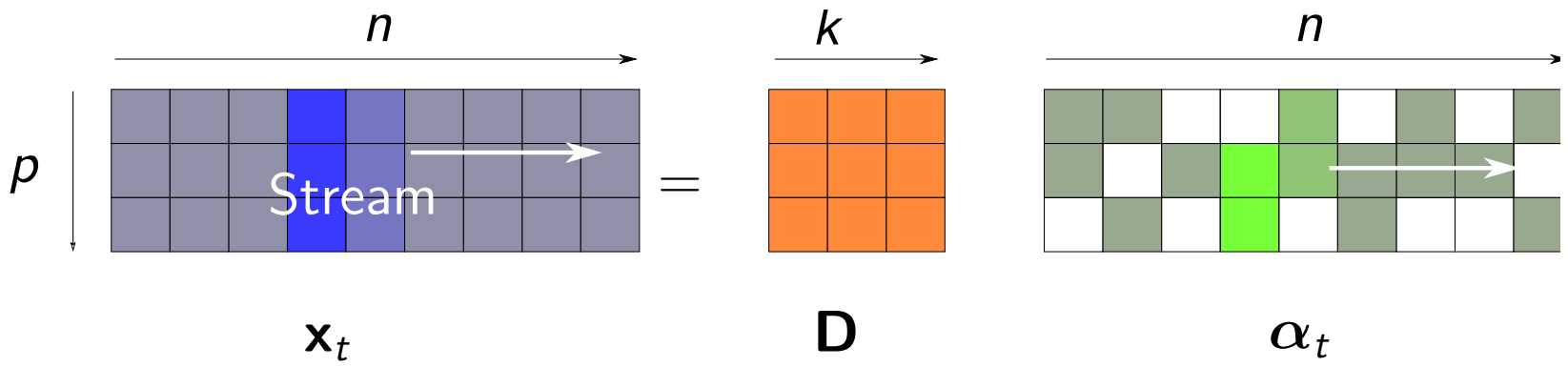
- Learn the left side factor [Olshausen and Field, 1997]

$$\min_{\mathbf{D}\in\mathcal{C}} \sum_{j=1}^{n} \|\mathbf{x}^j - \mathbf{D}\boldsymbol{\alpha}^j\|_2^2 + \lambda\Omega(\boldsymbol{\alpha}^j) \quad \boldsymbol{\alpha}^j = \operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^k} \|\mathbf{x}^i - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda\Omega(\boldsymbol{\alpha})$$

- Naive approach: alternated minimization

Online dictionary learning [Mairal et al., 2010]

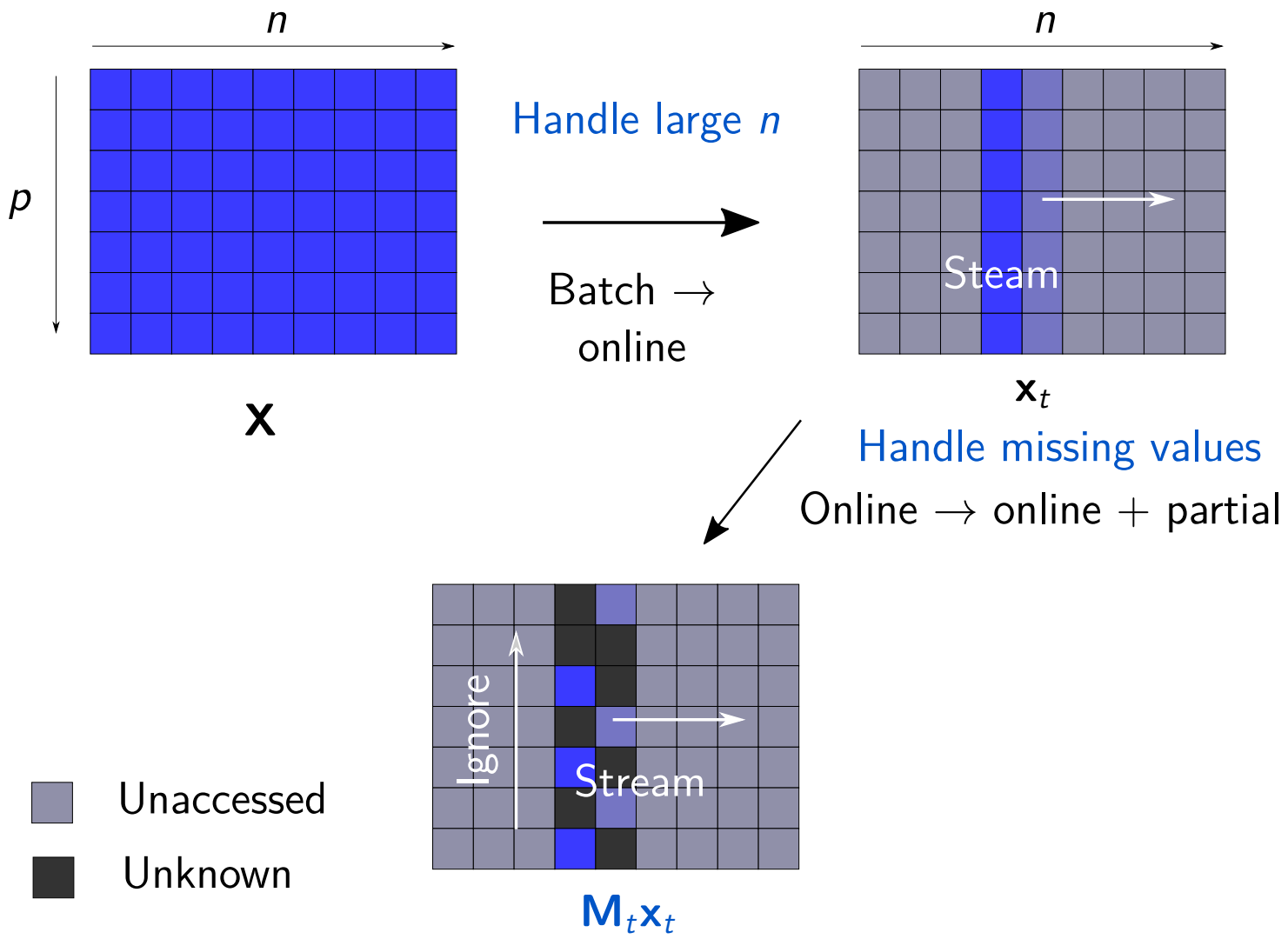
- At iteration t, select $\mathbf{x}_t$ in $\{\mathbf{x}^j\}_j$ (user ratings), improve $\mathbf{D}$
- Single iteration complexity $\propto$ sample dimension $\mathcal{O}(p)$
- $(\mathbf{D}_t)_t$ converges in a few epochs (one for large n)



- Very efficient in computer vision / networks / fMRI / hyperspectral images

Can we use it efficiently for recommender systems ?

In short: Handling missing values



Leverage streaming + partial access to samples

In detail: online dictionary learning

- Objective function involves *latent* codes (right side factor)

$$\min_{D \in \mathcal{C}} \frac{1}{t} \sum_{i=1}^{t} \|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i^*(\mathbf{D})\|_2^2, \quad \boldsymbol{\alpha}_i^*(\mathbf{D}) = \operatorname*{argmin}_{\boldsymbol{\alpha}} \frac{1}{2}\|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda\Omega(\boldsymbol{\alpha})$$

- Replace latent codes by codes computed with *old* dictionaries
- Build an upper-bounding surrogate function

$$\min \frac{1}{t} \sum_{i=1}^{t} \|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 \qquad \boldsymbol{\alpha}_i = \operatorname*{argmin}_{\boldsymbol{\alpha}} \frac{1}{2}\|\mathbf{x}_i - \mathbf{D}_{i-1}\boldsymbol{\alpha}\|_2^2 + \lambda\Omega(\boldsymbol{\alpha})$$

- Minimize surrogate — updateable online at low cost

In detail: online dictionary learning

Algorithm outline

1. **Compute code**

$$\boldsymbol{\alpha}_t = \underset{\boldsymbol{\alpha} \in \mathbb{R}^k}{\operatorname{argmin}} \|\mathbf{x}_t - \mathbf{D}_{t-1}\boldsymbol{\alpha}\|_2^2 + \lambda\Omega(\boldsymbol{\alpha}_t)$$

2. **Update the surrogate function**

$$g_t = \frac{1}{t}\sum_{i=1}^{t} \|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 = \operatorname{Tr}\left(\frac{1}{2}\mathbf{D}^\top\mathbf{D}\mathbf{A}_t - \mathbf{D}^\top\mathbf{B}_t\right)$$

$$\mathbf{A}_t = (1-\frac{1}{t})\mathbf{A}_{t-1} + \frac{1}{t}\boldsymbol{\alpha}_t\boldsymbol{\alpha}_t^\top \qquad \mathbf{B}_t = (1-\frac{1}{t})\mathbf{B}_{t-1} + \frac{1}{t}\mathbf{x}_t\boldsymbol{\alpha}_t^\top$$

3. **Minimize surrogate**

$$\mathbf{D}_t = \underset{\mathbf{D}\in\mathcal{C}}{\operatorname{argmin}}\, g_t(\mathbf{D}) \qquad \nabla g_t = \mathbf{D}\mathbf{A}_t - \mathbf{B}_t$$

In detail: online dictionary learning

Algorithm outline

1. **Compute code** – $\mathbf{x}_t \rightarrow$ complexity depends on p

$$\boldsymbol{\alpha}_t = \underset{\boldsymbol{\alpha} \in \mathbb{R}^k}{\operatorname{argmin}} \|\mathbf{x}_t - \mathbf{D}_{t-1}\boldsymbol{\alpha}\|_2^2 + \lambda\Omega(\boldsymbol{\alpha}_t)$$

2. **Update the surrogate function** – Complexity in $\mathcal{O}(p)$

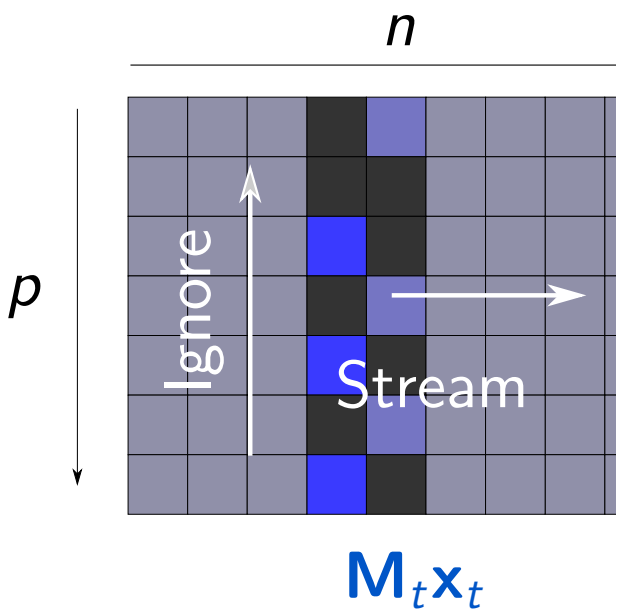
$$g_t = \frac{1}{t}\sum_{i=1}^{t} \|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 = \operatorname{Tr}\left(\frac{1}{2}\mathbf{D}^\top\mathbf{D}\mathbf{A}_t - \mathbf{D}^\top\mathbf{B}_t\right)$$

$$\mathbf{A}_t = (1-\frac{1}{t})\mathbf{A}_{t-1} + \frac{1}{t}\boldsymbol{\alpha}_t\boldsymbol{\alpha}_t^\top \qquad \mathbf{B}_t = (1-\frac{1}{t})\mathbf{B}_{t-1} + \frac{1}{t}\mathbf{x}_t\boldsymbol{\alpha}_t^\top$$

3. **Minimize surrogate** – Complexity in $\mathcal{O}(p)$

$$\mathbf{D}_t = \underset{\mathbf{D} \in \mathcal{C}}{\operatorname{argmin}}\, g_t(\mathbf{D}) \qquad \nabla g_t = \mathbf{D}\mathbf{A}_t - \mathbf{B}_t$$

Specification for a new algorithm



- **Constrained** : use only known ratings from Ω
- **Efficient**: single iteration in $\mathcal{O}(s)$, # of ratings provided by user t
- **Principled**: follows the online matrix factorization algorithm as much as possible

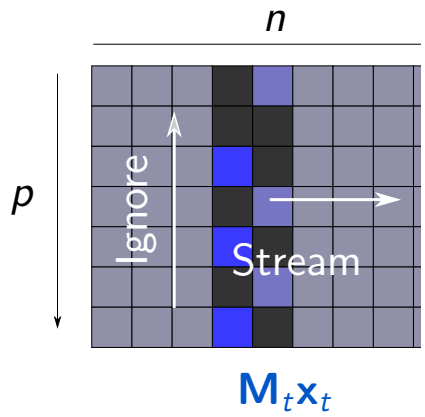
Missing values in practice

- **Data stream**: $(\mathbf{x}_t)_t \rightarrow$ masked $(\mathbf{M}_t\mathbf{x}_t)_t$

$=$ ratings from user t

- **Dimension**: p (all items) $\rightarrow s$ (rated items)
  - Use only $\mathbf{M}_t\mathbf{x}_t$ in algorithm computation

$\rightarrow$ **complexity in** $\mathcal{O}(s)$



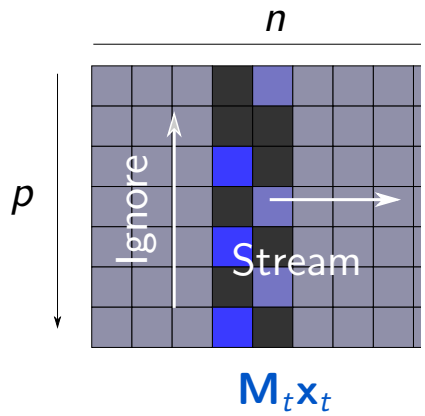
Missing values in practice

- **Data stream**: $(\mathbf{x}_t)_t \rightarrow$ masked $(\mathbf{M}_t\mathbf{x}_t)_t$

$=$ ratings from user t

- **Dimension**: p (all items) $\rightarrow s$ (rated items)
  - Use only $\mathbf{M}_t\mathbf{x}_t$ in algorithm computation

$\rightarrow$ **complexity in $\mathcal{O}(s)$**



Adaptation to make

- Modify all parts of the algorithm to obtain $\mathcal{O}(s)$ complexity

1. **Code computation**
2. **Surrogate update**
3. **Surrogate minimization**

Subsampled online dictionary learning

Check out paper !

Original online MF

1. Code computation

$$\boldsymbol{\alpha}_t = \underset{\boldsymbol{\alpha}\in\mathbb{R}^k}{\operatorname{argmin}} \|\mathbf{x}_t - \mathbf{D}_{t-1}\boldsymbol{\alpha}\|_2^2 + \lambda\Omega(\boldsymbol{\alpha}_t)$$

2. Surrogate aggregation

$$\mathbf{A}_t = \frac{1}{t}\sum_{i=1}^{t}\boldsymbol{\alpha}_i\boldsymbol{\alpha}_i^\top$$

$$\mathbf{B}_t = \mathbf{B}_{t-1} + \frac{1}{t}(\mathbf{x}_t\boldsymbol{\alpha}_t^\top - \mathbf{B}_{t-1})$$

3. Surrogate minimization

$$\mathbf{D}^j \leftarrow p^{\perp}_{\mathcal{C}_j^r}\left(\mathbf{D}^j - \frac{1}{(\mathbf{A}_t)_{j,j}}(\mathbf{D}\mathbf{A}_t^j - \mathbf{B}_t^j)\right)$$

Our algorithm

1. Code computation: masked loss

$$\boldsymbol{\alpha}_t = \underset{\boldsymbol{\alpha}\in\mathbb{R}^k}{\operatorname{argmin}} \|\mathbf{M}_t(\mathbf{x}_t - \mathbf{D}_{t-1}\boldsymbol{\alpha})\|_2^2 + \lambda\frac{\operatorname{rk}\mathbf{M}_t}{p}\Omega(\boldsymbol{\alpha}_t)$$

2. Surrogate aggregation

$$\mathbf{A}_t = \frac{1}{t}\sum_{i=1}^{t}\boldsymbol{\alpha}_i\boldsymbol{\alpha}_i^\top$$

$$\mathbf{B}_t = \mathbf{B}_{t-1} + \frac{1}{\sum_{i=1}^{t}\mathbf{M}_i}(\mathbf{M}_t\mathbf{x}_t\boldsymbol{\alpha}_t^\top - \mathbf{M}_t\mathbf{B}_{t-1})$$

3. Surrogate minimization

$$\mathbf{M}_t\mathbf{D}^j \leftarrow p^{\perp}_{\mathcal{C}_j}\left(\mathbf{M}_t\mathbf{D}^j - \frac{1}{(\mathbf{A}_t)_{j,j}}\mathbf{M}_t(\mathbf{D}(\mathbf{A}_t^j - (\mathbf{B}_t^j)\right)$$

Déroulé

1. Matrix factorization for recommender systems
   - Collaborative filtering
   - Matrix factorization formulation
   - Existing methods
2. Subsampled online dictionary learning
   - Dictionary learning – existing methods
   - Handling missing values efficiently
   - New algorithm
3. Results
   - Setting
   - Benchmarks
   - Parameter setting

Experiments

Validation : Test RMSE (rating prediction) vs CPU time

Baseline : Coordinate descent solver [Yu et al., 2012] for

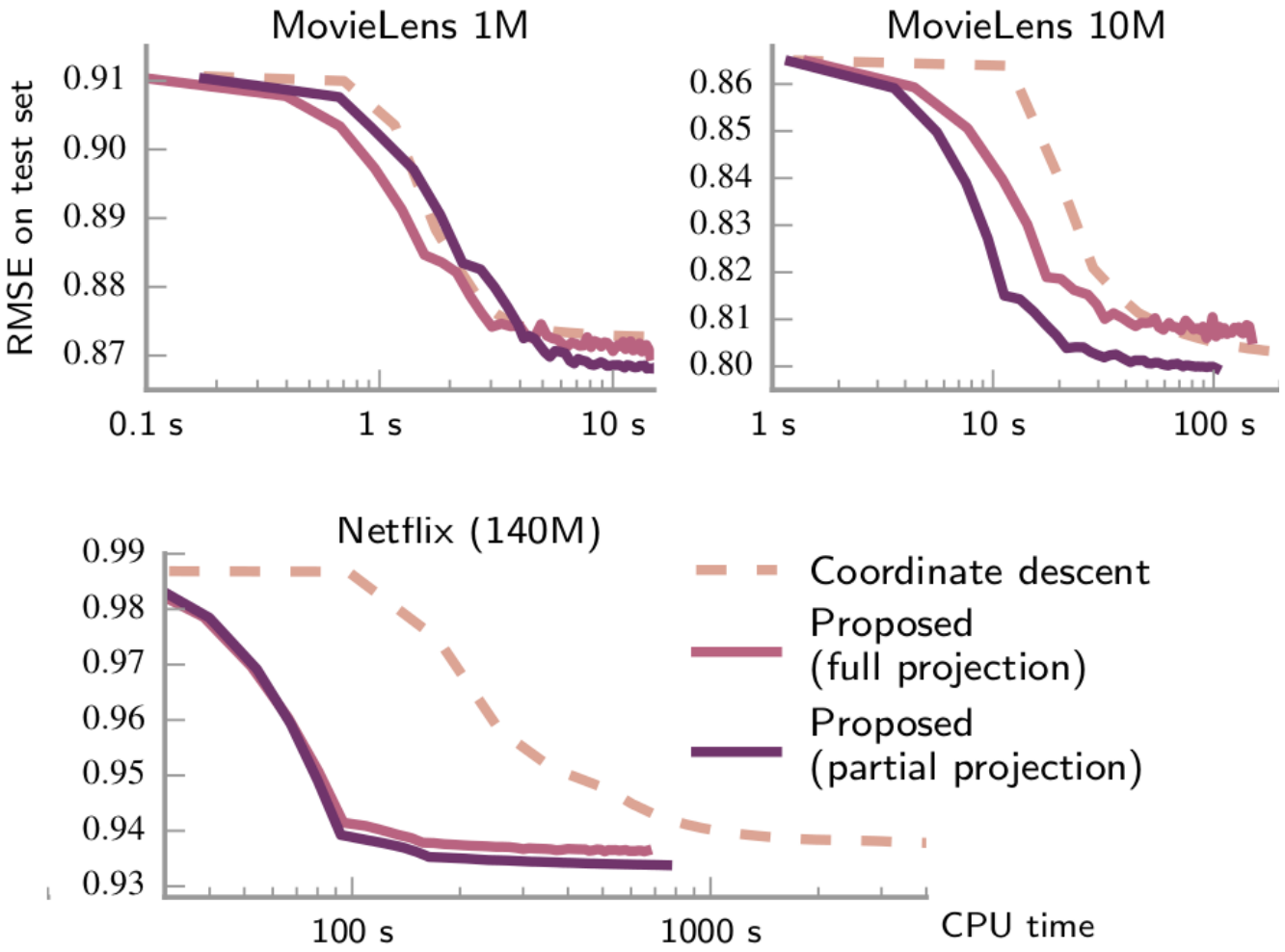
$$\min_{\substack{\mathbf{D}\in\mathbb{R}^{p\times k}\\ \mathbf{A}\in\mathbb{R}^{k\times n}}} \sum_{(i,j)\in\Omega} (\mathbf{x}_{ij} - \mathbf{d}_i^\top \boldsymbol{\alpha}^j)^2 + \lambda(\|\mathbf{D}\|_F^2 + \|\mathbf{A}\|_F^2)$$

- Fastest solver available apart from SGD — hyperparameters
- ↑ Our method has a learning rate with little influence

Datasets : Movielens, Netflix

- Publicly available
- Larger one in the industry...

Results



Scalable algorithm: speed-up improves with size

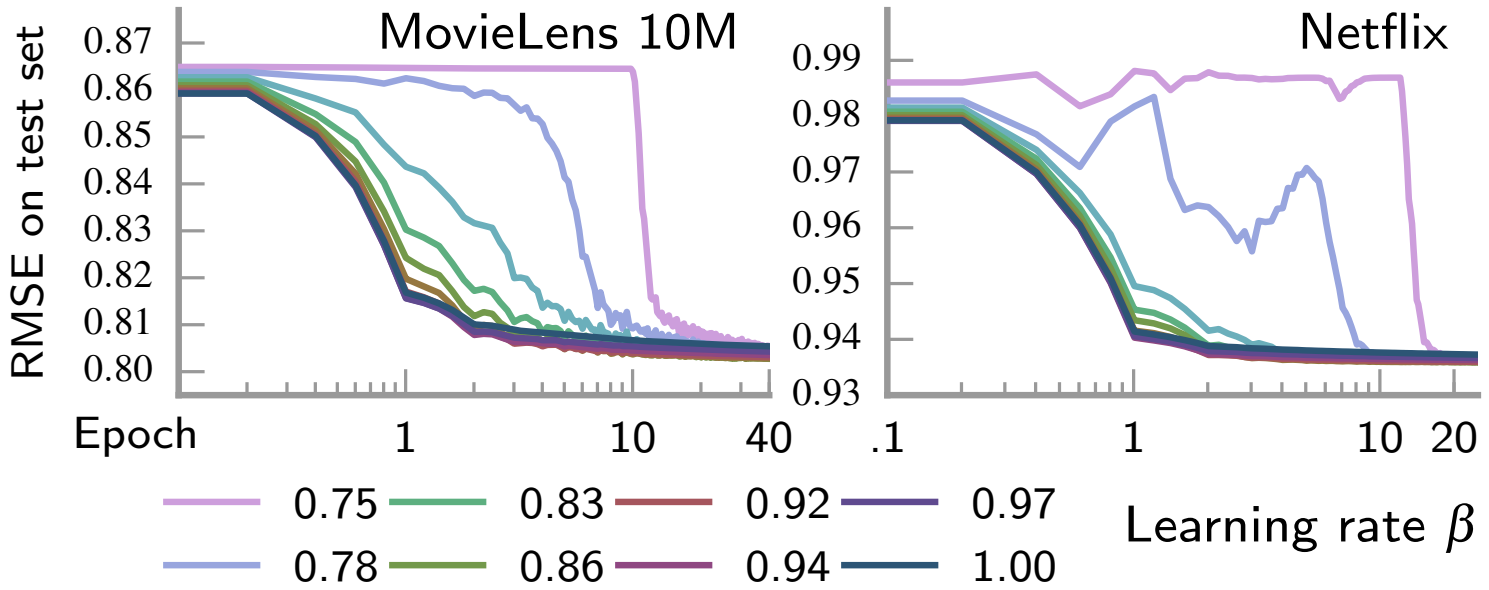
Performance

Dataset	Test RMSE		Convergence time		Speed
	CD	**SODL**	CD	**SODL**	-up
ML 1M	0.872	**0.866**	**6 s**	8 *s*	$\times$0.75
ML 10M	0.802	**0.799**	223 *s*	**60 s**	$\times$3.7
NF (140M)	0.938	**0.934**	1714 *s*	**256 s**	$\times$6.8

- Outperform coordinate descent beyond 10M ratings
- Same prediction performance
- Speed-up 6.8$\times$ on Netflix
- Simple model: RMSE is not state-of-the-art

Robustness to learning rate

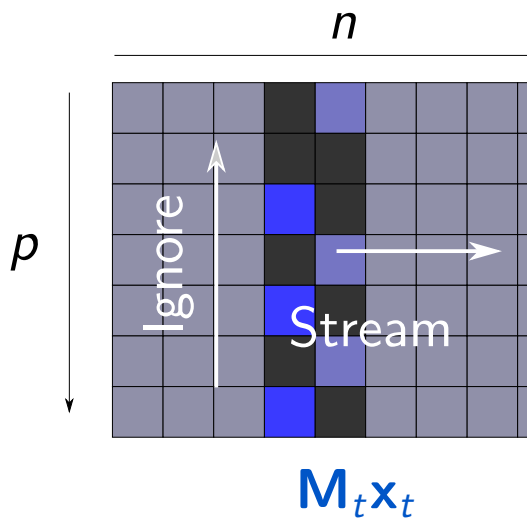
- Learning rate in algorithm to be set in $[0.75, 1]$ ($\leftarrow$ theory)
- **In practice**: Just set it in $[0.8, 1]$



Conclusion

Take-home message

Online matrix factorization can be adapted to handle missing value efficiently, with very good performance in reccommender system



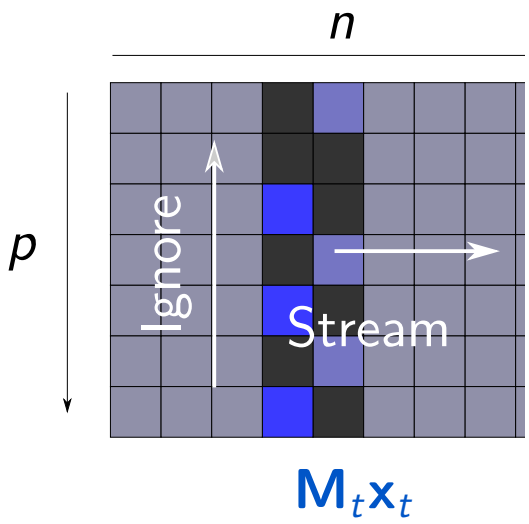
Algorithm usable in any rich model involving matrix factorization

- Python package `http://github.com/arthurmensch/modl`
- Article/slides at `http://amensch.fr/publications`

Conclusion

Take-home message

Online matrix factorization can be adapted to handle missing value efficiently, with very good performance in reccommender system



Algorithm usable in any rich model involving matrix factorization

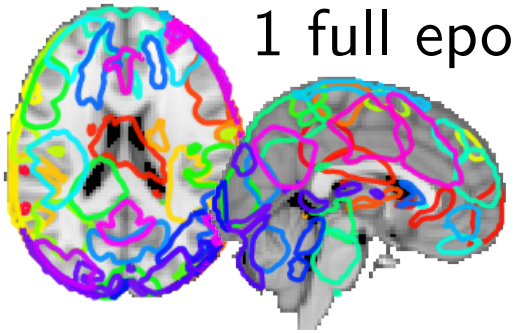
- Python package `http://github.com/arthurmensch/modl`
- Article/slides at `http://amensch.fr/publications`

Questions ?

Appendix: Resting-state fMRI

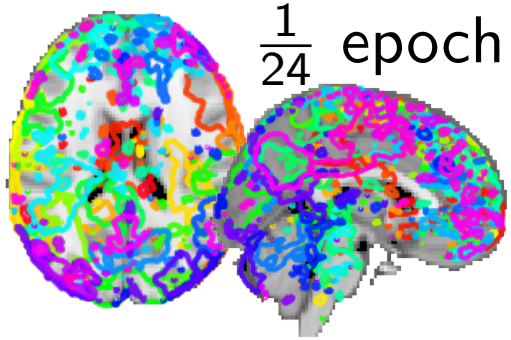
Online dictionary learning

1 full epoch



235 h run time

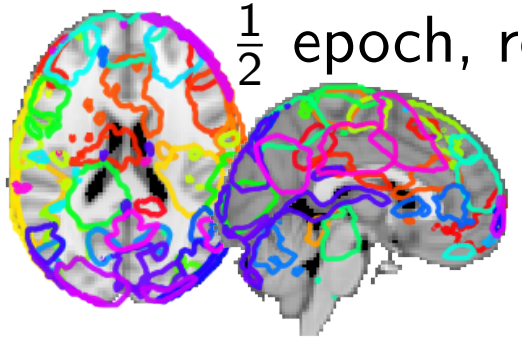
$\frac{1}{24}$ epoch



10 h run time

Proposed method

$\frac{1}{2}$ epoch, reduction r=12



10 h run time

Qualitatively, usable maps are obtained **10**× **faster**

Bibliography I

[Bell and Koren, 2007] Bell, R. M. and Koren, Y. (2007).

Lessons from the Netflix prize challenge.

ACM SIGKDD Explorations Newsletter, 9(2):75–79.

[Mairal et al., 2010] Mairal, J., Bach, F., Ponce, J., and Sapiro, G. (2010).

Online learning for matrix factorization and sparse coding.

The Journal of Machine Learning Research, 11:19–60.

[Olshausen and Field, 1997] Olshausen, B. A. and Field, D. J. (1997).

Sparse coding with an overcomplete basis set: A strategy employed by V1?

Vision Research, 37(23):3311–3325.

[Yu et al., 2012] Yu, H.-F., Hsieh, C.-J., and Dhillon, I. (2012).

Scalable coordinate descent approaches to parallel matrix factorization for recommender systems.

In *Proceedings of the International Conference on Data Mining*, pages 765–774. IEEE.