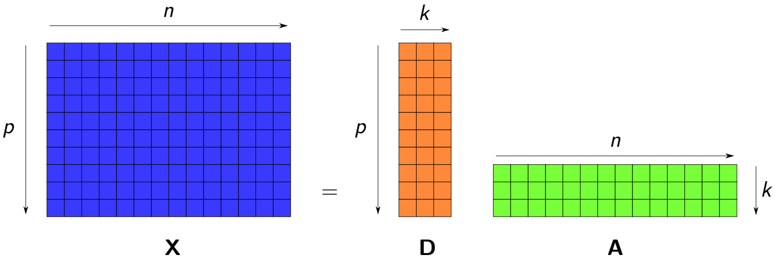
Dictionary Learning for Massive Matrix Factorization

Arthur Mensch, Julien Mairal,
Gaël Varoquaux, Bertrand Thirion

Inria/CEA Parietal, Inria Thoth

June 20, 2016

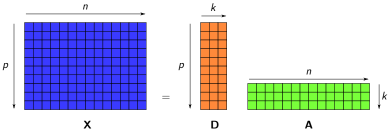
Matrix factorization

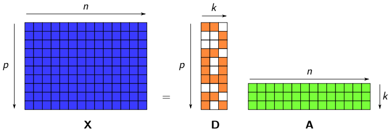


- $\mathbf{X} \in \mathbb{R}^{p \times n} = \mathbf{DA} \in \mathbb{R}^{p \times k} \times \mathbb{R}^{k \times n}$
- Flexible tool for unsupervised data analysis
- Dataset has lower underlying complexity than appearing size

How to scale it to very large datasets ? (Brain imaging, **2TB**)

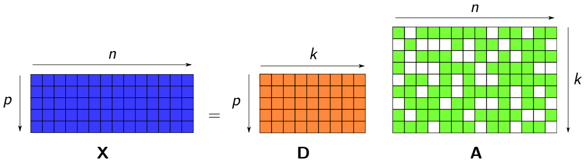
Matrix factorization





Low rank factorization : $k < p$ ...with optional sparse factors

$\rightarrow$ **interpretable** data (fMRI, genetics, topic modeling)



Overcomplete dictionary learning $k \gg p$ - sparse **A**
[Olshausen and Field, 1997]

Formalism and methods

Non-convex formulation

$$\min_{\mathbf{D}\in\mathcal{C},\mathbf{A}\in\mathbf{R}^{k\times n}} \|\mathbf{X}-\mathbf{DA}\|_2^2 + \lambda\Omega(\mathbf{A})$$

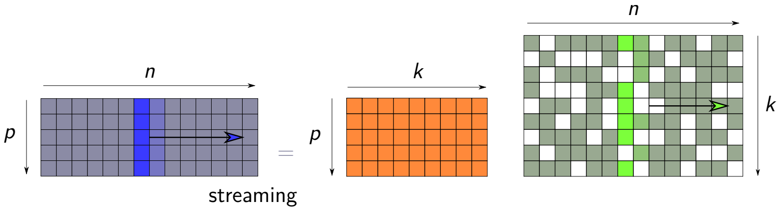
- Constraints on $\mathbf{D}$
- Penalty on $\mathbf{A}$ (ℓ_1, ℓ_2)

Naive resolution

- Alternated minimization: use full $\mathbf{X}$ at each iteration
- Very slow : single iteration in $\mathcal{O}(p\,n)$

Online matrix factorization

- Stream $(\mathbf{x}_t)$, update $\mathbf{D}$ at each t [Mairal et al., 2010]
- Single iteration in $\mathcal{O}(p)$, a few epochs



$\mathbf{x}_t$ $\mathbf{D}$ $\boldsymbol{\alpha}_t$

- Large n, regular p, *eg* image patches:

$$p = 256 \qquad n \approx 10^6 \qquad \mathbf{1GB}$$

- Both (sparse) low-rank factorization / sparse coding

Scaling-up for massive matrices

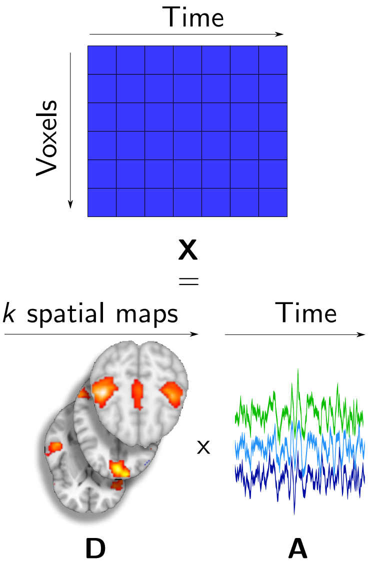
Functional MRI (HCP dataset)

- Brain "movies" : space $\times$ time
- Extract k *sparse* networks

$p = 2 \cdot 10^5 \quad n = 2 \cdot 10^6 \quad$ **2 TB**

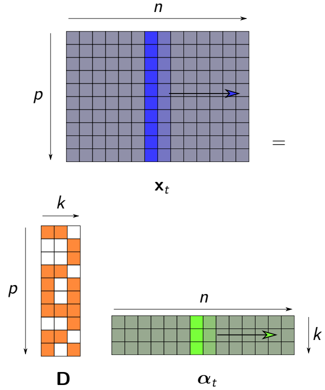
- Way larger than vision problems
- Unusual setting: data is large in *both* directions

Also useful in *collaborative filtering*



Scaling-up for massive matrices

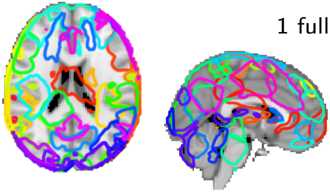
Out-of-the-box online algorithm ?



Limited time budget ?

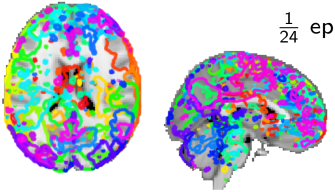
Need to accomodate large p

1 full epoch



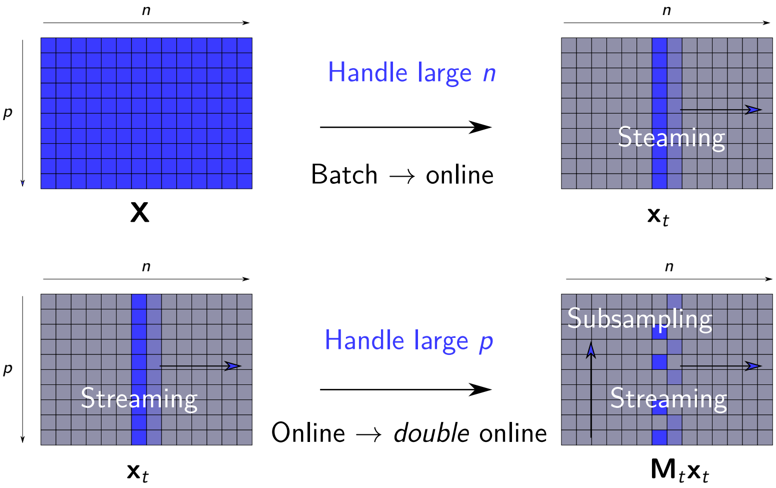
235 h run time

$\frac{1}{24}$ epoch



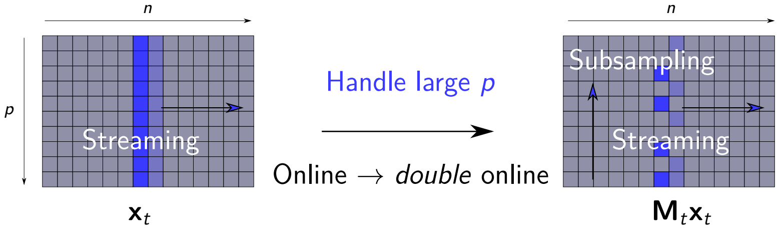
10 h run time

Scaling-up in both directions



Online learning + partial random access to samples

Scaling-up in both directions



- Low-distorsion lemma [Johnson and Lindenstrauss, 1984]
- Random linear alebra [Halko et al., 2009]
- *Sketching* for data reduction [Pilanci and Wainwright, 2014]

Algorithm design

Online dictionary learning [Mairal et al., 2010]

1. **Compute code** – $\mathcal{O}(p)$

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

2. **Update surrogate** – $\mathcal{O}(p)$

$$g_t = \frac{1}{t}\sum_{i=1}^{t} \|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2$$

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

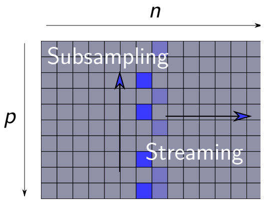
$$\mathbf{D}_t = \operatorname*{argmin}_{\mathbf{D} \in \mathcal{C}} g_t(\mathbf{D}) = \operatorname*{argmin}_{\mathbf{D} \in \mathcal{C}} \operatorname{Tr}\left(\mathbf{D}^\top \mathbf{D}\mathbf{A}_t - \mathbf{D}^\top \mathbf{B}_t\right)$$

$\mathbf{x}_t$ access $\rightarrow$ $\mathcal{O}(p)$ algorithm (complexity dependency in p)

Introducing subsampling

Iteration cost in $\mathcal{O}(p)$: can we reduce it?

- $\mathbf{x}_t \to \mathbf{M}_t \mathbf{x}_t$, $p \to \mathrm{rk}\,\mathbf{M}_t = s$
- Use only $\mathbf{M}_t \mathbf{x}_t$ in algorithm computation: **complexity in $\mathcal{O}(s)$**



$\mathbf{M}_t \mathbf{x}_t$

Our contribution

- Adapt the 3 parts of the algorith to obtain $\mathcal{O}(s)$ complexity

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

[Szabó et al., 2011]: dictionary learning with missing value – $\mathcal{O}(p)$

1. Code computation

Linear regression with random sampling

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

approximative solution of

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

validity in high dimension, with incoherent features:

$$\mathbf{D}^\top \mathbf{M}_t \mathbf{D} \approx \frac{s}{p} \mathbf{D}^\top \mathbf{D} \qquad \mathbf{D}^\top \mathbf{M}_t \mathbf{x}_t \approx \frac{s}{p} \mathbf{D}^\top \mathbf{x}_t$$

2. Surrogate update

Original algorithm: $\mathbf{A}_t$ and $\mathbf{B}_t$ used in dictionary update

- $\mathbf{A}_t = \frac{1}{t}\sum_{i=1}^{t} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_i^\top$ same as in online algorithm
- $\mathbf{B}_t = (1 - \frac{1}{t})\mathbf{B}_{t-1} + \frac{1}{t}\mathbf{x}_t \boldsymbol{\alpha}_t^\top = \frac{1}{t}\sum_{i=1}^{t} \mathbf{x}_i \boldsymbol{\alpha}_i^\top$ Forbidden

Partial update of $\mathbf{B}_t$ at each iteration

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

- Only $\mathbf{M}_t \mathbf{B}$ is updated
- Behaves like $\mathbb{E}_{\mathbf{x}}[\mathbf{x}\boldsymbol{\alpha}]$ for large t

3. Surrogate minimization

Original algorithm : block coordinate descent with projection on $\mathcal{C}$

$$\min_{\mathbf{D}\in\mathcal{C}} g_t(\mathbf{D}) \qquad \mathbf{D}_j \leftarrow p^{\perp}_{\mathcal{C}_j}(\mathbf{D}_j - \frac{1}{\mathbf{A}_{j,j}}(\mathbf{D}(\mathbf{A}_t)_j - (\mathbf{B}_t)_j))$$

Forbidden update of full $\mathbf{D}$ at iteration t

Cautious update

Leave dictionary unchanged for unseen features $(\mathbf{I} - \mathbf{M}_t)$

$$\min_{\substack{\mathbf{D}\in\mathcal{C} \\ (\mathbf{I}-\mathbf{M}_t)\mathbf{D}=(\mathbf{I}-\mathbf{M}_t)\mathbf{D}_{t-1}}} g_t(\mathbf{D})$$

$\mathcal{O}(s)$ update in block coordinate descent

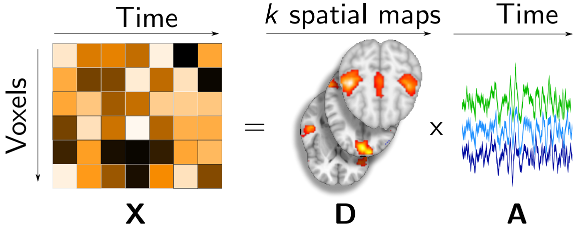
$$\mathbf{D}_j \leftarrow p^{\perp}_{\mathcal{C}^r_j}(\mathbf{D}_j - \frac{1}{(\mathbf{A}_t)_{j,j}}(\mathbf{M}_t(\mathbf{D}(\mathbf{A}_t)_j - (\mathbf{B}_t)_j)))$$

ℓ_1 ball $C^r_j = \{\mathbf{D} \in \mathcal{C}, \|\mathbf{M}_t\mathbf{D}\|_1 \leq \|\mathbf{M}_t\mathbf{D}_{t-1}\|_1\}$

Resting-state fMRI

HCP dataset

- One brain image per second
- 200 sessions $n = 2 \cdot 10^6$ $p = 2 \cdot 10^5$



Sparse decomposition $k = 70$ $\mathcal{C} = \mathcal{B}_1^k$ $\Omega = \| \cdot \|_2^2$

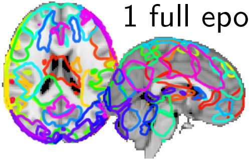
Validation

Increase reduction factor $\frac{p}{s}$

- Objective function on test set vs CPU time

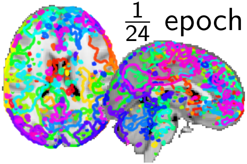
Resting-state fMRI

Online dictionary learning



1 full epoch

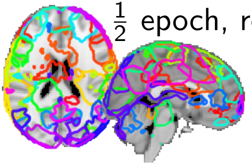
235 h run time



$\frac{1}{24}$ epoch

10 h run time

Proposed method

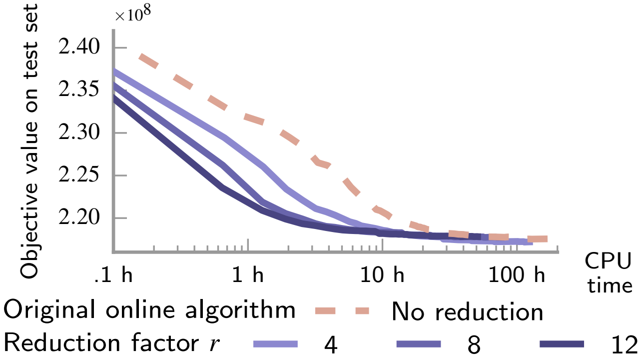


$\frac{1}{2}$ epoch, reduction $r = 12$

10 h run time

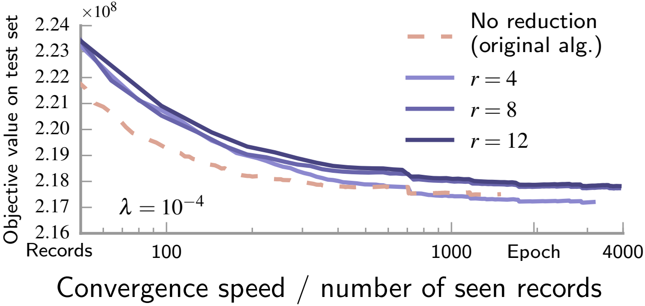
Qualitatively, usable maps are obtained **10× faster**

Resting-state fMRI



Speed-up close to reduction factor $\frac{p}{s}$

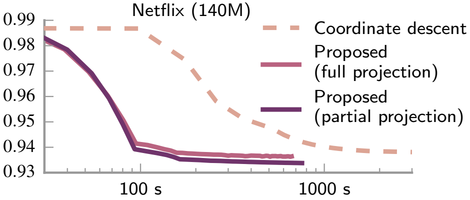
Resting-state fMRI



Information is acquired faster

Collaborative filtering

- $\mathbf{M}_t \mathbf{x}_t$ movie ratings from user t
- vs. coordinate descent for MMMF loss (no hyperparameters)



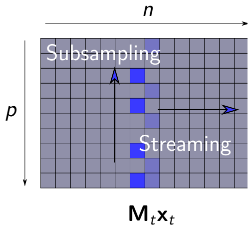
Dataset	Test RMSE		Speed
	CD	**MODL**	-up
ML 1M	0.872	**0.866**	$\times$0.75
ML 10M	0.802	**0.799**	$\times$**3.7**
NF (140M)	0.938	**0.934**	$\times$**6.8**

- Outperform coordinate descent beyond 10M ratings
- Same prediction performance
- Speed-up 6.8$\times$ on Netflix

Conclusion

Take-home message

Loading stochastic subsets of sample streams can drastically accelerates online matrix factorization



- Reduce CPU (+IO) load at each iteration
- *cf* Gradient Descent vs SGD

An order of magnitude speed-up on two different problems

- Python package `http://github.com/arthurmensch/modl`
- Heuristic at contribution time
- A follow-up algorithm has convergence guarantees

Questions ? (Poster # 41 this afternoon)

Bibliography I

[Halko et al., 2009] Halko, N., Martinsson, P.-G., and Tropp, J. A. (2009).
Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions.
arXiv:0909.4061 [math].

[Johnson and Lindenstrauss, 1984] Johnson, W. B. and Lindenstrauss, J. (1984).
Extensions of Lipschitz mappings into a Hilbert space.
Contemporary mathematics, 26(189-206):1.

[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.

Bibliography II

[Pilanci and Wainwright, 2014] Pilanci, M. and Wainwright, M. J. (2014).
Iterative Hessian sketch: Fast and accurate solution approximation for constrained least-squares.
arXiv:1411.0347 [cs, math, stat].

[Szabó et al., 2011] Szabó, Z., Póczos, B., and Lorincz, A. (2011).
Online group-structured dictionary learning.
In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2865–2872. IEEE.

[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.

Appendix

Collaborative filtering

Streaming uncomplete data

- $\mathbf{M}_t$ is *imposed* by user t
- Data stream : $\mathbf{M}_t \mathbf{x}_t$ movies ranked by user t
  - Proposed by [Szabó et al., 2011]), with $\mathcal{O}(p)$ complexity

Validation: Test RMSE (rating prediction) vs CPU time
Baseline: Coordinate descent solver [Yu et al., 2012] solving related loss

$$\sum_{i=1}^{n} (\|\mathbf{M}_t(\mathbf{X}_t - \mathbf{D}\boldsymbol{\alpha}_t)\|_2^2 + \lambda \|\boldsymbol{\alpha}_t\|_2^2) + \lambda \|\mathbf{D}\|_2^2$$

- Fastest solver available apart from SGD – no hyperparameters
- Our method is not sensitive to hyperparameters

Algorithm

Our algorithm

1. Code computation

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

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

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

Original online MF

1. Code computation

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

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

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