université PARIS-SACLAY

NNT : 2018SACLS300

Thèse de doctorat

Apprentissage de représentations en imagerie fonctionnelle

Thèse de doctorat de l'Université Paris-Saclay
préparée à Université Paris-Sud et Inria

Ecole doctorale n°580 Sciences et technologies de l'information et de la communication (ED STIC)
Spécialité de doctorat : Mathématiques et Informatique

Thèse présentée et soutenue à Gif-sur-Yvette, le 28 septembre 2018, par

ARTHUR MENSCH

Composition du Jury :

Jalal Fadili Professeur, ENSICAEN, GREYC, CNRS	Rapporteur – président
Bin Yu Professeure, UC Berkeley, États-Unis	Rapporteure – absente
Moritz Große-Wentrup Professeur, LMU München, Allemagne	Examinateur
Aapo Hyvärinen Professeur, UCL, Gatsby Institute, Royaume-Uni	Examinateur
Bertrand Thirion Directeur de recherche, Inria, CEA, équipe Parietal	Directeur de thèse
Gaël Varoquaux Chargé de recherche, Inria, CEA, équipe Parietal	Co-encadrant de thèse
Julien Mairal Chargé de recherche, Inria, équipe Thoth	Co-encadrant de thèse

LEARNING REPRESENTIONS FROM FUNCTIONAL MRI DATA

ARTHUR MENSCH

Dissertation submitted in partial fulfillement of the requirements for the degree of Doctor of Philosophy.

Université Paris-Saclay
Sciences et Technologies de l'Information et de la Communication

September 2015 – September 2018
Parietal team, Inria Saclay / Neurospin, CEA. France

PH.D. COMMITTEE

DIRECTOR:
Pr. Bertrand Thirion, Inria, CEA, Parietal team, Saclay, France

SUPERVISORS:
Pr. Julien Mairal, Inria, Thoth team, Grenoble, France
Pr. Gaël Varoquaux, Inria, CEA, Parietal team, Saclay, France

REVIEWERS:
Pr. Jalal Fadili, ENSICAEN, GREYC, CNRS, Caen, France
Pr. Bin Yu, University of California, Berkeley, USA

EXAMINERS:
Pr. Moritz Große-Wentrup, LMU München, Germany
Pr. Aapo Hyvärinen, University College London, UK

JURY PRESIDENT:
Pr. Jalal Fadili, ENSICAEN, GREYC, CNRS, Caen, France

This thesis was prepared in Parietal team, at Inria Saclay and Neurospin, CEA, from September 2015 to September 2018. It was funded by a grant from the Ministère de l'Enseignement Supérieur et de la Recherche.

© Arthur Mensch, September 2018

ABSTRACT

Thanks to the advent of functional brain-imaging technologies, cognitive neuroscience is accumulating maps of neural activity responses to specific tasks or stimuli, or of spontaneous activity. In this work, we consider data from functional Magnetic Resonance Imaging (fMRI), that we study in a machine learning setting: we learn a model of brain activity that should generalize on unseen data. After reviewing the standard fMRI data analysis techniques, we propose new methods and models to benefit from the recently released large fMRI data repositories. Our goal is to learn richer representations of brain activity. We first focus on unsupervised analysis of terabyte-scale fMRI data acquired on subjects at rest (resting-state fMRI). We perform this analysis using matrix factorization. We present new methods for running sparse matrix factorization/dictionary learning on hundreds of fMRI records in reasonable time. Our leading approach relies on introducing randomness in stochastic optimization loops and provides speed-up of an order of magnitude on a variety of settings and datasets. We provide an extended empirical validation of our stochastic subsampling approach, for datasets from fMRI, hyperspectral imaging and collaborative filtering. We derive convergence properties for our algorithm, in a theoretical analysis that reaches beyond the matrix factorization problem. We then turn to work with fMRI data acquired on subject undergoing behavioral protocols (task fMRI). We investigate how to aggregate data from many source studies, acquired with many different protocols, in order to learn more accurate and interpretable decoding models, that predicts stimuli or tasks from brain maps. Our multi-study shared-layer model learns to reduce the dimensionality of input brain images, simultaneously to learning to decode these images from their reduced representation. This fosters transfer learning in between studies, as we learn the undocumented cognitive common aspects that the many fMRI studies share. As a consequence, our multi-study model performs better than single-study decoding. Our approach identifies universally relevant representation of brain activity, supported by a few task-optimized networks learned during model fitting.

Finally, on a related topic, we show how to use dynamic programming within end-to-end trained deep networks, with applications in natural language processing.

RÉSUMÉ

Grâce aux avancées technologiques dans le domaine de l'imagerie fonctionnelle cérébrale, les neurosciences cognitives accumulent une grande quantité de cartes spatiales décrivant de manière quantitative l'activité neuronale suscitée dans le cerveau humain en réponse à des tâches ou des stimuli spécifiques, ou de manière spontanée. Dans cette thèse, nous nous intéressons plus particulièrement aux données issues de l'imagerie par résonance magnétique fonctionnelle (IRMf), que nous étudions dans un cadre d'apprentissage statistique. Dans ce cadre notre objectif est d'apprendre des modèles d'activité cérébrale à partir des données. Nous proposons différentes nouvelles manières de profiter de la grande quantité de données IRMf disponible.

Tout d'abord, nous considérons les données d'IRMf de repos, que nous analysons grâce à des méthodes de factorisation de matrices. L'utilisation de ce type de méthode est classique dans un contexte d'apprentissage statistique non-supervisée. Dans le cas de l'IRM fonctionnelle, l'objectif est d'extraire des données un nombre réduit de cartes du cerveau (dites de réseaux fonctionnels) sur lesquelles les données peuvent être projetées avec une faible perte de signal. Les réseaux obtenus (que nous souhaitons parcimonieux) délimitent des régions cérébrales dans lesquelles le signal d'activation est fortement corrélé. Malheureusement, la taille des données des nouvelles études d'IRM fonctionnelle de repos (plusieurs millions d'image tridimensionelles, qui contiennent plusieurs centaines de milliers de voxels chacune) rend très coûteux la décomposition de ces données via une factorisation matricielle, et donc l'extraction de réseaux fonctionnels informés par une quantité de données inédite à ce jour.

En conséquence, nous présentons de nouvelles méthodes pour calculer en un temps raisonnable une factorisation parcimonieuse d'une matrice de donnée constituée plusieurs centaines d'enregistrements d'IRMf. En premier lieu, nous proposons d'effectuer un prétraitement des données d'entrée à l'aide de projections aléatoires, avant d'apprendre une décomposition matricielle depuis les données réduites. Si cette méthode nous permet de traiter en moins d'une journée des données d'une taille de l'ordre de 50 Go, elle n'est pas adaptée pour procéder à l'extraction de réseaux à partir de récents jeux de données de plusieurs téra-octets, comme celui du *Human Connectome Project*, qui propose des enregistrements pour plus de mille sujets.

Notre méthode principale, proposée en deuxième partie, introduit une réduction aléatoire de la dimension des données, via un sous-échantillonage, dans une boucle d'apprentissage en ligne qui résoud le problème de factorisation parcimonieuse. L'algorithme proposé converge plus de 10 fois plus vite que les meilleures méthodes existantes, pour différentes configurations et sur plusieurs jeux de données. Nous effectuons une vaste validation expérimentale de notre approche de sous-échantillonnage aléatoire. Nous proposons une étude

théorique et asymptotique des propriétés de convergence de notre algorithme, dans le cadre plus général des algorithmes de majorisation-minimisation.

Dans un troisième temps, nous nous intéressons aux données d'IRMf d'activation. Nous démontrons comment agréger différents études acquises suivant des protocoles distincts afin d'apprendre des modèles joints de décodage plus justes et interprétables. Notre modèle multi-études apprend à réduire la dimension des images cérébrales en entrée en même temps qu'il apprend à les classifier, pour chacune des études, à partir de leurs représentations réduites. Cela suscite un transfert d'information entre les études. En conséquence, notre modèle multi-étude est plus performant que les modèles de décodage appris sur chaque étude séparément. Notre approche identifie une représentation universellement pertinente de l'activité cérébrale, supportée par un petit nombre de réseaux optimisés pour l'identification de tâches.

Pour finir, sur un sujet connexe, nous nous intéressons à de nouvelles méthodes pour effectuer de la prédiction structurée, avec des applications variées en traitement du langage naturel. Nous proposons une manière générique de relâcher les algorithmes de programmation dynamique qui apparaissent dans les méchanismes d'inférence pour la prédiction de structures (par exemple, l'étiquettage syntaxique d'une phrase). Cela permet d'entraîner les représentations intermédiaires, paramétrées par des réseaux de neurones profonds, des données d'entrée en aval de ces méchanismes d'inférence.

ACKNOWLEDGMENTS

I warmly thank Pr. Jalal Fadili and Pr. Bin Yu, who accepted to review this manuscript, as well as Pr. Eric Moulines, Pr. Moritz Große-Wentrup and Pr. Aapo Hyvärinen, for accepting to be part of my defense jury.

I am most grateful to my three supervisors, Bertrand Thirion, Gaël Varoquaux and Julien Mairal, for guiding me throughout these three years. You were extraordinarily available for advise, provided great directions of research, showed a great patience when it came to teach me how to write, how to prove things, how to gain insight on what we were doing. Working at your side taught me rigor, perseverance and hard-work, and I hope to live up to your teachings in the future.

I am indebted to Mathieu Blondel, with whom it was a joy to work in Kyoto and alongside whom I discovered other aspects of machine learning. My three years of peregrinations owe much to the great people of Parietal team. I am thankful to Mehdi Rahim, Alexandre Abraham and Michael Eickenberg who were great mentors and examples, to Jérôme Dockès who joined me at Parietal after so many years of friendship, to Carole Lazarus, Loubna El Gueddari and Hamza Cherkaoui for being great coffee sharers beyond the first-floor/second-floor rift, to Pierre Ablin, Mathurin Massias and Jérôme-Alexis Chevalier for being zealous co-animators of the Club des amateurs du plateau de Saclay, to Patricio Cerda for being a great first person to meet every morning in Porte d'Orléans, to Kamalaker Reddy Dadi for being a most reliable colleague, to Thomas Moreau for often sparring me reviews in conferences, to Guillaume Lemaître, Joan Massich Vall and Joris van den Bossche for forming a most enjoyable agile software team, to André Manoel, Ana Luísa Pinho, Loïc Estève, Daria La Rocca, Darya Chyzhyk, Denis Engemann and Demian Wassermann who were great people to discuss with. I thank Alexandre Gramfort, Joseph Salmon and Philippe Ciuciu, who were knowledgeable advisors and great supporters during my stay. I thank Régine Bricquet, Tiffany Caristan, Corinne Petitot and Stéphanie Druetta for their efficiency at handling administrative issues.

I dearly thank Elvis Dohmatob and Olivier Grisel who, besides being great friends, taught me a lot lot and formed superb collaborators. I thank Alberto Bietti, Anna Korba, Eugène N'Diaye, Thibaud Rahier without whom conferences and summer schools would not have been as fun, and who were great companions in the machine learning community. Finally, I thank my friends from every horizon (fortunately not everyone does machine learning for a living yet), my family, my parents, brothers and sister, without whom none of this would have been possible. And of course, Émilie, of infinite patience, for whom ça n'a pas dû être facile tous les jours.

CONTENTS

1 OVERVIEW 12
1.1 Organization of the manuscript . . . 12
1.2 A note on chapter ordering . . . 15

I (MF FOR) FUNCTIONAL NEURO-IMAGING

2 NEURO-IMAGING BACKGROUND 17
2.1 Studying the brain through functional MRI . . . 17
2.2 Resting-state functional MRI . . . 19
2.3 Task fMRI data analysis . . . 22
2.4 Conclusion . . . 27
3 DICTIONARY LEARNING FOR FMRI 28
3.1 Matrix factorization for resting-state fMRI . . . 28
3.2 Dictionary learning for resting-state fMRI . . . 30
3.3 Time-compressed dictionary learning . . . 31
3.4 Validation and results of compressed DL . . . 34
3.5 Changing model and going beyond . . . 39

II HUGE MATRIX FACTORIZATION

4 STOCHASTIC SUBSAMPLING FOR HUGE MATRIX FACTORIZATION 44
4.1 Overview of Part II . . . 44
4.2 Background and proposed approach . . . 45
4.3 Prior art: online matrix factorization . . . 47
4.4 Algorithm outline . . . 50
4.5 Subsampled online matrix factorization . . . 52
5 SOMF ALGORITHM PROPERTIES 60
5.1 Prior art: stochastic majorization-minimization . . . 60
5.2 Stochastic approximate majorization-minimization . . . 62
5.3 Convergence analysis . . . 62
5.4 Conclusion . . . 68
6 SUBSAMPLED ONLINE MATRIX FACTORIZATION IN PRACTICE 69
6.1 Experiments with SOMF . . . 69
6.2 Extension to matrix completion . . . 76
6.3 Conclusion of Part II . . . 80

III DEEPER MODELS FOR MULTI-STUDY COGNITIVE MAPPING

7 LEARNING MULTI-STUDY NEURAL REPRESENTATIONS OF COGNITION 83
7.1 Introduction . . . 83
7.2 Results . . . 85
7.3 Discussion . . . 91
7.4 Detailed method . . . 94
7.5 Design discussion . . . 99
7.6 Data corpus and references . . . 107

IV NEW ALGORITHMIC LAYERS FOR DEEP STRUCTURE PREDICTION

8 DIFFERENTIABLE DYNAMIC PROGRAMMING 110
8.1 Introduction . . . 111
8.2 Smoothed max operators . . . 112
8.3 Differentiable dynamic programming layers . . . 113
8.4 Examples of computational graphs . . . 118
8.5 Differentiable structured prediction . . . 121
8.6 Structured and sparse attention . . . 125
8.7 Conclusion . . . 126

V CONCLUSION

9 CONCLUSION 129
9.1 Software . . . 130
Bibliography 131

Appendices

A PROOFS OF CHAPTER 7 — SOMF AND SAMM ANALYSIS 148
A.1 Proofs of convergence . . . 148
B PROOFS AND RESULTS FROM CHAPTER 8 — DIFFERENTIABLE DP 163
B.1 Proofs and detailed derivations . . . 163
B.2 Examples of algorithm instantiations . . . 172
B.3 Experimental details and further results . . . 176

ACRONYMS

ACC	Anterior cingulate cortex
ADHD	Attention Deficit Hyperactivity Disorder
BART	Balloon analog risk taking
BCD	Block coordinate descent
BLEU	Bilingual evaluation understudy score
BOLD	Blood-oxygen-level dependant
CRF	Conditional random field
CPU	Central processing unit
DAG	Directed acyclic graph
DL	Dictionary learning
DLPFC	Dorsolateral prefrontal cortex
DP	Dynamic programming
DTW	Dynamic time warping
ECOG	Electrocorticography
EEG	Electro-encephalography
FISTA	Fast iterative soft thresholding algorithm
FFA	Fusiform face area
fMRI	Functional magnetic resonance imaging
GLM	General linear model
GPU	Graphics Processing Unit
HCP	Human Connectome Project
HRF	Hemodynamic-response function
ICA	Independent component analysis
IO	In-out
IPS	Intraparietal sulcus
IRMf	Imagerie à résonance magnétique fonctionnelle
L-BFGS	Limited memory Broyden–Fletcher–Goldfarb–Shanno algorithm
LDA	Latent discriminant analysis
LSTM	Long-short term memory network
MAP	Maximum a posteriori
MEG	Magneto-encephalography
MF	Matrix factorization
MFCC	Mel-frequency cepstral coefficients
MNI	Montreal Neurological Institute
MRI	Magnetic resonance imaging
MSTON	Multi-study task-optimized networks
NER	Named entity recognition
NMF	Non-negative matrix factorization
OMF	Online matrix factorization
PCA	Principal component analysis
RMSE	Root mean square error
SAGA	Stochastic average gradient amélioré
SAMM	Stochastic approximate majorization-minimization
SGD	Stochastic gradient descent

SMM	Stochastic majorization-minimization
SOMF	Stochastic online matrix factorization
SPCA	Sparse principal component analysis
SVD	Singular value decomposition
SVM	Support vector machine
SVR	Support vector regression
UKBB	UK BioBank

NOTATION

We denote scalars, vectors and matrices using lower-case, bold lower-case and bold upper-case letters, *e.g.*, x, $\mathbf{x}$ and $\mathbf{X}$. We denote the elements of $\mathbf{X}$ by $x_{i,j}$, its rows by $\mathbf{x}_i$, and its columns by $\mathbf{x}^{(i)}$. Depending on context, subscript will also be used to denote iteration number, as in $\mathbf{x}_t$, the value of $\mathbf{x}$ at iteration t of a given algorithm. We use calligraphic font $\mathcal{X}$ to denote ensembles. When dealing with approximated value and comparing them to a ground truth, we use superscript * to denote the non-approximated value. Finally, we often use the notation $\bar{x}$ to denote an empirical or true expected value. Specific notations will be recalled where needed.

Notation	Name	Definition
$[1, n] = [n]$	Integers from 1 to n	$\{1 \dots, n\}$
$\|\mathbf{x}\|_2$	Vector euclidean norm (ℓ_2 norm)	$\left(\sum_{i=1}^n x_i^2\right))^{1/2}$
$\|\mathbf{x}\|_1$	Vector/matrix ℓ_1 norm	$\sum_{i=1}^n \|x_i\|$
$\langle \mathbf{x}, \mathbf{y} \rangle$	Vector scalar product	$\sum_{i=1}^n x_i y_i$
$\operatorname{supp}(\mathbf{x})$	Support of $\mathbf{x}$ in $\mathbb{R}^n$	$\{j \in [1, n] : x_j \neq 0\}$
$\|\mathbf{X}\|_F^2$	Matrix Frobenius norm	$\left(\sum_{i,j1=}^{n,m} x_{i,j}^2\right))^{1/2}$
$\langle \mathbf{X}, \mathbf{Y} \rangle$	Frobenius scalar product	$\sum_{i,j=1}^{n,m} x_{i,j} y_{i,j}$
$\operatorname{Tr} \mathbf{X}$	Trace of $\mathbf{X}$	$\sum_{i=1}^n x_{i,i}$
$\operatorname{Diag}(\mathbf{x})$	Diagonal matrix with diagonal $\mathbf{x}$	
$\mathbf{X}^\dagger$	Moore-Penrose pseudo-inverse	
$\mathcal{B}_{1/2}$	ℓ_1/ℓ_2 unit ball in $\mathbb{R}^n$	
$\operatorname{conv}(\mathcal{Y})$	Convex hull of $\mathcal{Y}$	
$H(\mathbf{q})$	Shannon Entropy	$\sum_i^n q_i \log q_i$
$\triangle^n$	$(n-1)$-probability simplex	$\{\boldsymbol{\lambda} \in \mathbb{R}_+^D : \|\boldsymbol{\lambda}\|_1 = 1\}$
$\mathcal{N}(\mu, \boldsymbol{\Sigma})$	Normal distribution	
$\mathbb{P}[A]$	Probability of event A	
$\mathbb{E}[X]$	Expected value of X	$\sum_A \mathbb{P}[X = x]x$

1 OVERVIEW

Functional MRI is a powerful brain-imaging modality: it allows to better understand how the brain work by recording with very little intrusion the brain activity of an active human subject with a good time (1s) and spatial resolution (1mm). The field of fMRI is becoming data intensive, as the number of publicly available studies is constantly growing, and as several acquisition campaigns on large cohorts have provided tens of thousand hours of brain records. This has called for changing analysis methods, that have been shifting from inferential statistics to statistical learning for the last 10 years. It is now consensual that models of brain activity recorded in fMRI should be *learned* from data, and validated by performing prediction on *left-out* data, which cast fMRI analysis as a machine-learning challenge. The size of newly released fMRI data requires strong adaptation of existing machine learning techniques, given their unusual shape: they are high dimensional (with hundreds of thousands of voxels), and come in numerous samples with low signal-to-noise ratio. In this thesis, we specifically address the problem of efficiently finding rich representation of brain activity using large-scale fMRI data repositories.

1.1 ORGANIZATION OF THE MANUSCRIPT

The following work is organized around three major research directions, that led to different series of publications.

1.1.1 *(Matrix factorization for) functional imaging analysis*

What is functional Magnetic Resonance Imaging (fMRI), how is functional MRI data analysed today, why does the growing amount of data requires new methods? We provide an overview of functional MRI analysis, in both unsupervised (resting-state data) and supervised (task data) settings in Chapter 2, which introduces the several data analysis formalisms we reuse throughout this work. In Chapter 3, we propose a new method based on random projections (Halko et al., 2011; Johnson and Lindenstrauss, 1984) to preprocess data and accelerate the extraction of functional networks from resting-state fMRI data using dictionary learning (Olshausen and Field, 1997; Varoquaux et al., 2011). This method is useful but hard to deploy on datasets with thousands of fMRI acquisitions: to circumvent this issue, and propose a new problem formalization, that will be central to Part II.

Published work

Mensch, A., Varoquaux, G., & Thirion, B. (2016b). Compressed online dictionary learning for fast fMRI decomposition. In *Proceedings of the IEEE International Symposium on Biomedical Imaging (ISBI)*,

Dohmatob, E., Mensch, A., Varoquaux, G., & Thirion, B. (2016). Learning brain regions via large-scale online structured sparse dictionary learning. *Advances in Neural Information Processing Systems*.

1.1.2 *Huge matrix factorization*

Matrix factorization plays a major role in functional MRI analysis, especially when dealing with resting-state data, *i.e.* data acquired on subjects left idle in their scanner. The size of the data produced by fMRI studies sets new scalability challenges for this category of pattern extraction methods. How can we factorize large (high dimensional) and tall (numerous samples) matrices in reasonable time? In Chapter 4, we propose a new flexible algorithm for matrix factorization, that is an order of magnitude faster than the fastest existing methods (Mairal et al., 2010) on the datasets we consider. Our algorithm, that we dub SOMF, is able to factorize huge, dense, and potentially square matrices, into factors that may be sparse, dense and potentially non-negative. It relies on a new optimization method that is both random in the sample (column) and feature (row) direction. We provide a complete theoretical analysis of the properties of SOMF in Chapter 5, in which we extend the class of stochastic majorization-minimization algorithms (Mairal, 2013b) by perturbating their various steps. We show that it enjoys the same convergence properties as existing methods. We perform an empirical validation of SOMF for a variety of domains in Chapter 6, where we discuss how it accelerates matrix factorization for hyperspectral imaging data, collaborative filtering (R. M. Bell and Koren, 2007), and of course resting-state fMRI.

Published work

Mensch, A., Mairal, J., Thirion, B., & Varoquaux, G. (2016a). Dictionary learning for massive matrix factorization. *Proceedings of the International Conference on Machine Learning (ICML)*,

Mensch, A., Mairal, J., Thirion, B., & Varoquaux, G. (2018b). Stochastic Subsampling for factorizing huge matrices. *IEEE Transactions on Signal Processing*, *66*(1), 113–128.

1.1.3 *Deeper and richer models for task fMRI and structured data*

For a bird's-eye view, functional MRI data is available in two major forms: a few large-scale studies (*e.g.*, Sudlow et al., 2015; Van

Essen et al., 2012), that perform resting-state and generic task protocols over hundreds to thousands of subjects; hundreds of smaller task fMRI studies, that explore specific aspects of cognition by applying carefully designed but disparate experimental protocols on a few dozens of subjects, who are asked to execute some tasks or stimulated in various ways.

The statistical power of data analysis methods for these smaller studies is limited by their sample size (Button et al., 2013). On the other hand, large efforts have been recently made to gather many small fMRI in the same public repositories (Poldrack et al., 2013). Can aggregating data from many small size sources and leveraging the large-size datasets allow us to learn more powerful models ? Many task fMRI studies indeed share some common cognitive aspects, which should allow to increase the effective classification power of learned models. Yet a major challenge for this approach lies in the fact that the relationships between various studies are undocumented. In Chapter 7, we design a new multi-layer model that performs decoding over dozens of studies aggregated together. The multi-layer, shared-parameter structure of our model allows to learn the relationship between protocols and permits effective transfer learning. In other words, it ensures that the information learned from classifying each study benefits the other studies. Our model has higher prediction performance than single-study decoders; it produces cognitive representation of brain activity over *multi-study task-optimized networks*, that form a universal and interpretable basis for inter-subject decoding.

Published work

> Mensch, A., Mairal, J., Bzdok, D., Thirion, B., & Varoquaux, G. (2017). Learning neural representations of human cognition across many fMRI studies. *Advances in Neural Information Processing Systems*.

1.1.4 *New algorithmic layers for deep structure prediction*

This last part, that may be considered as an appendix, was prepared during an internship at NTT Communication Science Labs, Kyoto, Japan. In Chapter 8, we depart from fMRI and linger on the idea, already present in Chapter 7, of introducing new components to existing models and train them in an end-to-end fashion. More precisely, we provide a general approach for introducing dynamic programming mechanisms within deep networks, and show how to make these mechanisms differentiable and therefore suitable for backpropagation training. Our approach, based on smoothing techniques (Nesterov, 2005), allows to perform dense or sparse inference within (simple) graphical models and backpropagate through it. We show how to apply it in natural language processing settings (neural machine translation, named entity recognition), and audio-to-score alignment.

Published work

> Mensch, A., & Blondel, M. (2018). Differentiable dynamic programming for structured prediction and attention. *Proceedings of the International Conference on Machine Learning (ICML)*.

1.1.5 *Software*

We developed several *Python* packages for reproducibility and reuse of the work presented in this dissertation. The detailed list is provided in Chapter 9.

1.2 A NOTE ON CHAPTER ORDERING

Appendices may be skipped at first read as they are not essential for understanding the overall story. The reader more interested in more theoretical machine learning and optimization would first go for chapters 4, 5 and 8. In contrast, chapters 3, 6 and 7 are more specifically focused on machine learning for fMRI.

Part 1

(MATRIX FACTORIZATION FOR) FUNCTIONAL NEURO-IMAGING

2 NEURO-IMAGING BACKGROUND

In this chapter, we introduce the general goals and the type of data that were studied in this thesis. This is useful to understand the various directions that we took. The general objective of this thesis was the analysis of large-scale functional Functional magnetic resonance imaging (fMRI) data. Its driving motivation could be stated as such: we wish to handle at reasonable cost the numerous studies that are now publicly available, in order bring more precise and general statistical representation of brain activity. Yet new methods and models have to be developed to 1) handle the size of data that ambitious data acquisition projects are now producing and 2) handle the protocol variability of past and future functional MRI studies, by increasing model complexity.

In this chapter, we provide a synthetic overview (Section 2.1) of the purpose of functional MRI in cognitive science, so as to make clearer the final interest of the presented methods to the reader. We refer to *e. g.,* Poldrack et al. (2011) for more in-depth reference. Functional MRI protocols come in two major flavors, task and resting-state, that were both studied in this thesis. We will present their general principles and their associated analysis techniques in Section 2.2 and 2.3.

This chapter is focused on neuroimaging rather than mathematical formulations: we delay the introduction of models and methods and their inscription within a larger machine-learning/signal processing literature to the relevant following chapters.

2.1 STUDYING THE BRAIN THROUGH FUNCTIONAL MRI

Let us first recall the basis of neuro-imaging, before focusing on functional MRI principles and technical aspects.

2.1.1 *Neuro-imaging*

Neuro-imaging sciences endeavor to measure brain activity from human or animal subjects and relate it to experimental conditions and behavior observations. It is founded on the observation[1] that cognition has measurable effects on the brain, that are somehow shared across subjects, and to some extent across species. In theory, this should allow to *map* the cognitive functions onto the physical brain and describe cognitive processes implementation in quantitative ways, that should reasonably generalize across subjects.

Means of signal acquisition are various and in constant evolution. Table 2.1 describes some of them. Modalities may be considered from different point of views: they vary in their level of intrusion, their spatial resolution, their temporal resolution and their level of noise.

[1] *Neuroscience can be traced back to Ancient Egypt medicine, that already observed relations between brain trauma localisation and behavior (Kandel et al., 1981).*

Modality	Principles	S. res.	T. res.	Intrusiveness
ECOG	Electric field evoked inside the implanted brain	1 mm	10 ms	Surgical intervention
EEG/MEG	Electric/Magnetic field evoked from *surface* activity	1 cm	10 ms	Harmless
fMRI	BOLD signal (whole brain)	2 mm	1 s	High **B** field

Table 2.1 – Examples of neuro-imaging modalities: all varies in resolution and intrusiveness. fMRI is often a good compromise.

Similarly, experimental protocols vary in how much they are close to day-to-day cognitive tasks (looking at blinking dots *versus* looking at a movie) and in the level of the cognitive functions they recruit (hearing beeps *versus* making complex risk-taking decisions).

2.1.2 *Functional Magnetic Resonance Imaging (fMRI)*

In this work, we focus on fMRI, a modality introduced by Ogawa et al. (1990). This mean of observation, which is performed in an MRI scanner, is non intrusive yet it provides a satisfying temporal and spatial resolution. Unlike Electro-encephalography (EEG) and Magneto-encephalography (MEG), functional MRI does not directly measure electric of magnetic field that are stemmed by neurons in activity. In contrast, it leverages Magnetic Resonance Imaging (MRI, Lauterbur, 1973)[2] to measure the variation in the level of oxygenated and deoxygenated blood within the blood vessels that irrigate neurons. As spiking neurons require hemoglobin-provided di-oxygen to produce energy, a neuronal activity increase in any volume of the brain is followed within five second by an increase of oxygenated blood in this volume and by an undershoot that lasts roughly 30 seconds. These oxygen-dependent variations are detectable through Magnetic resonance imaging (MRI), and are extracted as the Blood-oxygen-level dependant (BOLD) signal.

[2] *in itself a powerful modality to discern in between biological tissues with various magnetic properties*

MODELISATION. The biological phenomena at stake are modelled as such: the observed BOLD signal is the result of convoluting the neuronal activity in the volume of interest with an Hemodynamic-response function (HRF), that models delay, amplitude and undershoot of the level of oxygenated blood within this volume. Namely, writing x_v and $\tilde{x}_v : [0, T] \to \mathbb{R}$, the continuous BOLD and neuronal activity that we wish to measure within voxel v, we assume that there is a function $\xi : \mathbb{R}^+ \to \mathbb{R}$ such that

$$x_v(t) = (\tilde{x}_v \circ \xi)(t) \quad \forall t \in [0, T], \tag{2.1}$$

where ξ has either an established form (see *e. g.,* Lindquist et al., 2009), or may be estimated from data (Makni et al., 2008; Pedregosa et al., 2015). This modelling is fundamentally linear, in the sense that the BOLD signal is assumed to be proportional to the neuronal activity. It will be central in task fMRI analysis (see Section 2.3). At the

end of the fMRI acquisition, we obtain a sequence of brain images, *i. e.* one time-series per voxel that records the intensity of the neural activity, convoluted by the hemodynamic response, within this voxel, plus noise from various sources. Typically, these time-series have a period of 0.8 to 3 seconds, and the space resolution (*i. e.* the volume of each voxel) varies from 1 mm^3 to 27 mm^3, depending on the spatial resolution that the scanner allows. Note that these volumes contain millions of neurons: although functional MRI has a good spatial resolution compared to other non intrusive modalities, it is still many orders of magnitude above the cellular level.

PREPROCESSING. fMRI raw data must typically be corrected for various noise sources that deteriorate the signal-to-noise ratio of the BOLD time series. Most importantly, subject head motion is recorded and regressed through within-record registration, as are physiological conditions (heart-beat, respiration). The physical artifacts related to the scanning process (*e. g.*, the fact that slices are recorded one after another, causing time jitter, and the non-uniformity of the base magnetic field) are also monitored and compensated in preprocessing steps. Typically, the public datasets on which we developed new techniques are already provided as pre-processed time series for which physiological and physical confounds have already been regressed.

GROUP-LEVEL ANALYSIS. In most functional MRI studies, the same acquisition protocol is performed on different subjects and potentially several times on each subject. Analysis of the BOLD signal may then be performed at a *subject-level*, or at the *group-level*. Depending on our goals, we may choose to model differently the *inter* and *intra-subject* variability of records. To perform group analysis, single-subject brain images are typically registered to a common template (the MNI space, introduced in Evans et al., 1993), so as to reduce the variability in brain shape. Even though this leads to a loss in anatomical information, this approach is motivated by the fact that brain networks are often located within well defined anatomic regions that are shared across subjects, modulo some non-rigid transformation. This work does not focus on inter-subject variability, and will assume that brain images arise from a distribution that is shared across subjects, once they have been registered to the MNI space.

2.2 RESTING-STATE FUNCTIONAL MRI

Resting-state fMRI data are central in modern fMRI analysis as it is cheap to acquire and contains much intrinsic information about brain functioning — we review it briefly, as it is central in this thesis.

2.2.1 *Protocol description*

The simplest way to obtain functional MRI data is indeed to follow the resting-state protocol. As its name indicates, it consists in acquir-

ing the BOLD signal from a subject that has been asked to *rest* in the scanner, namely to do nothing in particular. No stimuli nor specific task is provided. This data acquisition process yields *unlabelled data*, in the sense that nothing is known as to the thought process that is going on in the subject mind. These data may be thought as *brain movies*, with 1 image per second, and in between 50,000 to 200,000 voxels per image. Resting-state protocol is the cheapest way to acquire fMRI data: as such, it is widely available: the Human Connectome Project (HCP, Van Essen et al., 2012) provides 4,000 records of 15 minutes acquired across 1,000 subjects, while the UK BioBank (UKBB) initiative (Sudlow et al., 2015) endeavors to gather data for 100,000 subjects.

2.2.2 *Scientific purpose*

Resting-state data contain interesting information regarding the subject brain. They allow to identify various *functional networks*, that correspond to spatial regions that tend to activate together. With these networks at hand, we hope to reduce the dimensionality of the signal from 10^5 voxels to a few hundreds components without loosing cognitive information. Uncovering these functional networks at scale is the driving motivation of a part of this thesis (Chapter 3 and 4), while we show how to use these networks in new decoding pipelines (see Section 2.3) in Chapter 7.

Historically, the functional networks uncovered from resting-state data have first been used to construct bio-markers for certain diseases: Alzeimer disease (Greicius, 2008), Parkinson disease (Wu et al., 2009), autism spectrum disorder (Abraham et al., 2017; Weng et al., 2010), Attention Deficit Hyperactivity Disorder (ADHD, Yu-Feng et al., 2007). It has also been shown to be related to behavior, *e. g.,* fluid intelligence (S. M. Smith et al., 2015). More precisely, the time correlation structure between the various functional networks is often of interest to better understand how the brain of a single subject functions. These correlations can be estimated in the framework of *functional connectivity* (Biswal et al., 1995; Fox and Raichle, 2007), that is still being refined (*e. g.,* Rahim et al., 2017).

Recently, many studies have also demonstrated the interest of using resting-state data to better frame inter-subject variability in more complex protocols involving controlled stimuli (Sabuncu et al., 2009; S. M. Smith et al., 2009). At a more fundamental level, resting-state analysis is central to better understand the role of functional networks in cognition, *e. g.,* the default-mode network (Greicius et al., 2003), a pre-eminent network in resting-state that tends to activate during mind-wandering, or amygdala (Roy et al., 2009), a brain region partly responsible for emotional responses.

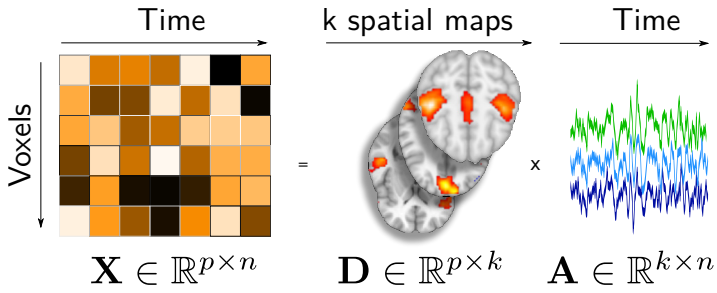


Figure 2.1 – Modelling of the brain signal in resting-state fMRI. The data matrix writes as a product of two matrices that yield spatial and temporal information respectfully.

2.2.3 *Linear modeling and functional networks*

Due to their very indirect nature, BOLD images are very noisy: it is estimated that cognitive tasks that subjects undertake only explains 5% of the variance of the voxel time-series (Raichle and Mintun, 2006). When trying to use BOLD data to learn a statistical model of the brain, we thus have to come up with models of reasonable complexity. Uncovering *functional networks* from resting-state images is thus often performed using the simplest model possible: brain images are assumed to be the linear combination of spatial functional networks (that can be seen template brain images), that are fixed across time. Mathematically, this can be written as follow: brain images containing the bold signal form a sequence in $\mathbb{R}^p$, that we denote $(\mathbf{x}_t)_{t \in [n]}$, where n is the number of single frames and p the number of voxels. We assume that the images from this sequence are approximately a linear combination of k spatial maps in $\mathbb{R}^p$ — the functional networks, written $(\mathbf{d}_j)_{j \in [k]}$, so that

$$\mathbf{x}_t = \sum_{j=1}^{k} \alpha_{t,j} \mathbf{d}_j + \boldsymbol{\epsilon}_t, \tag{2.2}$$

where $\boldsymbol{\alpha}_t \in \mathbb{R}^k$ denotes the loadings at time t, associated with the networks $(d_j)_j$ and $\epsilon_t \in \mathbb{R}^p$ is a residual term. The spatial maps, gathered in a matrix $\mathbf{D} \in \mathbb{R}^{p \times k}$, will be *estimated* from data $\mathbf{X} \in \mathbb{R}^{p \times n}$, so as to minimize residuals and enforce some meaningful properties over $\mathbf{D}$. The model is illustrated in Figure 2.1. Estimating $\mathbf{D}$ will be the purpose of Chapter 3 and 6.

Once again, in multi-subjects studies, we may choose whether or not to model inter-subject variability or consider it as a confound. Typically, we may choose to estimate a different dictionary $\mathbf{D}_s$ for each subject s, or simply consider each brain image from each subject as a linear combination of the same dictionary $\mathbf{D}$. We follow the latter path in this work, as it is more readily amenable to recent datasets with thousands of subjects.

2.3 TASK FMRI DATA ANALYSIS

Functional MRI may also be acquired with more complex protocols, that involve presenting sensitive stimuli or ask the acquired subject to perform a task. Performing such *task* fMRI study typically requires to design of an experimental protocol that will be presented to the subject several time during acquistion. The objective of this approach is to relate the presented stimuli or the observed reactions of the subject (*i. e.* experimental *base condition*) to the subject's brain activity — this is known as *encoding*. Reversely, we may wish to predict base conditions from brain images, namely to *decode* the brain into experimental conditions.

2.3.1 *Encoding stimuli into the brain: standard fMRI analysis*

Encoding tries to relate a *base condition* sequence (*e. g.*, sequences of different images presented to the subject) to brain activity. For this, we once again use a *linear* model, in what is known as the General linear model (GLM) in the literature (Friston et al., 1994). Typically, conditions can be modelled as multi-dimensional time-series $x : [0, T] \to \mathbb{R}^k$, where k is the number of different stimuli that are presented to the subject during the acquisition and T is the length of the experiment. For example, Haxby et al. (2001b) present images of faces, cats, and scissors to the subject: in this case, the three time series $\tilde{y}_1$, $\tilde{y}_2$ and $\tilde{y}_3$ corresponds to the onsets of face, cat and scissors visualization, respectively. Similarly, functional localizer protocols (*e. g.*, Pinel et al., 2007b) show a variety of stimuli to subjects, that are known to recruit localized parts of the cortex. Base conditions include visual and auditory stimuli, computation and motor commands.

2.3.1.1 *The General Linear Model*

We assume that these stimuli immediately trigger a neuronal response $\tilde{x}_v$ within each voxel v of the subject brain, and that this response is the sum of the response of this voxel to each stimulus $\tilde{y}_i$ for $i \in [k]$. Namely, there exists $\beta \in \mathbb{R}^k$ and an i.i.d. noise time-series ϵ_v such that, for all $t \in [0, T]$,

$$\begin{aligned} \tilde{x}_v(t) &= \sum_{i=1}^{q} \beta_{v,i} \tilde{y}_i(t) + \tilde{\epsilon}_v(t), \quad \text{and thus, convoluting with } \xi, \\ x_v(t) &= \sum_{i=1}^{q} \beta_{v,i} y_i(t) + \epsilon_v(t), \text{ where } \epsilon_v(t) \sim \mathcal{N}(0, \sigma_v^2) \end{aligned} \tag{2.3}$$

where we used the HRF model (2.1) to recover a model relating the BOLD signal x_v to the convoluted signals $(y_i \triangleq \tilde{y}_i \circ \xi)_{i \in [k]}$. Each voxel is thus associated with a vector β_v, that contains the linear susceptibility of that voxel to each condition presented in the protocol. Figure 2.2 illustrates the standard model (2.3) for a single voxel v.

We estimate the $(\beta_v)_v$ from data using linear regression. Namely, we define $X \in \mathbb{R}^{n \times p}$ the discretized matrix of BOLD time-series and

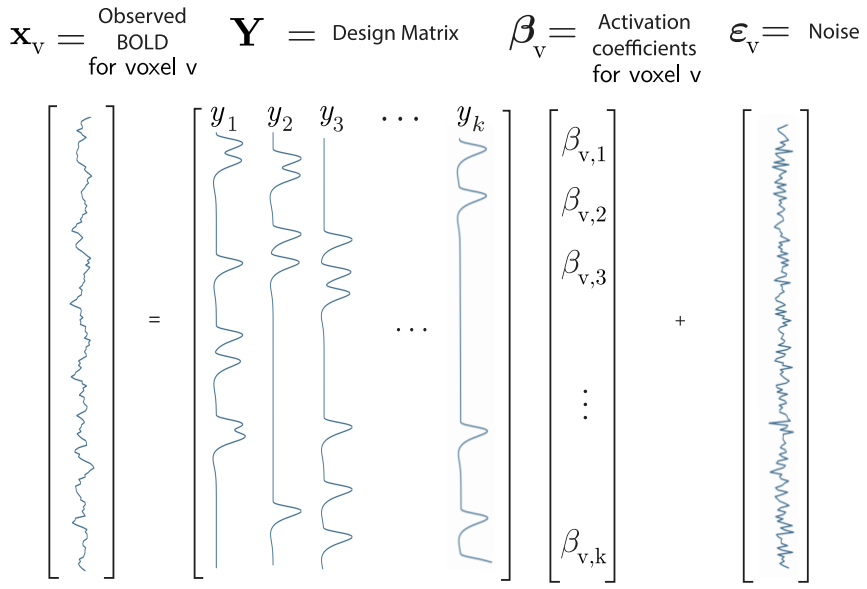


Figure 2.2 – The General Linear Model relates the BOLD signal in each voxel v to the base conditions $(\mathbf{y}_i)_{i\in[k]}$ of the experiments, convoluted by the HRF at this voxel. Those form the design matrix $\mathbf{Y}$. Adapted from Pedregosa (2015).

$\mathbf{Y} \in \mathbb{R}^{n\times k}$ the discretized matrix of HRF-convoluted stimuli $(\mathbf{y}_i)_i$ — known as the *design matrix*[3]. We compute $\hat{\mathbf{B}} = \mathbf{Y}^\dagger\mathbf{X} \in \mathbb{R}^{k\times p}$ (the β-maps), that solve $\min_{\mathbf{B}\in\mathbb{R}^{k\times p}} \|\mathbf{X} - \mathbf{B}\mathbf{Y}\|_F^2$. These maps may be either computed for a single subject, or at the group-level (concatenating experiments for different subject across time).

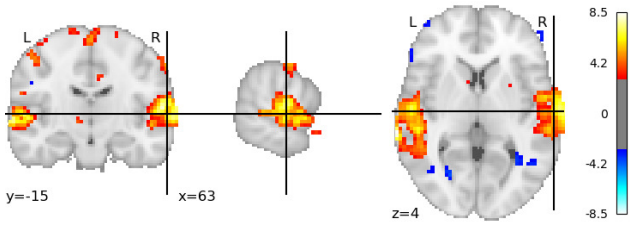
[3] *For coherence with previous and next sections,* $\mathbf{X}$ *and* $\mathbf{Y}$ *are inverted w.r.t. classical notations in linear regression.* $\mathbf{X}$ *is transposed w.r.t. Section 2.2.*

2.3.1.2 *Statistical maps*

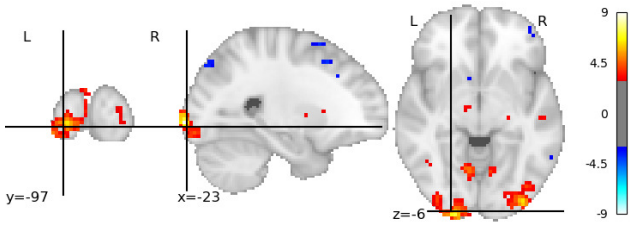
To provide a statistical interpretation of the $\hat{\mathbf{B}}$ maps, those are typically transformed into *z-score* maps. We wish to know how much the susceptibility of voxel v to the condition i differs from zero, compared to the noise in the estimation of regressors. For this, we compute the following t-statistic for each voxel:

$$t_{v,i} = \frac{\hat{\beta}_{v,i}}{\hat{\mathbb{V}}_{\epsilon_v}^{1/2}(\hat{\beta}_{v,i})}, \quad \text{where} \quad \hat{\mathbb{V}}_{\epsilon_v}(\hat{\beta}_{v,i}) = (\mathbf{Y}\mathbf{Y}^\top)^{-1}_{i,i}\,\hat{\sigma}_v^2 \tag{2.4}$$

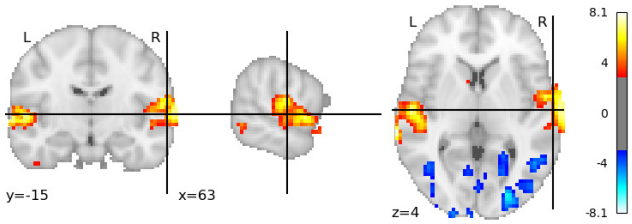
and $\hat{\sigma}_v^2 = \|\mathbf{x}_v - \boldsymbol{\beta}_v\mathbf{Y}\|_2^2/(n-k)$ is estimated from data. Then, t-statistics are converted to z-statistics by applying an Student to Gaussian transform. We thus obtain a set of maps $\boldsymbol{Z} \in \mathbb{R}^{p\times k}$: these maps forms the systematic output of task fMRI studies, analysed from an *univariate* point-of-view. Z-maps may be used to select voxels of interest for a given stimuli. For example, we may want to select the voxels of the map $\mathbf{t}_i \in \mathbb{R}^p$ that differ from zero with associated p-value $p < 0.05$. We thus obtain a selection of voxels that are triggered preferentially by condition i, and a *localization* of the brain regions recruited by this condition. This localization is best represented as a *thresholded* statistical map. We display in Figure 2.3a and 2.3b the z-maps associated with video stimuli and audio stimuli, from a single-subject functional localizer analysis. We may also use non-thresholded z-maps as



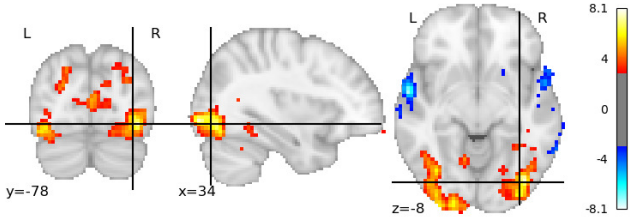
(a) Auditory stimulus



(b) Video stimulus



(c) Audio-video contrast



(d) Video-audio contrast

Figure 2.3 – Base condition and contrast z-maps for a single subject analysis of the functional localizer protocol. Thresholded z-maps outline brain functional regions. Adapted from *Nistats* documentation, with data from Pinel et al., 2007b

inputs of a *decoding* task, as they constitute a summary of the effect of each base conditions on each subject brain — non thresholded maps will be the input data in Chapter 7.

2.3.1.3 *Contrasts*

Neuroscientists often prefer to estimate the z-statistic associated with some *contrast* between regressors, *e.g.*, $\hat{\beta}_{v,1} - \hat{\beta}_{v,2}$ for face-vs-cats in Haxby et al. (2001b) experiment. This approach, known as cognitive subtraction (Petersen et al., 1989), is typically useful to compare two base conditions that share some common aspects (both stimuli are images in previous example) but differs in others (the content of images differs). Contrasting regressors then allow to narrow the condition whose effect is measured — *e.g.*, identify face recognition brain regions. Back to the functional localizer protocol, we also plot the *audio-video* and *video-audio* contrast z-maps in Figure 2.3. The con-

trast z-maps better outline the visual and auditory cortex than base condition z-maps — and are indeed preferred for discussion. Formulas to derive z-maps for contrasts are easily adapted from equation (2.4). [1]

2.3.2 *Decoding brain images into stimuli*

We turn to presenting decoding for task fMRI, as this will provide background for Chapter 7, where we classify z-maps from many maps and subjects.

The standard analysis presented in Section 2.3.1 handles each voxel separately (in a mass univariate setting). It thus captures how different regions of the brain may interact for a specific task, and misses some information (Cox and Savoy, 2003; Haxby et al., 2001b).

Decoding infers more information by identifying task/conditions using *whole brain images* as input. Those brain images should already summarize brain activation either at the subject or the group level, and are typically the z-maps or β-maps obtained running the GLM (Mumford et al., 2012). Decoding thus performs *multi-variate pattern analysis* of voxel activation (see Haynes, 2015, for a review), and provides new information on the relation between brain patterns and stimuli.

2.3.2.1 *Statistical learning formulation*

The problem of task identification is cast as a *classification* or *regression problem*: we wish to learn a model that can detect or predict a specific task from a (group or individual) summary of the effect of several tasks on the brain. Mathematically, this is formulated as follows: we wish to assign a target (stimuli/behavior) $y \in \{0, k\}$ to each brain activation maps (typically, z-maps) $\mathbf{X} \in \mathbb{R}^p$, using a model f_θ, that is fitted to train data $(\mathbf{X}_i, \mathbf{y}_i)_{i \in [n]}$. For example, in the Haxby et al. (2001b) experiment, the target corresponds to the category of the visual stimuli presented to the subject. The *statistical learning* framework (see *e. g.*, Bishop, 1995) demands that the performance of models be tested on left-out data (*e. g.*, depending on the target discovery, left-out groups, subjects, records). If the model f_θ performs well on left-out brain images, we may introspect it and study its classification boundaries. Those boundaries contain information regarding which brain region is selectively recruited by the studied stimuli.

Although learned models may be chosen arbitrarily, the fMRI community widely prefers linear classification/regression models, that seem to provide the best performances. Starting from the seminal work of Cox and Savoy (2003), Support Vector Machines (SVM/SVR, Hearst et al., 1998) have been very popular in the field, as well as Linear Discriminant Analysis (LDA, Fisher, 1938). For instance, Figure 2.4 shows the brain map that discriminates face from cat visual

1. In Chapter 7, we will poll statistical maps from many studies and avoid using complex contrasts, as designing those requires in-depth knowledge of each protocol and hardly scales-up to large repository analysis.

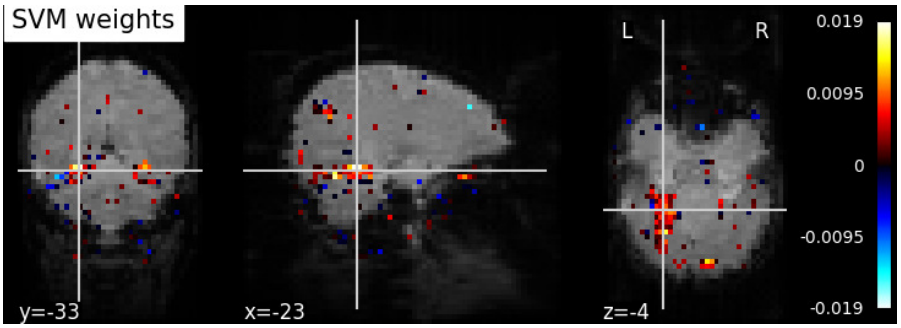


Figure 2.4 – Classification map for decoding face vs cat stimuli in the Haxby et al. (2001b) experiment. Taken from *Nilearn* documentation.

stimuli. It is computed by fitting a linear SVM classifier to z-maps associated with face and cat stimuli. We observe that the Fusiform face area (FFA) region possesses very high coefficients, as expected: it is known to be crucial for face recognition (Kanwisher et al., 1997).

2.3.2.2 *Handling scarse high dimensional data*

Decoding in fMRI is a typical hard learning setting. The available data are relatively scarce: each recorded subject provides as much statistical maps as tested conditions. Even in very large studies like HCP, that costed more than $40 millions, we may only work with $43,000$ statistical maps, that corresponds to 18 base conditions recorded twice on 1200 subjects. Although we can hope to increase the size of the datasets by considering the raw BOLD time-series and increase performance by performing decoding in the time domain (Loula et al., 2017), decoding from statistical maps remains a strong baseline. Morally, we thus own little data with high signal.

On the other hand, the data to decode (*i. e.* classify or use for regression) are very high dimensional, as they live in the whole brain space, of typically $p = 200,000$ voxels. Estimating models, even linear ones, without overfitting requires some care. Several approaches have been used for this, that may be combined together. They may roughly be categorized in four categories.

— *Feature selection*. We may perform decoding from a reduced voxel space $\mathbb{R}^q$. It may found by searching the brain space with a small sliding window (*Searchlight*, Kriegeskorte et al., 2006), or by simple univariate feature selection (Pereira et al., 2009), *e. g.*, using analysis of variance.

— *Region-of-interest selection*. We may restrict the voxel space in which decoding is performed to brain regions which we assume to contain the relevant signal (see *e. g.*, Etzel et al., 2009, for a review). Selection should be performed beforehand if we wish to avoid biasing relevance results. It typically relies on manually defined regions based on anatomical landmarks (Devlin and Poldrack, 2007), or on regions obtained using functional localizers (Saxe et al., 2006).

— *Dimension reduction*. We project the input brain signal onto a lower dimensional space, using parcellation or brain decomposi-

tion obtained beforehand (see Mourão-Miranda et al., 2005, for seminal work in fMRI). Parcellation or base brain vectors may be obtained from anatomical atlases (*e. g.,* Desikan et al., 2006), or through data-driven analysis (S. M. Smith et al., 2009; Yeo et al., 2011), typically using resting-state data as in Section 2.2.

— *Regularization*. We may regularize the decoding linear models to inject priors on the classification maps we wish to obtain. Regularization includes sparsifying penalties — *e. g.,* Lasso (Tibshirani, 1996; Yamashita et al., 2008) or Group-Lasso regression (Ng and Abugharbieh, 2011; Yuan and Lin, 2006) if we wish the sparsity patterns to be shared across maps — and structure inducing penalties such as total variation norms (Michel et al., 2011).

Note that increasing the amount of data available allows to reduce the amount of regularization to be injected, as we will see reviewing Dohmatob et al. (2016) work in Chapter 3. It also allows to work in higher dimension. This is the starting point of Chapter 7, where we endeavor to gather multiple studies to increase the number of statistical maps usable to estimate decoding models.

2.4 CONCLUSION

We have reviewed the three essentials blocks of fMRI analysis from a statistical learning point of views: resting-state analysis with unsupervised methods, encoding methods (a.k.a. standard analysis) for univariate voxel activity analysis, and decoding methods, meant to perform multi-variate pattern recognition in voxel space. In this thesis, we will start working on resting-state data (Chapter 3 and Part II), before moving to the development of new models for task fMRI analysis (Part III). The following Chapter 3 introduces matrix-factorization methods for resting-state analysis, and proposes new approaches to handle large fMRI datasets.

3

DICTIONARY LEARNING FOR FMRI: DATA COMPRESSION, MODEL TRANSPOSITION

In this chapter, we first review how resting-state functional MRI time-series can be analysed through sparse or dense matrix factorization techniques, and the computational issues of existing formulations. We then introduce a new approach based on random projections to handle these computational issues for sparse matrix factorization of medium-scale datasets. This approach was presented in

> Mensch, A., Varoquaux, G., & Thirion, B. (2016b). Compressed online dictionary learning for fast fMRI decomposition. *Proceedings of the IEEE International Symposium on Biomedical Imaging (ISBI)*.

It is easy to implement and reasonably efficient, and was thus promptly proposed in *Nilearn* (Abraham et al., 2014), an end-user fMRI analysis software for neuroscientists. We linger on the validation methods developed in the context of this work, as they outline the hard problem of validation and parameter selection in matrix factorization applications.

Finally, we analyse the practical limitations of the above approach, and show how we can reformulate sparse matrix factorization for fMRI in a more convenient way. This reformulation will be used extensively in the experimental section of Chapter 4.

3.1 MATRIX FACTORIZATION FOR RESTING-STATE FMRI

As presented in Section 2.2, resting-state fMRI data analysis implies, as an initial step, to decompose a set of 3D records $(x_s)_s$ (BOLD time-series sampled in a volumic voxel grid) into a sum (2.2) of spatially located *functional networks* that isolate parts of the brain signal. Functional networks should form a relevant basis for the experimental signals, *i.e.* represent these signals in a low-dimensional space, and explain the time-series variation with low residuals. As such, functional networks have been successfully used for feature extraction before performing statistical learning — decoding, biomarker extraction, etc.

3.1.1 *Model and data*

Mathematically, multi-record resting-state fMRI data constitute a set of matrices $(\mathbf{X}^s)_{s\in[N]}$ in $(\mathbb{R}^{n\times p})^t$, with p voxels per volume[4], n_s temporal samples per record, and N subjects. Rewriting (2.2) with matrix

[4] *Recall that we have flattened the 3D brain images using a binary brain mask.*

product, we seek to decompose subject s records on k base components:

$$\forall s \in [N], \quad \mathbf{X}^s \approx \mathbf{A}^s \mathbf{D}^s \quad \text{with } \mathbf{D}^s \in \mathbb{R}^{k \times p}, \mathbf{A}^s \in \mathbb{R}^{n_s \times k}. \tag{3.1}$$

We are especially interested in $(\mathbf{D}^s)_s$, which contain the functional networks for subject s. Existing decomposition techniques vary in the criterion they optimize, and on the hierarchical model they rely on at the group level. In the following, we will work with the most simple hierarchical model, that consists in performing time concatenation of the records – first proposed by Calhoun et al. (2001) for ICA. Namely, we extract a single set $\mathbf{D} \in \mathbb{R}^{k \times p}$ of functional networks for all subjects, *i.e.* $\mathbf{D}^s = \mathbf{D}$ for all s. We denote $n \triangleq \sum_s n_s$, $\mathbf{A} \in \mathbb{R}^{n \times k}$ and $\mathbf{X} \in \mathbb{R}^{n \times p}$ the vertical concatenation of $(\mathbf{A}^s)_s$ and $(\mathbf{X}^s)_s$, and seek to decompose $\mathbf{X}$ as $\mathbf{X} \approx \mathbf{A}\mathbf{D}$.

3.1.2 *Existing decomposition methods*

While performing Principal component analysis (PCA) on brain images has been the first method to be proposed to extract basis components from fMRI data, Independent Component Analysis (ICA, A. J. Bell and Sejnowski, 1995; Hyvärinen and Oja, 2000) is presently the most popular decomposition technique in the field (Calhoun et al., 2001; McKeown et al., 1998). It involves finding a spatial basis $\mathbf{D} \in \mathbb{R}^{p \times k}$ that is closest to a set of spatially *independent* sources. Works preceding this thesis (Varoquaux et al., 2011) have shown that good results can be obtained imposing *sparsity* rather than *independence* over the spatial components $\mathbf{D}$. They rely on the *dictionary learning* formulation, first introduced by Olshausen and Field, 1997, following seminal work on sparsity (Tibshirani, 1996). Like ICA, dictionary learning permits to extract functional networks that are reproducible across subjects, and that match regions extracted from task fMRI studies. We will focus on this method in the following.

3.1.3 *Computational caveats*

All the aforementioned decomposition techniques suffer from their lack of scalability, as they were initially designed to be applied to small datasets. The recent increase in publicly available dataset size like HCP (Van Essen et al., 2012) has revealed their limits in terms of memory usage and computational time. Consequently, efforts have been made to make decomposition methods available for large-scale studies, possibly with several groups. They involve using a more complex hierarchical model for dictionary learning (Varoquaux et al., 2011) or incremental PCA techniques (S. M. Smith et al., 2014). However, the latter only proposes PCA and ICA based decomposition methods, which do not naturally yield sparse maps, and the former suffers from its computational complexity. As of 2015, running a standard decomposition algorithm with 200 components on the full HCP dataset (4 TB) required more than a week of of computation on a very large workstation.

3.2 DICTIONARY LEARNING FOR RESTING-STATE FMRI

In this chapter, we focus on dictionary learning methods for fMRI, and show how to make them more scalable in both time and memory. Let us start by reviewing the use of dictionary learning in the context of fMRI.

3.2.1 *Sparse brain map extraction with dictionary learning*

A good spatial decomposition of $\mathbf{X} \in \mathbb{R}^{n \times p}$ should allow a good reconstruction of data and the components it contains should be *spatially localized*. To that effect, we seek to find a dictionary $\mathbf{D}$ *sparse* in voxel space, hoping that sparsity translates into maps with a few small connected regions. Such a decomposition setting can be formalized in the Dictionary learning (DL) optimization framework, that combines a sparsity inducing penalty to a reconstruction loss. We seek to find k dense *temporal* atoms $\mathbf{A}$, *i.e.* base time-series, that will constitute loadings for k sparse spatial maps $\mathbf{D}$, with good signal recovery. This leads to the following optimization problem, similar to the original formulation[5] of Olshausen and Field (1997):

$$\min_{\substack{\mathbf{A} \in \mathbb{R}^{n \times k}, \\ \mathbf{D} \in \mathbb{R}^{k \times p}}} \frac{1}{2} \|\mathbf{X} - \mathbf{A}\,\mathbf{D}\|_F^2 + \lambda \|\mathbf{D}\|_1 \quad \text{s.t.} \quad \forall j \in [k], \|\mathbf{a}^{(j)}\|_2 \leqslant 1 \tag{3.2}$$

[5] *that used a smooth version of the ℓ_1 penalty.*

For a given solution $(\mathbf{A}, \mathbf{D})$ of (3.2), every row $\mathbf{d}_j$ of $\mathbf{D}$ contains the loadings related to the j-th temporal atoms $\mathbf{a}^j$. In this case, $\mathbf{D}$ corresponds to a spatial *code* and $\mathbf{A}$ to a temporal *dictionary*[6]. Thanks to the ℓ_1 penalty on $\mathbf{D}$, we expect each row to be *sparse* (Tibshirani, 1996). The constraint on the columns of $\mathbf{A}$ simply prevent $\mathbf{D}$ from becoming arbitrarily small, which would cancel out the ℓ_1 penalty effect.

[6] *The terminology code/dictionary is here transposed compared to the usual computer vision DL problem.*

Mairal et al., 2010 introduce an efficient online algorithm to solve (3.2). In our case, it streams *voxel time-series*, *i. e.* streams columns of $\mathbf{X}$, to progressively learn $\mathbf{A}$. We will thoroughly present and extend this algorithm in Chapter 4. For now, it suffices to know that under conditions satisfied in neuro-imaging, the algorithm finds (asymptotically) a matrix $\mathbf{A}$ that is a stationary point of the following objective, equivalent to (3.2),

$$\min_{\mathbf{A} \in \mathbb{R}^{n \times k}} \|\mathbf{X} - \mathbf{A}\,\mathbf{D}(\mathbf{A})\|_F^2 \quad \text{s.t.} \quad \forall j \in [k], \|\mathbf{a}^{(j)}\|_2 \leqslant 1, \text{ where}$$

$$\mathbf{D}(\mathbf{A}) \triangleq \operatorname*{argmin}_{\mathbf{D} \in \mathbb{R}^{k \times p}} \frac{1}{2p} \|\mathbf{X} - \mathbf{A}\,\mathbf{D}\|_F^2 + \lambda \|\mathbf{D}\|_1, \tag{3.3}$$

although it never materializes the full matrix $\mathbf{D}$, and only solve the Lasso problem (3.3) for small fractions of $\mathbf{X}$ columns, which form *mini-batches* of voxels. The final spatial components that we are interested in are then obtained solving the final Lasso problem

$$\mathbf{D} = \operatorname*{argmin}_{\mathbf{D} \in \mathbb{R}^{k \times p}} \frac{1}{2} \|\mathbf{X} - \mathbf{A}\,\mathbf{D}\|_F^2 + \lambda \|\mathbf{D}\|_1.$$

A good initialization for temporal atoms is required to obtain an exploitable solution as the dictionary learning objective (3.2) is not convex. It can typically be obtained by computing time-series associated

with an initial guess on activation maps $\mathbf{D}_{\text{init}}$, *e.g.*, obtained from known anatomical brain networks. The initial temporal atoms $\mathbf{A}_{\text{init}}$ are then computed by solving $\min_{\mathbf{A}} \|\mathbf{X} - \mathbf{A}\mathbf{D}_{\text{init}}\|_F^2$.

3.2.2 *Scalability challenge*

Following Mairal et al. (2010), online dictionary learning has an overall complexity of $\mathcal{O}(n\,p\,k^2)$, as convergence is typically reached within few epochs on resting-state fMRI data. In theory, the dictionary learning problem therefore scales linearly in the size of the data. However, on large rest fMRI datasets, online dictionary learning faces two main challenges that make wall-clock processing time grow faster than data size.

OUT-OF-CORE REQUIREMENTS. For datasets like HCP ($n = 4 \cdot 10^6$, $p = 2 \cdot 10^5$), typical computers are unable to hold all data in memory. It is thus necessary to stream the data from disk, which is only reasonably efficient if the data are stored in the same direction as they are accessed. Yet online DL algorithms require to pass 3 times over data, during which samples are accessed in different directions (row-wise for initialization, columnwise for DL and final Lasso regression), while fMRI images are naturally stored row-wise in our formalism. For the sake of efficiency, storage copy and manipulation is required, which is a serious issue for neuroscientists dealing with over 1TB datasets. Going out-of-core sets a large performance gap between small datasets and large datasets.

GRID SEARCH. The sparsity of the maps obtained depends critically on parameter λ, that scales non trivially with n. It is therefore impossible to set it independently from the spatial resolution, and several runs must be performed to obtain the most interesting maps, relative to their neurological relevance or a supervised validation criterion. Grid search should be run in parallel for efficiency, which is a serious issue when doing out-of-core computation, as simultaneous access to the disk from different processes makes the pipeline IO-bound. Reducing the dataset size therefore reduces disk and memory usage, which permits the efficient use of more CPUs.

Both issues suggest to reduce memory usage by reducing the size of the dataset while keeping the essential part of its signal. Being able to keep data in memory indeed avoids drastic loss in performance, while reducing the dataset size should also bring a quasi-linear improvement in theoretical complexity. We will first address this dimension reduction idea using random projections.

3.3 TIME-COMPRESSED DICTIONARY LEARNING

It is possible to obtain reasonably exploitable components from a 40 record dataset with a total of 6,000 samples. This drove us to investigate how large datasets, that provide more than 10^5 samples, can

be reduced to fit in memory before performing dictionary learning, with controlled perturbation compared to DL on non-reduced data.

3.3.1 *Reducing the temporal dimension before learning the dictionary*

Resting-state brain images are not uniformly scattered in voxel space, and should exhibit some low dimension structure: we expect them to be scattered close to some low rank subspace of $\mathbb{R}^p$, spanned by a set of m_s vector $\mathbf{X}_r^s \in \mathbb{R}^{m_s \times p}$. We thus perform a *hierarchical* rank reduction: $\mathbf{X}^s$ is first approximated by a rank m_s surrogate matrix $\mathbf{P}_s\mathbf{X}_r^s$, where $\mathbf{P}_s$ is a matrix of $\mathbb{R}^{m_s \times n_s}$. A final rank k decomposition is computed over concatenated data. We show that such reduction preserves enough signal to allow good map extraction. Geometrically, we project $\mathbf{X}$ onto a low rank subset of $\mathbb{R}^{n_s \times p}$, defining

$$\mathbf{P}^s \triangleq \underset{\mathbf{P} \in \mathbb{R}^{m_s \times n_s}}{\operatorname{argmin}} \|\mathbf{X}^s - \mathbf{P}^\top \mathbf{P}\mathbf{X}^s\|_F^2, \quad \mathbf{X}_r^s \triangleq \mathbf{P}^s\mathbf{X}^s. \tag{3.4}$$

We may then write $\mathbf{X}^s = \mathbf{P}^\top \mathbf{X}_r^s + \mathbf{E}^s$, where $\mathbf{E}^s$ is a residual full rank noise matrix. We approximate $\mathbf{X}^s$ with $\mathbf{X}_r^s$ at *subject* level to retain subject variability. We denote $\mathbf{X}_r \in \mathbb{R}^{m \times p}$ the vertical concatenation of $(\mathbf{X}_r^s)_s$, where $m = \sum_s m_s$. Replacing $\mathbf{X}$ by $\mathbf{X}_r$ in (3.2), we obtain the *reduced* dictionary learning objective

$$\min_{\substack{\mathbf{A} \in \mathbb{R}^{m \times k}, \\ \mathbf{D} \in \mathbb{R}^{k \times p}}} \frac{1}{2}\|\mathbf{X}_r - \mathbf{A}\,\mathbf{D}\|_F^2 + \lambda\|\mathbf{D}\|_1 \quad \text{s.t.} \quad \forall j \in [k], \|\mathbf{a}^{(j)}\|_2 \leqslant 1 \tag{3.5}$$

The dimensionality of the solution $\mathbf{A}_r$ is changed compared to (3.2), whereas the size of $\mathbf{D}$ remains the same, as no reduction has been performed in voxel space. Any solution $\mathbf{D}_r$, that we recover solving

$$\mathbf{D}_r = \underset{\mathbf{D} \in \mathbb{R}^{k \times p}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{X}_r - \mathbf{A}_r\,\mathbf{D}\|_F^2 + \lambda\|\mathbf{D}\|_1$$

can therefore still be interpreted as spatial functional networks. Importantly, we must have $m > k$ so that $\mathbf{X}_r$ is at least of rank k to recover k sparse activation maps. On the other hand, we show that using $(\mathbf{X}_r^s)_s$ matrices with $m_s < k$ still provides good results.

When applying the online dictionary learning algorithm to compressed data $\mathbf{X}_r$, time and memory complexity are reduced by a factor $\alpha = \frac{m}{n}$, where m should typically be of the same order than k. This speed-up becomes supra-linear when reduction allows to go from out-of-core to in-core computation. Of course, compression comes with an additional cost, namely the time required for matrix reduction — we analyse this cost further on.

While (3.4) can be seen as another way of decomposing $(\mathbf{X}^s)_s$, let us stress that this decomposition is performed in voxel space, in contrast with dictionary learning itself, that identifies a good basis in *time* space. The objective is to quickly find a good *summary* of each $(\mathbf{X}^s)_s$ prior to applying dictionary learning, so as to reduce the dimensionality of the dictionary learning problem. We summarize the compressed DL approch in Figure 3.1.

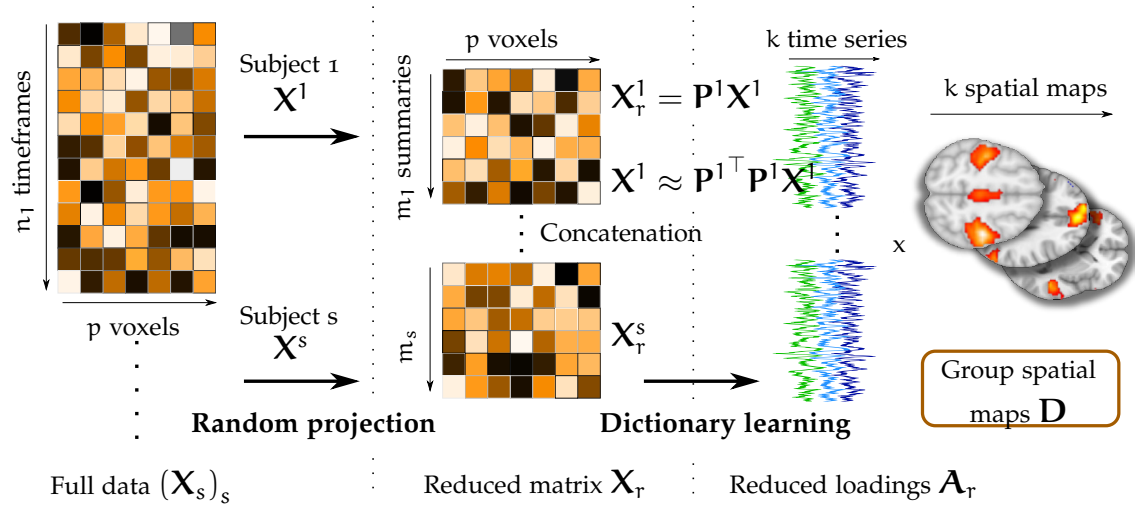


Figure 3.1: Summary of the compressed dictionary learning approach for fast sparse spatial maps extraction.

3.3.2 *Time compression methods*

We compare two approaches for performing temporal dimension reduction:

RANGE-FINDING APPROACH $\mathbf{X}_r^s$ can be computed so as to exactly minimize $\|\mathbf{E}_s\|_F^2$ with truncated Singular value decomposition (SVD), following Eckart-Young-Mirsky theorem (1936). However, exact SVD computation is typically performed in $\mathcal{O}(p\,n^2)$, which is above the complexity of dictionary learning and makes prior data reduction useless when trying to reduce both computation time and memory usage. Fortunately, we show that we do not need to work with the exact best rank-m_s approximation of $\mathbf{X}$ to obtain a satisfying $\mathbf{V}$. Following Halko et al. (2011), we seek $\mathbf{P}^s \in \mathbf{R}^{m_s \times n_s}$ such that

$$\|\mathbf{X}^s - \mathbf{P}^{s\top}\mathbf{P}^s\mathbf{X}^s\|_F \approx \min_{\text{rank}(\mathbf{Y}^s)\leqslant m_s} \left(\|\mathbf{E}^s\|_F^2 = \|\mathbf{X}^s - \mathbf{Y}^s\|_F^2\right)$$

In Alg. 4.4, Halko et al. (2011) propose a fast, randomized algorithm to compute such $\mathbf{P}^s$, with high probability control[7] of the residual error $\|\hat{\mathbf{E}}^s - \mathbf{E}^s\|_F$, where $\hat{\mathbf{E}}_s \triangleq \mathbf{X}^s - \mathbf{P}^{s\top}\mathbf{P}^s\mathbf{X}^s$. We use the output of this algorithm to set $\mathbf{X}_r^s \triangleq \mathbf{P}^s\mathbf{X}^s$, and proceed to solving (3.5)

For a single subject, the randomized range finding *(rf)* algorithm has a $\mathcal{O}(n\,p\,m_s)$ complexity, which is typically much lower than the complexity of dictionary learning, provided $m_s \ll k^2$. Dimension reduction may be performed in parallel for all subjects, or sequentially if few CPUs are available. Respectively, computation gain are expected if $max_s(m_s) \ll k^2$ or $m \ll k^2$, which will be the case for datasets of reasonable size. In practice, we show in Section 3.4.2 that the cost of reduction becomes negligible with respect to the reduction of dictionary learning cost, when the reduction ratio is high enough.

[7] *Due to Johnson and Lindenstrauss (1984) lemma.*

SUBSAMPLING. In a more straightforward way, we can set $\mathbf{X}_r^s$ to contains a subset of $\mathbf{X}^s$ rows, of size m_s. This category of reduction includes time subsampling *(ss)* (*e. g.*, taking 1 frame every 5 seconds)

of resting-state fMRI records. In this case, $\|\hat{\mathbf{E}}^s_{ss} - \mathbf{E}^s\|_F$ cannot be controlled, and is expected to be larger than $\|\hat{\mathbf{E}}^s_{rf} - \mathbf{E}^s\|_F$. Subsampling, for example, is expected to alias high frequency signal in records, preventing the recovery of activation maps with high frequency temporal loadings dictionary learning on reduced data. This approach will serve as a baseline for measuring the benefits of the range-finding approach.

3.4 VALIDATION AND RESULTS OF COMPRESSED DICTIONARY LEARNING

Assessing the validity of compression before dictionary learning is in itself a challenge as it requires to compare sets of spatial maps that are obtained with a *stochastic* algorithm. We design a metric to assess the overlap between two sets of functional networks despite this property. We then demonstrate the efficiency of our compression approach on two resting-state fMRI datasets.

3.4.1 *Validation*

COMPARING SETS OF MAPS. Validation of dictionary learning methods for resting-state fMRI is challenging, as there is no ground truth to assess the quality of resulting maps. However, we can assess how much a set of maps $\mathbf{D}_r$ obtained on a reduced dataset $\mathbf{X}_r$ from (3.5) is comparable to a set of maps $\mathbf{D}$ obtained on $\mathbf{X}$ solving (3.2). The metric used between b two sets of maps $\mathbf{D}$ and $\mathbf{D}_r$ should be invariant to map ordering and map scaling. To enforce this invariance, we find the best one-to-one coupling between maps of $\mathbf{D}$ and $\mathbf{D}_r$ and compute the mean correlation between all *best assigned* couple of maps. Formally,

$$c_{i,j} \triangleq \frac{|(\mathbf{d}_{(i)})^\top \mathbf{d}_{r,(j)}|}{\|\mathbf{d}_{(i)}\|_2 \|\mathbf{d}_{r,(j)}\|_2}, \qquad d(\mathbf{D}, \mathbf{D}_r) \triangleq \max_{\boldsymbol{\Omega} \in \mathcal{S}_k} \operatorname{Tr}(\boldsymbol{\Omega}\mathbf{C}),$$

where S_k is the set of permutation matrices of $\mathbb{R}^{k\times k}$. $\mathbf{C}$ holds the absolute cosine similarities between each pair of maps of $\mathbf{D}$ and $\mathbf{D}_r$, while $d(\mathbf{D}, \mathbf{D}_r)$ is the mean correlation between all best assigned maps. $\boldsymbol{\Omega}$ can be computed efficiently using the Hungarian algorithm (Kuhn, 1956), as k remains reasonably small ($k < 1000$).

HANDLING RANDOM RESULTS. (3.2) and (3.5) admits many local minima that depend on the online DL algorithm initialization, and on the order used for streaming the matrix columns. We index the different runs of DL with an integer ξ that represents the *seed* used for this algorithm. For every compressed/non-compressed matrix $\mathbf{Y}$ obtain from $\mathbf{X}$, we expect the maps $\mathbf{D}(\mathbf{Y}, \xi)$ output by dictionary learning to capture the same neurological/physical phenomena, regardless of the seed ξ. We wish to be capable of measuring the effect of compression on the quality of the output spatial maps, despite the stochastic confounders inherent to the nature of the online DL algorithm. For

this, we perform $l = 10$ different runs of online DL algorithm with different seeds $(\xi_i)_{i\in[l]}$, sampled randomly, as in Himberg et al. (2004). We then write $\mathbf{D}_l(\mathbf{Y}) = [\mathbf{D}(\mathbf{Y}, \xi_1)^\top, \dots, \mathbf{D}(\mathbf{Y}, \xi_l)^\top]^\top$ the vertical concatenation of the output maps using matrix $\mathbf{Y}$ and consider the metric

$$d_l(\mathbf{X}, \mathbf{Y}) \triangleq d\big(\mathbf{D}_l(\mathbf{X}), \mathbf{D}_l(\mathbf{Y})\big). \tag{3.6}$$

As l grows, $d_l(\mathbf{X}, \mathbf{X}_r)$ becomes the ideal metric for measuring the effect of compressing $\mathbf{X}$ into $\mathbf{X}_r$ over the quality of the DL output, as its variance over selecting seeds reduces. In practice, we may yet only work with a finite number of runs. To circumvent this issue, we compare $d_l(\mathbf{X}, \mathbf{X})$ to $d_l(\mathbf{X}, \mathbf{Y})$, where the seeds used to produce the left and right matrices $\mathbf{D}_l$ in (3.6) are sampled differently. The latter metric should be close to the former if the compression does not destroy too much information — in other word, we compare the deviation of solutions due to data compression to their variance due to randomness.

3.4.2 *Results*

We validate our reduction framework over two different datasets with different size: the ADHD dataset, with 40 records, $n = 150$ time steps per record (2 GB); a subset of HCP dataset, using 40 subjects, 2 records per subject, and $n = 400$ (25 GB).

Dictionary learning output depends on its initialization, and the problem of choosing the *best* number of components k is very ill-posed. We bypass these problems by choosing $k = 70$ for HCP, $k = 20$ for ADHD dataset, and use reference ICA-based maps RSN20 and RSN70 from S. M. Smith et al. (2009) for initialization.

For benchmarking, we measure CPU time only, *i. e.* ignore IO timings as they are very platform dependent. Note that our method is also beneficial for IO as it has a lower memory footprint than full-scale dictionary learning and may thus avoid out-of-core computation. We use *scikit-learn* for computation, along with the *Nilearn* neuro-imaging library. Code for the methods and experiments is available online.

METRIC VALIDITY. We perform the following experiment. We first choose λ to obtain little overlapping maps when running dictionary learning on the non-compressed matrix $\mathbf{X}$. Then, we compute $d_l(\mathbf{Y}, \mathbf{X})$ with $\mathbf{Y} = \{\mathbf{X}, (\mathbf{X}_r)_{rf,m}, (\mathbf{X}_r)_{ss,m}\}$, for various $m \in [n/40, n]$. As the relationship between λ and a given level of sparsity depends on m, we run DL on $\mathbf{Y}$ on a range of λ so as to find the value that matches best the reference run.

Figure 3.2 shows the behavior of $d_l(\mathbf{X}, \mathbf{Y})$ as l increases. Observing $d_l(\mathbf{X}, \mathbf{X})$ (blue curve), we see that running DL several times does produce sets of maps that overlap more and more" $d_l(\mathbf{X}, \mathbf{X})$ increases and its variance across seeds decrease. This suggest that our indicator does cater for randomness in DL algorithms and is a principled approach for comparing compressed and uncompressed dictionary learning.

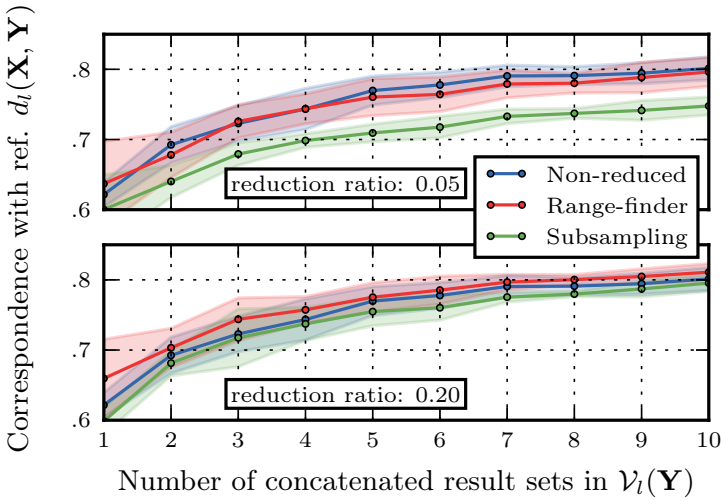


Figure 3.2 – Correspondence between compressed DL map-set and non-compressed DL map-set. We use different compression methods and ratios and increase the number of runs to show d_l stabilization; variance over runs computed using 4 different seed sequences; ADHD dataset.

QUANTITATIVE PERFORMANCE. For $\alpha > .025$, and $l \geqslant 2$, Figure 3.2 shows that the distance between DL maps and compressed DL maps is comparable, as $d_l(\mathbf{X}, \mathbf{X}_r) \approx d_l(\mathbf{X}, \mathbf{X})$. Maps from $\mathbf{D}_l(\mathbf{X})$ and $\mathbf{D}_l(\mathbf{X}_r)$, respectively obtained with or without compression, have therefore the same quality for neuroscientists — it is not possible to tease them apart more than we can tease apart the results of DL obtained with different seeds. For large compression factors, typically $\alpha < .1$ on ADHD and $\alpha < .05$ on HCP, range-finding reduction performs significantly better than subsampling. Both methods perform similarly for small compression factors, which shows that subsampling already provides good low-rank approximation of $\mathbf{X}$ for large m. Using the range-finding algorithm for the proposed hierarchical compression model is therefore useful when drastically reducing data size, typically when loading very large datasets in memory.

QUALITATIVE ACCURACY. We validate qualitatively our results, as this is crucial in DL decomposition: maps obtained from reduced data should capture the same underlying neurological networks as reference maps. In Fig. 3.3, we display matched maps when comparing two map-sets. For this, we find matchings between the maps $\mathbf{D}_l(\mathbf{X})$ and $\mathbf{D}_l(\mathbf{X}_r)$, and we display the maps corresponding to the median-value of this matching. Maps are strongly alike from a neurological perspective. In particular, maps do not differ more between our reduced dictionary learning approach and the reference algorithm than across two runs of the reference algorithm with different seeds.

TIME AND QUALITY TRADEOFF. For efficient neuroimaging data analysis, the important quantity is the tradeoff between quality of the results and computation time. On Figure 3.4, we plot $d(\mathbf{D}_l(\mathbf{X}), \mathbf{D}_l(\mathbf{Y}))$

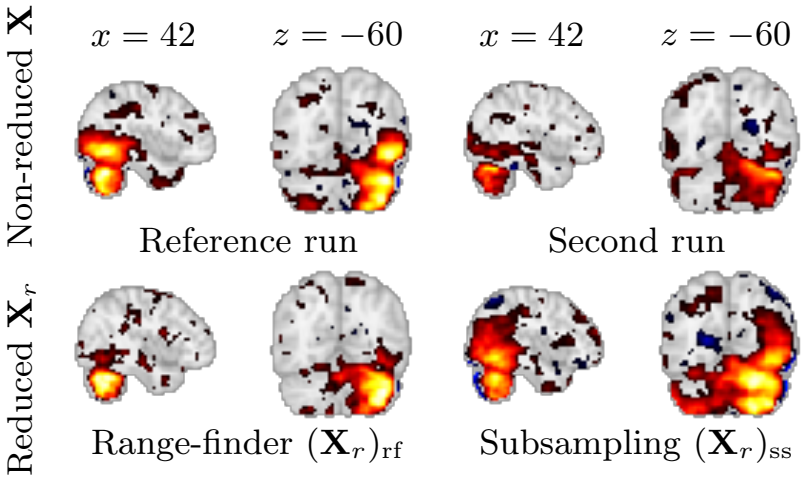


Figure 3.3 – Median-aligned maps with various compression methods. Subset of the HCP dataset; reduction ratio $\alpha = .025$.

against computational CPU time, for various $\mathbf{Y}$. Using range-finding algorithm and to a lesser extent time subsampling on data before map decomposition does not significantly deteriorate results up to large reduction factor, while allowing large gains in time *and* memory. Compression can be higher for larger datasets: we can compress the HCP-derived dataset up to 40 times, and the ADHD dataset up to 20 times, keeping $d_l(\mathbf{X}, \mathbf{X}_r)$ within the standard deviation of $d_l(\mathbf{X}, \mathbf{X})$.

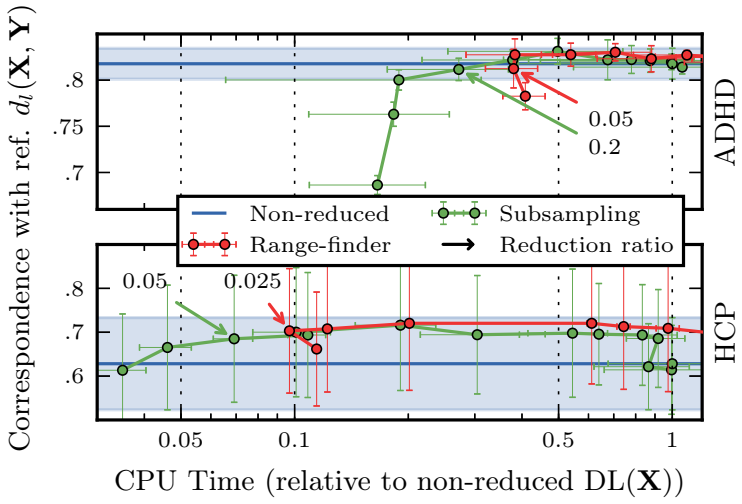


Figure 3.4 – Time/quality trade-off using range-finder projectors and subsampling before DL; blue stripe recalls correspondence of results when performing different runs of DL on the full matrix $\mathbf{X}$. $l = 10, 3$ for ADHD, HCP; variance over runs computed using 4 distinct sequences of seeds.

The range-finder algorithm adds a time overhead that shift the performance curve towards the right for large compression ratios. However, it allows 4 times lower memory usage than subsampling for equivalent quality and time budget. It thus provides a higher overall

		CPU Time		Overlap $d_l(\mathbf{X}, \mathbf{Y})$	
Dataset	Ratio α	Reduced	Non-red.	Red $\mathbf{Y} = \mathbf{X}_r$	Non-red $\mathbf{Y}$=$\mathbf{X}$
HCP	.025	**849 s**	7425 s	.703±.141	.628±.105
ADHD	.05	**71 s**	186 s	.796±.020	.801±.016

Table 3.1 – Time/quality trade-off of compressed dictionary learning, compared to reference dictionary learning, for good trade-off compression ratios α.

efficiency when considering IO timings. Note that benchmarks were performed on a single core, while reduction can be parallellized over subjects to reduce its overhead — range-finding should therefore always be preferred to trivial subsampling.

We outline good time/quality trade-off reduction ratios in Fig. 3.4 and provide numerical values in Table 3.1. Those ratios depend on the number of functional networks k and on the input dataset, but any reasonably low reduction ratio (typically, setting $m_s \gtrsim k$) is likely to produce good results with little quality loss. Following this strategy, we set $\alpha = .025$ and performed the entire processing of a subset of 100 subjects of the HCP dataset (384 GB) on a single workstation (with 64 GB RAM) in less than 7 hours, obtaining usable functional networks. Note that the following Chapters 6 and 7 instead consider full HCP releases (with 500, then 900 subjects), which do not fit in memory even with drastic compression.

3.4.3 *Discussion*

In the previous sections, we introduced the use of a randomized range finding algorithm to reduce large scale datasets before performing dictionary learning and extracting spatial maps. To prove efficiency of time reduction before dictionary learning, we have designed a meaningful indicator to measure result maps *correspondence*, and have demonstrated that fMRI time samples have a low rank structure that allows range finding projection to be more efficient than simple subsampling. This approach enables a 40-fold data reduction upon loading each record.

Unfortunately, this approach has strong limitations when it comes to handling datasets with thousands of subjects. First, as fMRI datasets are acquired and thus stored *time-wise*, *i.e.* row-wise in our formalism, the compressed matrix $\mathbf{X}_r$ ultimately needs to be materialized in memory (an out-of-core approach would require to transpose $\mathbf{X}_r$ on disk, an operation that would become the bottleneck of the entire pipeline). This is only possible if $\mathbf{X}_r$ remains of reasonable size. For 4TB datasets like HCP, even with a compression factor of 40, this requires 100GB of memory per compressed DL process, and is therefore beyond the capacity of regular machines.

Secondly, the temporal dictionary learning approach suffers from its three-phase nature: initialization from known *spatial components*, learning of *temporal atoms*, sparse regression into *spatial components*. We would hope to be able to learn spatial components $\mathbf{D}$ in a single pass over data.

Finally, the idea of using temporal time-series as *samples* for online dictionary has two major drawbacks. First, it makes the selection of the sparsity parameter λ dependent on the dimensionality of samples n. This implies that a good λ depends on the number and length of records in a study. As a consequence, λ must be recomputed using grid-search for new studies, or if the study size increases. It would be much more convenient if the λ parameter depended on the spatial resolution of the scanner. Furthermore, the present approach does not permit to refine a trained dictionaries using new records from new subjects. Building on this discussion, we now propose a reformulation of the data decomposition problem that will prove much more convenient.

3.5 CHANGING MODEL AND GOING BEYOND

The crux of the limitations above lies in the fact that the online dictionary learning algorithm focuses on learning the left-side factor of the matrix factorization $\mathbf{X} = \mathbf{AD}$. In our formalism, the left-side factor $\mathbf{A}$ corresponds to *temporal components*, whereas we are actually interested in recovering *spatial components*. This forces us to 1) stream artificial *temporal samples*, whereas the data are presented in the form of *spatial samples* and 2) to compute the right side factor at the end of the dictionary learning loop. We therefore propose to transpose the matrix factorization problem (3.1), so as to learn directly the spatial components on the left-side factor. The dictionary learning problem requires to be adapted to enforce sparsity on this factor. We will see that the new problem offers further flexibility for enforcing structure over spatial maps. More importantly, it will allow the use of a much faster algorithm presented in Chapter 4.

3.5.1 *Transposed problem*

We may choose to stack brain images acquired during resting-state protocols columnwise instead of columnwise, and obtain matrices $\mathbf{X}^s \in \mathbb{R}^{p \times n}$, where p is the number of acquired voxels and n the number of time samples. Similar to (3.1), we wish to find a dictionary $\mathbf{D}^s \in \mathbb{R}^{p \times k}$ of sparse spatial components (functional networks) and a code $\mathbf{A}^s \in \mathbb{R}^{k \times n}$ of temporal loadings:

$$\mathbf{X}^s \approx \mathbf{D}\mathbf{A}^s \quad \text{at subject level,} \qquad \mathbf{X} \approx \mathbf{D}\boldsymbol{A} \quad \text{at group level.} \tag{3.7}$$

Like before, we assume that all records are decomposed on the same spatial dictionary $\mathbf{D}$, and $\mathbf{X}$ and $\boldsymbol{A}$ are formed of horizontal concatenation of $(\mathbf{X}^s)_s$ and $(\mathbf{A}^s)_s$. Note that, compared to (3.1), the spatial components $\mathbf{D}$ are now found on the *left*-side factor in (3.7).

This is by design: the online dictionary learning algorithm is precisely designed to find such left-side factor by streaming the columns of $\mathbf{X}$. We have to adapt the penalties and constraints used for our purpose: we want to find a sparse dictionary $\mathbf{D}$, and a dense code $\mathbf{A}$. Inspired by Mairal et al. (2010), we consider the following objective function

$$\min_{\substack{\mathbf{D}\in\mathbb{R}^{p\times k},\\ \mathbf{A}\in\mathbb{R}^{k\times n}}} \frac{1}{2}\|\mathbf{X}-\mathbf{D}\,\mathbf{A}\|_F^2+\lambda\|\mathbf{A}\|_2^2 \quad \text{s.t.} \quad \forall j\in[k], \|\mathbf{d}^{(j)}\|_1 \leqslant 1. \tag{3.8}$$

We have replaced the ℓ_2 ball constraints on the atoms of the dictionary by ℓ_1 ball constraints. Such constraint has a similar sparsifying effect as the ℓ_1-penalty had on previous DL formulation. For $\mathbf{A}$ fixed, minimizing (3.8) over $\mathbf{D}$ amounts to minimizing a quadratic function over a polytope with vertices: the solution $\mathbf{D}$ tends to be located at the edges of this polytope, which corresponds to sparse $(\mathbf{d}^j)_j$. The ℓ_2 penalty plays a similar role as the ℓ_2 ball constraint in (3.2), and ensures that the constraints over $\mathbf{D}$ are well saturated. The sparsity of solutions $\mathbf{D}$ increases with λ. This setting bears some similarity with the original Sparse principal component analysis (SPCA) formulation from Zou et al. (2006). Compared to their formulation, no orthogonality constraint is enforced on $\mathbf{D}$. This is better suited to our purpose we typically want to capture components that explain different but *comparable* aspect of the variance of brain images.

Online dictionary learning constructs a sequence $(\mathbf{D}_t)_t$ that converges toward a critical point of the objective function

$$\min_{\mathbf{D}\in\mathbb{R}^{p\times k}} \|\mathbf{X}-\mathbf{D}\,\mathbf{A}(\mathbf{D})\|_F^2 \quad \text{s.t.} \quad \forall j\in[k], \|\mathbf{d}^{(j)}\|_1 \leqslant 1, \text{ where}$$
$$\mathbf{A}(\mathbf{D}) \triangleq \operatorname*{argmin}_{\mathbf{A}\in\mathbb{R}^{k\times n}} \frac{1}{2}\|\mathbf{X}-\mathbf{D}\,\mathbf{A}\|_F^2+\lambda\|\mathbf{A}\|_2^2 \tag{3.9}$$

by streaming mini-batches of b samples $\mathbf{x}_t\in\mathbb{R}^{p\times b}$, and solving small ridge regression problems

$$\boldsymbol{\alpha}_t = \operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^{k\times b}} \frac{1}{2}\|\mathbf{x}_t-\mathbf{D}_{t-1}\,\boldsymbol{\alpha}\|_F^2+\lambda\|\boldsymbol{\alpha}\|_2^2$$

at each iteration. Asymptotically, we obtain a dictionary $\mathbf{D}$ that solves (3.8). Through this reformulation, we tackle the three limitations of the original DL formulation discussed above:

— It allows to learn $\mathbf{D}$ in one phase only, starting from a known dictionary $\mathbf{D}_{\text{init}}$ and streaming columns of $\mathbf{X}$.

— It requires to stream brain images, which is how fMRI data are acquired and stored — typically, we may retrain a dictionary with new data easily.

— Its regularization parameter λ should be adapted to p but not to n, which is less cumbersome when adapting pipelines to new studies with different number of samples.

To learn the spatial maps $\mathbf{D}\in\mathbb{R}^{p\times k}$, we are yet confronted to the same high dimensionality problem that existed when learning

$\mathbf{A} \in \mathbb{R}^{n \times k}$ in the original problem: p is typically of the order of 10^5, while n ranges from 10^4 to 10^6 in Section 3.3 and 3.4. Tackling the computational cost created by the high dimensionality of data will be the full purpose of Chapter 4. The approach will differ radically from Section 3.4 as we are now interested in the *left-side* term of $\mathbf{X}$ factorization, and can perform *online* compression.

Before closing this chapter, we explore a first extension that (3.8) reformulation permits: enforcing structured penalties over the spatial dictionary $\mathbf{D}$ by slightly modifying the online dictionary learning algorithm of Mairal et al. (2010).

3.5.2 *Complex spatial regularization for brain maps*

The current constraint set $\mathcal{C} = \{\mathbf{D} \in \mathbb{R}^{p \times k}, \|\mathbf{d}^{(j)}\|_1 \leqslant 1 \, \forall j \in [k]\}$ used in (3.8) over $\mathbf{D}$ enforces sparsity over the dictionary $\mathbf{D}$.

In neuro-imaging, we may wish to obtain function networks whose regularity goes beyond simple sparsity. Interpretable and efficient functional networks should 1) have only a few non-zero voxels 2) these voxels should form well defined, small connex components. This may improve the interpretability of functional networks in the light of known anatomical structure. For this reason, Dohmatob, M., et al. (2016) proposed to augment the objective (3.9) with a further penalty on the dictionary, and to solve

$$\begin{aligned} &\min_{\mathbf{D} \in \mathbb{R}^{p \times k}} \|\mathbf{X} - \mathbf{D}\,\mathbf{A}(\mathbf{D})\|_F^2 + \gamma \sum_{j=1}^{k} \|\nabla \mathbf{d}^{(j)}\|_F^2 \\ &\quad \text{s.t.} \quad \forall j \in [k], \|\mathbf{d}^{(j)}\|_1 \leqslant 1, \text{ where} \\ &\mathbf{A}(\mathbf{D}) \triangleq \operatorname*{argmin}_{\mathbf{A} \in \mathbb{R}^{k \times n}} \frac{1}{2}\|\mathbf{X} - \mathbf{D}\,\mathbf{A}\|_F^2 + \lambda \|\mathbf{A}\|_2^2 \end{aligned} \tag{3.10}$$

where $\|\nabla \mathbf{d}^{(j)}\|_F^2$ is known as a *GraphNet* penalty in neuro-imaging (Grosenick et al., 2013). It penalizes the spatial variations of the dictionary component $\mathbf{d}^{(j)} \in \mathbb{R}^p$, viewed as a 3 dimensional image. By pushing the dictionary components to have little spatial variation, we force the learned brain components to be more focal and to exhibits well localized connected components (*i. e.* spatial "blobs").

In the formulation above, we view the spatial discrete derivative operators $\partial_{\{x,y,z\}} : \mathbb{R}^p \to \mathbb{R}^p$, and the spatial discrete gradient operator $\nabla : \mathbb{R}^p \to \mathbb{R}^{p \times 3} \triangleq [\partial_x, \partial_y, \partial_y]$ as linear operators on the space of 3D brain images. The new introduced penalty is smooth, and requires minimal modifications of the online matrix factorization algorithm to run. We simply need to evaluate the its gradient for each component j: a classical derivation shows that it is $2\gamma\Delta\mathbf{d}^{(j)}$, where the linear operator $\Delta : \mathbb{R}^{p \times p}$ computes the discrete Laplacian of $\mathbf{d}^{(j)}$. This gradient is computable in a time proportional to the number of voxels; introducing the GraphNet penalty in the objective (3.10) only slows down online matrix factorization iterations by a constant factor 3.

Introducing the supplementary GraphNet penalty in the problem (3.10) improves the quality of final brain components when learning

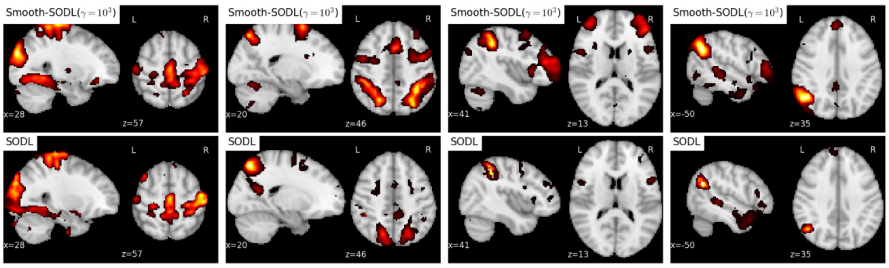


Figure 3.5 – Introducing a GraphNet penalty over the spatial dictionary in the matrix factorization objective improves the quality of output sparse spatial maps (top), compared to simply relying on the ℓ_1 constraints of (3.9) (bottom). Extracted from (Dohmatob et al., 2016), courtesy of its first author.

from small datasets, as illustrated in Figure 3.5. On the other hand, it introduces a further hyperparameter γ that needs to be tuned. The interest of the method decreases as we move to larger datasets such as HCP, as the sparse maps that we learn from these datasets are naturally focal and well localized, without tailored modification of the objective (3.9) beyond the ℓ_1 ball constraints.

Part II

HUGE MATRIX FACTORIZATION

4

STOCHASTIC SUBSAMPLING FOR HUGE MATRIX FACTORIZATION

Part II is the most mathematically heavy of this thesis. We depart from fMRI analysis to dig deeper into the core of matrix factorization algorithms.

4.1 OVERVIEW OF PART II

As discussed in Chapter 3, existing approaches for matrix factorization are not readily usable to extract base sparse or sparsifying components from terabyte scale datasets. We therefore designed a matrix-factorization algorithm, called *subsampled online matrix factorization* (SOMF), that scales to input matrices with both huge number of rows and columns.

SOMF can learn factors sparse or dense and/or non-negative, which makes it suitable for dictionary learning, sparse component analysis, and non-negative matrix factorization. In brief, SOMF streams matrix columns while subsampling them to iteratively learn the matrix factors. At each iteration, the row dimension of a new sample is reduced by subsampling, resulting in lower time complexity compared to a simple streaming algorithm. We present SOMF in detail in this chapter. Beforehand, we provide context on matrix factorization in machine learning and signal processing. We review some algorithms that were proposed, and detail online matrix factorization (OMF), on which SOMF is based.

Our method comes with convergence guarantees to reach a stationary point of the matrix-factorization problem. The convergence analysis is based on analyzing the robustness to perturbation of a wider category of algorithm, known as stochastic majorization-minimization algorithms (Mairal, 2013b). We present the generalized *stochastic approximate majorization-minimization* framework and the convergence analysis of SOMF in Chapter 5.

In Chapter 6, we demonstrate the efficiency of SOMF on massive functional Magnetic Resonance Imaging data (2 TB of resting-state data, that corresponds to the release of 500 subjects of the HCP study), and on patches extracted from hyperspectral images (103 GB). For both problems, in which we use different penalties on rows and columns, we obtain significant speed-ups compared to state-of-the-art algorithms. Finally, we present a adaptation of our algorithm to *explicit collaborative filtering*, which provides large speed-ups compared to the fastest methods available for models relying on matrix factorization.

This part is based on a line of two publications. In

> Mensch, A., Mairal, J., Thirion, B., & Varoquaux, G. (2016a). Dictionary learning for massive matrix factorization. *Proceedings of the International Conference on Machine Learning (ICML)*,

we provide a first version of the SOMF algorithm, with a large empirical study of its performance on resting-state fMRI data and on collaborative filtering. In an extended journal version

> Mensch, A., Mairal, J., Thirion, B., & Varoquaux, G. (2018b). Stochastic Subsampling for factorizing huge matrices. *IEEE Transactions on Signal Processing*, *66*(1), 113–128,

we modify the SOMF algorithm to be able to establish some asymptotic convergence guarantees. We provide a full analysis of SOMF asymptotic behavior, based on a larger theoretical framework. We validate our method on hyperspectral images, and on fMRI data.

4.2 BACKGROUND AND PROPOSED APPROACH

4.2.1 *Matrix factorization in machine learning*

Matrix factorization is a flexible approach to uncover latent factors in low-rank or sparse models. With sparse factors, it is used in dictionary learning, and has proven very effective for denoising and visual feature encoding in signal and computer vision (see e.g., Mairal et al., 2014). When the data admit a low-rank structure, matrix factorization has proven very powerful for various tasks such as matrix completion (Candès and Recht, 2009; Srebro et al., 2004), word embedding (Levy and Goldberg, 2014; Pennington et al., 2014), or network models (Y. Zhang et al., 2009). It is flexible enough to accommodate a large set of constraints and regularizations, and has gained significant attention in scientific domains where interpretability is a key aspect, such as genetics (H. Kim and Park, 2007) and of course neuro-imaging, as we discussed in Chapter 3. In this chapter, our goal is to adapt matrix-factorization techniques to huge-dimensional datasets, i.e., with large number of columns n and large number of rows p. Specifically, our work is motivated by the rapid increase in sensor resolution, for example in hyperspectral imaging or fMRI, and the challenge that the resulting high-dimensional signals pose to current algorithms.

As a widely-used model, the literature on matrix factorization is very rich and two main classes of formulations have emerged. The first one addresses a convex-optimization problem with a penalty promoting low-rank structures, such as the trace or max norms (Srebro et al., 2004). This formulation has strong theoretical guarantees (Candès and Recht, 2009), but lacks scalability for huge datasets or sparse factors. For these reasons, we focus on a second type of approach, which relies on non-convex optimization. Stochastic (or online) optimization methods have been developed in this setting. Unlike classical alternate minimization procedures, they learn matrix decomposi-

tions by observing a single matrix column (or row) at each iteration. In other words, they stream data along one matrix dimension. Their cost per iteration is significantly reduced, leading to faster convergence in various practical contexts. More precisely, two approaches have been particularly successful: stochastic gradient descent (Bottou, 2010) and stochastic majorization-minimization methods (Mairal, 2013b; Razaviyayn et al., 2013). The former has been widely used for matrix completion (see R. M. Bell and Koren, 2007; Burer and Monteiro, 2004; Recht and Ré, 2013, and references therein), while the latter has been used for dictionary learning with sparse and/or structured regularization (Mairal et al., 2010). Despite those efforts, stochastic algorithms are currently unable to deal efficiently with matrices that are large in both dimensions.

4.2.2 *Proposed approach: leveraging stochastic optimization and randomization*

We propose a new matrix-factorization algorithm that can handle such matrices. It builds upon the stochastic majorization minimization framework of Mairal (2013b), which we generalize for our problem. In this framework, the objective function is minimized by iteratively improving an upper-bound surrogate of the function (*majorization* step) and minimizing it to obtain new estimates (*minimization* step). The core idea of our algorithm is to approximate these steps to perform them faster. We carefully introduce and control approximations, so to extend convergence results of Mairal (2013b) when neither the majorization nor the minimization step is performed exactly.

For this purpose, we borrow ideas from *randomized* methods in machine learning and signal processing. Indeed, quite orthogonally to stochastic optimization, efficient approaches to tackle the growth of dataset dimension have exploited random projections (Bingham and Mannila, 2001; Johnson and Lindenstrauss, 1984) or sampling, reducing data dimension while preserving signal content. Large-scale datasets often have an intrinsic dimension which is significantly smaller than their ambient dimension. Good examples are biological datasets (McKeown et al., 1998) and physical acquisitions with an underlying sparse structure enabling compressed sensing (Candès and Tao, 2006). In this context, models can be learned using only random data summaries, also called sketches. For instance, randomized methods (see Halko et al., 2011, for a review) are efficient to compute PCA (Rokhlin et al., 2009), a classic matrix-factorization approach, and to solve constrained or penalized least-square problems (Lu et al., 2013; Sarlos, 2006). On a theoretical level, recent works on *sketching* (Pilanci and M. Wainwright, 2015; Raskutti and Mahoney, 2015) have provided bounds on the risk of using random summaries in learning.

Using random projections as a pre-processing step is not appealing in our applicative context since factors learned on reduced data are not interpretable. On the other hand, it is possible to exploit *random sampling* to approximate the steps of online matrix factorization.

Factors are learned in the original space whereas the dimension of each iteration is reduced together with the computational cost per iteration.

Notation

We recall that matrices are written using bold capital letters and vectors using bold small letters (e.g., $\mathbf{X}$, $\boldsymbol{\alpha}$). We use superscript to specify the column (sample or component) number, and write $\mathbf{X} = [\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}]$. We use subscripts to specify the *iteration* number, as in $\boldsymbol{x}_t$. The floating bar, as in $\bar{g}_t$, is used to stress that a given value is an average over iterations, or an expectation. The superscript * is used to denote an exact value, when it has to be compared to an inexact value, e.g., to compare $\boldsymbol{\alpha}_t^\star$ (exact) to $\boldsymbol{\alpha}_t$ (approximation).

4.3 PRIOR ART: ONLINE MATRIX FACTORIZATION

We first clarify the matrix factorization problem of interest, that generalizes the one we came across in Chapter 3. We recall a specific stochastic algorithm to solve it observing one column (or a mini-batch) at every iteration. In Chapter 5, we cast this algorithm in the stochastic majorization-minimization framework (Mairal, 2013b), which we will use in the convergence analysis.

4.3.1 *Problem statement*

In our setting, the goal of matrix factorization is to decompose a matrix $\mathbf{X} \in \mathbb{R}^{p \times n}$ — typically n signals of dimension p — as a product of two smaller matrices:

$$\mathbf{X} \approx \mathbf{D}\mathbf{A} \quad \text{with} \quad \mathbf{D} \in \mathbb{R}^{p \times k} \text{ and } \mathbf{A} \in \mathbb{R}^{k \times n},$$

with potential sparsity or structure requirements on $\mathbf{D}$ and $\mathbf{A}$. In signal processing, sparsity is often enforced on the code $\mathbf{A}$. As previously discussed in Chapter 3, this problem is known as *dictionary learning* (Olshausen and Field, 1997). In such a case, the matrix $\mathbf{D}$ is called the "dictionary" and $\mathbf{A}$ the sparse code. We use this terminology throughout this work.

Generalizing (3.8), learning the factorization is typically performed by minimizing a quadratic data-fitting term, with constraints and/or penalties over the code and the dictionary:

$$\min_{\substack{\mathbf{D} \in \mathcal{C} \\ \mathbf{A} \in \mathbb{R}^{k \times n}}} \quad \sum_{i=1}^{n} \frac{1}{2} \left\| \boldsymbol{x}^{(i)} - \mathbf{D}\boldsymbol{\alpha}^{(i)} \right\|_2^2 + \lambda \, \Omega(\boldsymbol{\alpha}^{(i)}), \tag{4.1}$$

where $\mathbf{A} \triangleq [\boldsymbol{\alpha}^{(1)}, \ldots, \boldsymbol{\alpha}^{(n)}]$, $\mathcal{C}$ is a column-wise separable convex set of $\mathbb{R}^{p \times k}$ and $\Omega : \mathbb{R}^p \to \mathbb{R}$ is a penalty over the code. Both constraint set and penalty may enforce structure or sparsity, though $\mathcal{C}$ has traditionally been used as a technical requirement to ensure that the penalty on $\mathbf{A}$ does not vanish with $\mathbf{D}$ growing arbitrarily large.

Two choices of $\mathcal{C}$ and Ω are of particular interest. The problem of dictionary learning sets $\mathcal{C}$ as the ℓ_2 ball for each atom and Ω to be the ℓ_1 norm. Due to the sparsifying effect of ℓ_1 penalty (Tibshirani, 1996), the dataset admits a *sparse* representation in the dictionary. On the opposite, finding a *sparse set* in which to represent a given dataset, with a goal akin to sparse PCA (Zou et al., 2006), requires to set as the ℓ_1 ball for each atom and Ω to be the ℓ_2 norm. Our work considers the *elastic-net* constraints and penalties (Zou and Hastie, 2005), which encompass both special cases. Fixing ν and μ in $[0, 1]$, we denote by $\Omega(\cdot)$ and $\|\cdot\|$ the elastic-net penalty in $\mathbb{R}^p$ and $\mathbb{R}^k$:

$$\Omega(\boldsymbol{\alpha}) \triangleq (1-\nu)\|\boldsymbol{\alpha}\|_1 + \frac{\nu}{2}\|\boldsymbol{\alpha}\|_2^2, \tag{4.2}$$
$$\mathcal{C} \triangleq \left\{ \mathbf{D} \in \mathbb{R}^{p\times k} / \|\mathbf{d}^{(j)}\| \triangleq (1-\mu)\|\mathbf{d}^{(j)}\|_1 + \frac{\mu}{2}\|\mathbf{d}^{(j)}\|_2^2 \leqslant 1 \right\}.$$

Following Mairal et al. (2010), we can also enforce the positivity of $\mathbf{D}$ and/or $\boldsymbol{A}$ by replacing $\mathbb{R}$ by $\mathbb{R}^+$ in $\mathcal{C}$, and adding positivity constraints on $\boldsymbol{A}$ in (4.1), as in non-negative sparse coding (Hoyer, 2004). We rewrite (4.1) as an empirical risk minimization problem depending on the dictionary only. The matrix $\mathbf{D}$ solution of (4.1) is indeed obtained by minimizing the empirical risk $\bar{f}$

$$\mathbf{D} \in \operatorname*{argmin}_{\mathbf{D}\in\mathcal{C}} \left(\bar{f}(\mathbf{D}) \triangleq \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{D}, \boldsymbol{x}^{(i)}) \right), \tag{4.3}$$
$$\text{where} \quad f(\mathbf{D}, \boldsymbol{x}) \triangleq \min_{\boldsymbol{\alpha}\in\mathbb{R}^k} \frac{1}{2}\left\|\boldsymbol{x} - \mathbf{D}\boldsymbol{\alpha}\right\|_2^2 + \lambda\,\Omega(\boldsymbol{\alpha}),$$

and the matrix $\boldsymbol{A}$ is obtained by solving the linear regression

$$\min_{\boldsymbol{A}\in\mathbb{R}^{k\times n}} \sum_{i=1}^{n} \frac{1}{2}\left\|\boldsymbol{x}^{(i)} - \mathbf{D}\boldsymbol{\alpha}^{(i)}\right\|_2^2 + \lambda\,\Omega(\boldsymbol{\alpha}^{(i)}).$$

The problem (4.1) is non-convex in the parameters $(\mathbf{D}, \boldsymbol{A})$, and hence (4.3) is not convex. However, the problem (4.1) is convex in both $\mathbf{D}$ and $\boldsymbol{A}$ when fixing one variable and optimizing with respect to the other. As such, it is naturally solved by alternate minimization over $\mathbf{D}$ and $\boldsymbol{A}$, which asymptotically provides a stationary point of (4.3). Yet, $\boldsymbol{X}$ has typically to be observed hundred of times before obtaining a good dictionary. Alternate minimization is therefore not adapted to datasets with many samples.

4.3.2 *Online matrix factorization*

When $\boldsymbol{X}$ has a large number of columns but a limited number of rows, the stochastic optimization method of Mairal et al. (2010) outputs a good dictionary much more rapidly than alternated minimization. In this setting (see Bottou et al., 2018, for a review), learning the dictionary is naturally formalized as an expected risk minimization

$$\min_{\mathbf{D}\in\mathcal{C}} \quad \bar{f}(\mathbf{D}) \triangleq \mathbb{E}_{\boldsymbol{x}}[f(\mathbf{D}, \boldsymbol{x})], \tag{4.4}$$

Algorithm 1 Online matrix factorization (Mairal et al., 2010, OMF)

Input: Initial iterate $\mathbf{D}_0$, sample stream $(\mathbf{x}_t)_{t>0}$, number of iterations T.

for t from 1 to T **do**

Draw $\mathbf{x}_t \sim \mathcal{P}$.

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

Update the *parameters* of aggregated surrogate $\bar{g}_t$:

$$\begin{aligned}\bar{\mathbf{C}}_t &= \Big(1-\frac{1}{t}\Big)\bar{\mathbf{C}}_{t-1} + \frac{1}{t}\boldsymbol{\alpha}_t\boldsymbol{\alpha}_t^\top.\\ \bar{\mathbf{B}}_t &= \Big(1-\frac{1}{t}\Big)\bar{\mathbf{B}}_{t-1} + \frac{1}{t}\mathbf{x}_t\boldsymbol{\alpha}_t^\top.\end{aligned} \tag{4.7}$$

Compute (using block coordinate descent):

$$\mathbf{D}_t = \operatorname*{argmin}_{\mathbf{D}\in\mathcal{C}} \frac{1}{2}\mathrm{Tr}\,(\mathbf{D}^\top\mathbf{D}\bar{\mathbf{C}}_t) - \mathrm{Tr}\,(\mathbf{D}^\top\bar{\mathbf{B}}_t).$$

Output: Final iterate $\mathbf{D}_T$.

where $\mathbf{x}$ is drawn from the data distribution and forms an i.i.d. stream $(\mathbf{x}_t)_t$. In the finite-sample setting, (4.4) reduces to (4.3) when $\mathbf{x}_t$ is drawn uniformly at random from $\{\mathbf{x}^{(i)}, i \in [1,n]\}$. We then write i_t the sample number selected at time t.

The online matrix factorization algorithm proposed by Mairal et al. (2010) is summarized in Algorithm 1. It draws a sample $\mathbf{x}_t$ at each iteration, and uses it to improve the current iterate $\mathbf{D}_{t-1}$. For this, it first computes the code $\boldsymbol{\alpha}_t$ associated to $\mathbf{x}_t$ on the current dictionary:

$$\boldsymbol{\alpha}_t \triangleq \operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^k} \frac{1}{2}\|\mathbf{x}_t - \mathbf{D}_{t-1}\boldsymbol{\alpha}\|_2^2 + \lambda\Omega(\boldsymbol{\alpha}). \tag{4.5}$$

Then, it updates $\mathbf{D}_t$ to make it optimal in reconstructing past samples $(\mathbf{x}_s)_{s\leqslant t}$ from previously computed codes $(\boldsymbol{\alpha}_s)_{s\leqslant t}$:

$$\mathbf{D}_t \in \operatorname*{argmin}_{\mathbf{D}\in\mathcal{C}}\Big(\bar{g}_t(\mathbf{D}) \triangleq \frac{1}{t}\sum_{s=1}^{t}\frac{1}{2}\|\mathbf{x}_s - \mathbf{D}\boldsymbol{\alpha}_s\|_2^2 + \lambda\Omega(\boldsymbol{\alpha}_s)\Big). \tag{4.6}$$

Importantly, minimizing $\bar{g}_t$ is equivalent to minimizing the quadratic function

$$\mathbf{D} \to \frac{1}{2}\mathrm{Tr}\,(\mathbf{D}^\top\mathbf{D}\bar{\mathbf{C}}_t^\top) - \mathrm{Tr}\,(\mathbf{D}^\top\bar{\mathbf{B}}_t),$$

where $\bar{\mathbf{B}}_t$ and $\bar{\mathbf{C}}_t$ are small matrices that summarize previously seen samples and codes:

$$\bar{\mathbf{B}}_t = \frac{1}{t}\sum_{s=1}^{t}\mathbf{x}_s\boldsymbol{\alpha}_s^\top \qquad \bar{\mathbf{C}}_t = \frac{1}{t}\sum_{s=1}^{t}\boldsymbol{\alpha}_s\boldsymbol{\alpha}_s^\top.$$

As the constraints $\mathcal{C}$ have a separable structure per atom, Mairal et al. (2010) uses projected block coordinate descent to minimize $\bar{g}_t$. The function gradient writes $\nabla\bar{g}_t(\mathbf{D}) = \mathbf{D}\bar{\mathbf{C}}_t - \bar{\mathbf{B}}_t$, and it is therefore

enough to maintain $\bar{\mathbf{B}}_t$ and $\bar{\mathbf{C}}_t$ in memory to solve (4.6). $\bar{\mathbf{B}}_t$ and $\bar{\mathbf{C}}_t$ are updated online, using the rules (4.7) (Algorithm 1).

The function $\bar{g}_t$ is an upper-bound surrogate of the true current empirical risk, whose definition involves the regression minima computed on current dictionary $\mathbf{D}$:

$$\bar{f}_t(\mathbf{D}) \triangleq \frac{1}{t}\sum_{s=1}^{t} \min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \frac{1}{2}\left\|\mathbf{x}_s - \mathbf{D}\boldsymbol{\alpha}\right\|_2^2 + \lambda\Omega(\boldsymbol{\alpha}) \leqslant \bar{g}_t(\mathbf{D}). \tag{4.8}$$

Using empirical processes theory (Van der Vaart, 2000), it is possible to show that minimizing $\bar{f}_t$ at each iteration asymptotically yields a stationary point of the expected risk (4.4). Unfortunately, minimizing (4.8) is expensive as it involves the computation of optimal current codes for every previously seen sample at each iteration, which boils down to naive alternate-minimization.

In contrast, $\bar{g}_t$ is much cheaper to minimize than $\bar{f}_t$, using block coordinate descent. It is possible to show that $\bar{g}_t$ converges towards a locally tight upper-bound of the objective $\bar{f}_t$ and that minimizing $\bar{g}_t$ at each iteration also asymptotically yields a stationary point of the expected risk (4.4). This establishes the correctness of the *online matrix factorization* algorithm (OMF). In practice, the OMF algorithm performs a single pass of block coordinate descent: the minimization step is inexact. This heuristic will be justified as one of our theoretical contribution in Chapter 5.

Mini-batches and learning weights

For efficiency, it is essential to use mini-batches $\{\mathbf{x}_s,\ s \in \mathcal{T}_t\}$ of size η instead of single samples in the iterations (Mairal et al., 2010). The surrogate parameters $\bar{\mathbf{B}}_t$, $\bar{\mathbf{C}}_t$ are then updated by the mean value of $\{(\mathbf{x}_s\boldsymbol{\alpha}_s^\top, \boldsymbol{\alpha}_s\boldsymbol{\alpha}_s^\top)\}_{s\in\mathcal{T}_t}$ over the batch. The optimal size of the mini-batches is usually close to k. (4.7) uses the sequence of weights $(\frac{1}{t})_t$ to update parameters $\bar{\mathbf{B}}_t$ and $\bar{\mathbf{C}}_t$. Mairal et al. (2010) replaces these weights with a sequence $(w_t)_t$, which can decay more slowly to give more importance to recent samples in $\bar{g}_t$. These weights will prove important in our analysis.

We turn to introduce the new matrix factorization algorithm at the core of our contribution.

4.4 ALGORITHM OUTLINE: STOCHASTIC SUBSAMPLING FOR HIGH DIMENSIONAL DATA DECOMPOSITION

The online algorithm presented in Section 4.3 is very efficient to factorize matrices that have a large number of columns (i.e., with a large number of samples n), but a reasonable number of rows — the dataset is not very high dimensional. However, it is not designed to deal with very high number of rows: the cost of a single iteration depends linearly on p. On terabyte-scale datasets from fMRI with $p = 2 \cdot 10^5$ features, the original online algorithm requires one week to reach convergence. This is a major motivation for designing new matrix factorization algorithms that scale in *both directions*.

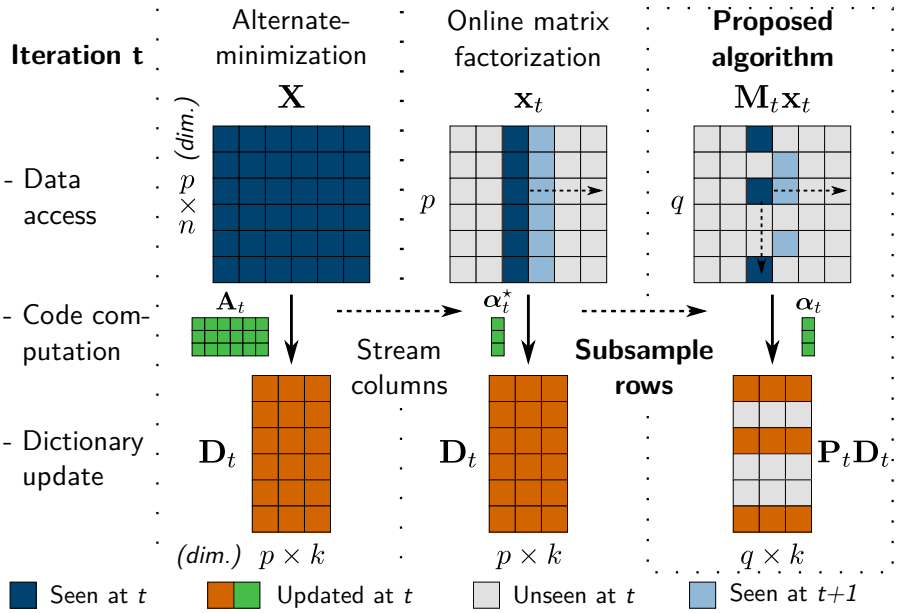


Figure 4.1 – *Stochastic subsampling* further improves online matrix factorization to handle datasets with large number of columns and rows. $\mathbf{X}$ is the input $p \times n$ matrix, $\mathbf{D}_t$ and $\mathbf{A}_t$ are respectively the dictionary and code at time t.

In the large-sample regime $p \gg k$, the underlying dimensionality of columns may be much lower than the actual p: the rows of a single column drawn at random are therefore correlated and redundant. This guides us on how to scale online matrix factorization with regard to the number of rows:

— The online algorithm OMF uses a *single* column of (or mini-batch) of $\mathbf{X}$ at each iteration to enrich the average surrogate and update the *whole* dictionary.

— We go a step beyond and use a *fraction* of a single column of $\mathbf{X}$ to refine a fraction of the dictionary.

More precisely, we draw a column and observe only *some* of its rows at each iteration, to refine these rows of the dictionary, as illustrated in Figure 4.1. To take into account all features from the dataset, rows are selected at random at each iteration: we call this technique *stochastic subsampling*. Stochastic subsampling reduces the efficiency of the dictionary update *per iteration*, as less information is incorporated in the current iterate $\mathbf{D}_t$. On the other hand, with a correct design, the cost of a single iteration can be considerably reduced, as it grows with the number of observed features. Section 6.1 shows that the proposed algorithm is an order of magnitude faster than the original OMF on large and redundant datasets.

First, we formalize the idea of working with a fraction of the p rows at a single iteration. We adapt the online matrix factorization algorithm, to reduce the iteration cost by a factor close to the ratio of selected rows. This defines a new online algorithm, called *subsampled online matrix factorization* (SOMF). At each iteration, it uses q rows of the column $\mathbf{x}_t$ to update the sequence of iterates $(\mathbf{D}_t)_t$.

4.5 SUBSAMPLED ONLINE MATRIX FACTORIZATION

Formally, as in online matrix factorization, we consider a sample stream $(x_t)_t$ in $\mathbb{R}^p$ that cycles onto a finite sample set $\{x^{(i)}, i \in [1, n]\}$, and minimize the empirical risk (4.3). Note that we solve the fully observed problem despite the use of subsampled data, unlike other recent work on low-rank factorization (Mardani et al., 2015). Extensions to partially observed data is discussed in Section 6.2.

4.5.1 *Stochastic subsampling*

We want to reduce the time complexity of a single iteration. In the original algorithm, the complexity depends linearly on the sample dimension p in three aspects:

— $x_t \in \mathbb{R}^p$ is used to compute the code α_t,

— it is used to update the surrogate parameters $\bar{B}_t \in \mathbb{R}^{p \times k}$,

— $D_t \in \mathbb{R}^{p \times k}$ is fully updated at each iteration.

Our algorithm reduces the dimensionality of these steps at each iteration, such that p becomes $q = \frac{p}{r}$ in the time complexity analysis, where $r > 1$ is a *reduction factor*. Formally, we randomly draw, at iteration t, a mask M_t that "selects" a random subset of x_t. We use it to drop a part of the features of x_t and to "freeze" these features in dictionary D at iteration t.

It is convenient to consider M_t as a $\mathbb{R}^{p \times p}$ random diagonal matrix, such that each coefficient is a Bernouilli variable with parameter $\frac{1}{r}$, normalized to be 1 in expectation. $\forall j \in [0, p-1]$,

$$\mathbb{P}\big[M_t[j,j] = r\big] = \frac{1}{r}, \quad \mathbb{P}\big[M_t[j,j] = 0\big] = 1 - \frac{1}{r}. \tag{4.9}$$

Thus, r describes the average proportion of observed features and $M_t x_t$ is a non-biased, low-dimensional estimator of x_t:

$$\mathbb{E}\big[\|M_t x_t\|_0\big] = \frac{p}{r} = q \qquad \mathbb{E}\big[M_t x_t\big] = x_t.$$

with $\|\cdot\|_0$ counting the number of non-zero coefficients. We define the pair of orthogonal projectors $P_t \in \mathbb{R}^{q \times p}$ and $P_t^\perp \in \mathbb{R}^{(p-q) \times p}$ that project $\mathbb{R}^p$ onto $\mathrm{Im}(M_t)$ and $\mathrm{Ker}(M_t)$. In other words, $P_t Y$ and $P_t^\perp Y$ are the submatrices of $Y \in \mathbb{R}^{p \times y}$ with rows respectively selected and not selected by M_t. In algorithms, $P_t Y \leftarrow Z \in \mathbb{R}^{q \times n}$ assigns the rows of Z to the rows of Y selected by P_t, by an abuse of notation.

In brief, subsampled online matrix factorization, defined in Algorithm 2, follows the outer loop of online matrix factorization, with the following major modifications at iteration t:

— it uses $M_t x_t$ and low-size statistics instead of x_t to estimate the code α_t and the surrogate g_t,

— it updates a subset of the dictionary $P_t D_{t-1}$ to reduce the surrogate value $\bar{g}_t(D)$. Relevant parameters of $\bar{g}_t$ are computed using $P_t x_t$ and α_t only.

Algorithm 2 Subsampled online matrix factorization (SOMF)

Input: Initial iterate $\mathbf{D}_0$, weight sequences $(w_t)_{t>0}$, $(\gamma_c)_{c>0}$, sample set $\{\mathbf{x}^{(i)}\}_{i>0}$, number of iterations T.

for t from 1 to T **do**

Draw $\mathbf{x}_t = \mathbf{x}^{(i)}$ at random and $\mathbf{M}_t$ following (4.9).

Update the regression parameters for sample i:

$$\begin{aligned} c^{(i)} &\leftarrow c^{(i)} + 1, & \gamma &\leftarrow \gamma_{c^{(i)}}. \\ \boldsymbol{\beta}_t^{(i)} &\leftarrow (1-\gamma)\mathbf{G}_{t-1}^{(i)} + \gamma \mathbf{D}_{t-1}^\top \mathbf{M}_t \mathbf{x}^{(i)}, & \boldsymbol{\beta}_t &\leftarrow \boldsymbol{\beta}_t^{(i)}. \\ \mathbf{G}_t^{(i)} &\leftarrow (1-\gamma)\mathbf{G}_{t-1}^{(i)} + \gamma \mathbf{D}_{t-1}^\top \mathbf{M}_t \mathbf{D}_{t-1}, & \mathbf{G}_t &\leftarrow \bar{\mathbf{G}}_t^{(i)}. \end{aligned}$$

Compute the approximate code for $\mathbf{x}_t$:

$$\boldsymbol{\alpha}_t \leftarrow \operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^k} \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{G}_t \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \boldsymbol{\beta}_t + \lambda\, \Omega(\boldsymbol{\alpha}). \tag{4.10}$$

Update the parameters of the aggregated surrogate $\bar{g}_t$:

$$\begin{aligned} \bar{\mathbf{C}}_t &\leftarrow (1-w_t)\bar{\mathbf{C}}_{t-1} + w_t \boldsymbol{\alpha}_t \boldsymbol{\alpha}_t^\top. \\ \mathbf{P}_t \bar{\mathbf{B}}_t &\leftarrow (1-w_t)\mathbf{P}_t \bar{\mathbf{B}}_{t-1} + w_t \mathbf{P}_t \mathbf{x}_t \boldsymbol{\alpha}_t^\top. \end{aligned}$$

Compute simultaneously (using Algorithm 3 for 1st line):

$$\begin{aligned} \mathbf{P}_t \mathbf{D}_t &\leftarrow \operatorname*{argmin}_{\mathbf{D}^r \in \mathcal{C}^r} \frac{1}{2}\mathrm{Tr}\left(\mathbf{D}^{r\top} \mathbf{D}^r \bar{\mathbf{C}}_t\right) - \mathrm{Tr}\left(\mathbf{D}^{r\top} \mathbf{P}_t \bar{\mathbf{B}}_t\right). \\ \mathbf{P}_t^\perp \bar{\mathbf{B}}_t &\leftarrow (1-w_t)\mathbf{P}_t^\perp \bar{\mathbf{B}}_{t-1} + w_t \mathbf{P}_t^\perp \mathbf{x}_t \boldsymbol{\alpha}_t^\top. \end{aligned} \tag{4.11}$$

Output: Final iterate $\mathbf{D}_T$.

We now present the three steps of SOMF in details. For comparison purpose, we write all variables that would be computed following the OMF rules at iteration t with a * superscript. For simplicity, in Algorithm 2 and in the following paragraphs, we assume that we use one sample per iteration —in practice, we use mini-batches of size η. The next derivations are transposable when a batch I_t is drawn at iteration t instead of a single sample i_t.

4.5.2 *Code computation*

In the OMF algorithm presented in Section 4.3, $\boldsymbol{\alpha}_t^\star$ is obtained by solving (4.5), namely

$$\boldsymbol{\alpha}_t^\star \in \operatorname*{argmin}_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{G}_t^\star \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \boldsymbol{\beta}_t^\star + \lambda\Omega(\boldsymbol{\alpha}), \tag{4.12}$$

where $\mathbf{G}_t^\star = \mathbf{D}_{t-1}^\top \mathbf{D}_{t-1}$ and $\boldsymbol{\beta}_t^\star = \mathbf{D}_{t-1}^\top \mathbf{x}_t$. For large p, the computation of $\mathbf{G}_t^\star$ and $\boldsymbol{\beta}_t^\star$ dominates the complexity of the regression step, which depends almost linearly on p. To reduce this complexity, we use *estimators* for $\mathbf{G}_t^\star$ and $\boldsymbol{\beta}_t^\star$, computed at a cost proportional to the reduced dimension q. We propose three kinds of estimators with different properties.

4.5.2.1 *Masked loss*

The most simple *unbiased* estimation of $\mathbf{G}_t^\star$ and $\boldsymbol{\beta}_t^\star$ whose computation cost depends on q is obtained by subsampling matrix products with $\mathbf{M}_t$:

$$\begin{aligned} \mathbf{G}_t &= \mathbf{D}_{t-1}^\top \mathbf{M}_t \mathbf{D}_{t-1} \\ \boldsymbol{\beta}_t &= \mathbf{D}_{t-1}^\top \mathbf{M}_t \mathbf{x}_t. \end{aligned} \tag{a}$$

Solving (4.10) then amounts to minimize the *masked loss*

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^k} \frac{1}{2}\|\mathbf{M}_t(\mathbf{x}_t - \mathbf{D}_{t-1}^\top \boldsymbol{\alpha})\|_2^2 + \lambda\Omega(\boldsymbol{\alpha}). \tag{4.13}$$

$\mathbf{G}_t$ and $\boldsymbol{\beta}_t$ are computed in a number of operations proportional to q, which brings a speed-up factor of almost r in the code computation for large p. On large data, using estimators (a) instead of exact $\mathbf{G}_t^\star$ and $\boldsymbol{\beta}_t^\star$ proves very efficient during the first epochs (cycles over the columns).[8] However, due to the masking, $\mathbf{G}_t$ and $\boldsymbol{\beta}_t$ are not *consistent* estimators: they do not converge to $\mathbf{G}_t^\star$ and $\boldsymbol{\beta}_t^\star$ for large t, which breaks theoretical guarantees on the algorithm output. Empirical results in Section 6.1.5 show that the sequence of iterates approaches a critical point of the risk (4.3), but may then oscillate around it.

[8] *Estimators (a) are also available in the infinite sample setting, when minimizing the expected risk (4.4) from a i.i.d sample stream* $(\mathbf{x}_t)_t$.

4.5.2.2 *Averaging over epochs*

At iteration t, the sample $\mathbf{x}_t$ is drawn from a finite set of samples $\{\mathbf{x}^{(i)}\}_i$. This allows to average estimators over previously seen samples and address the non-consistency issue of (a). Namely, we keep in memory $2n$ estimators, written $(\mathbf{G}_t^{(i)}, \boldsymbol{\beta}_t^{(i)})_{1\leqslant i\leqslant n}$. We observe the sample $i = i_t$ at iteration t and use it to update the i-th estimators $\bar{\mathbf{G}}_t^{(i)}$, $\bar{\boldsymbol{\beta}}_t^{(i)}$ following

$$\begin{aligned} \boldsymbol{\beta}_t^{(i)} &= (1-\gamma)\mathbf{G}_{t-1}^{(i)} + \gamma \mathbf{D}_{t-1}^\top \mathbf{M}_t \mathbf{x}^{(i)} \\ \mathbf{G}_t^{(i)} &= (1-\gamma)\mathbf{G}_{t-1}^{(i)} + \gamma \mathbf{D}_{t-1}^\top \mathbf{M}_t \mathbf{D}_t^{(i)}, \end{aligned}$$

where γ is a weight factor determined by the number of time the one sample i has been previously observed at time t. Precisely, given $(\gamma_c)_c$ a decreasing sequence of weights,

$$\gamma = \gamma_{c_t^{(i)}} \quad \text{where} \quad c_t^{(i)} = \left|\left\{s \leqslant t, \mathbf{x}_s = \mathbf{x}^{(i)}\right\}\right|.$$

All others estimators $\{\mathbf{G}_t^{(j)}, \boldsymbol{\beta}_t^{(j)}\}_{j\neq i}$ are left unchanged from iteration $t-1$. The set $\{\mathbf{G}_t^{(i)}, \boldsymbol{\beta}_t^{(i)}\}_{1\leqslant i\leqslant n}$ is used to define the *averaged* estimators

$$\begin{aligned} \mathbf{G}_t \triangleq \mathbf{G}_t^{(i)} &= \sum_{s\leqslant t, \mathbf{x}_s = \mathbf{x}^{(i)}} \gamma_{s,t}^{(i)} \mathbf{D}_{s-1}^\top \mathbf{M}_s \mathbf{D}_{s-1} \\ \boldsymbol{\beta}_t \triangleq \boldsymbol{\beta}_t^{(i)} &= \sum_{s\leqslant t, \mathbf{x}_s = \mathbf{x}^{(i)}} \gamma_{s,t}^{(i)} \mathbf{D}_{s-1}^\top \mathbf{M}_s \mathbf{x}^{(i)}, \end{aligned} \tag{b}$$

where $\gamma_{s,t}^{(i)} = \gamma_{c_t^{(i)}} \prod_{s<t, x_s = x^{(i)}} (1 - \gamma_{c_s^{(i)}})$. Using β_t and G_t in (4.10), α_t minimizes the masked loss averaged over the previous iterations where sample i appeared:

$$\min_{\alpha \in \mathbb{R}^k} \sum_{\substack{s \leqslant t \\ x_s = x^{(i)}}} \frac{\gamma_{s,t}^{(i)}}{2} \|M_s(x^{(i)} - D_{s-1}^\top \alpha)\|_2^2 + \lambda \Omega(\alpha). \tag{4.14}$$

The sequences $(G_t)_t$ and $(\beta_t)_t$ are *consistent* estimations of $(G_t^\star)_t$ and $(\beta_t^\star)_t$ — consistency arises from the fact that a single sample $x^{(i)}$ is observed with different masks along iterations. Solving (4.14) is made closer and closer to solving (4.12), to ensure the correctness of the algorithm (see Section 5.3). Yet, computing the estimators (b) is no more costly than computing (a) and still permits to speed up a single iteration close to r times. In the mini-batch setting, for every $i \in I_t$, we use the estimators $G_t^{(i)}$ and $\beta_t^{(i)}$ to compute $\alpha_t^{(i)}$. This method has a memory cost of $\mathcal{O}(n\,k^2)$. This is reasonable compared to the dataset size [1] if $p \gg k^2$.

4.5.2.3 *Exact Gram matrix computation*

To reduce the memory usage, another strategy is to use the *true* Gram matrix G_t and the estimator β_t from (b):

$$\begin{aligned} G_t &\triangleq G_t^\star = D_{t-1}^\top D_{t-1} \\ \beta_t &\triangleq \sum_{s \leqslant t, x_s = x^{(i)}} \gamma_{s,t}^{(i)} D_{s-1}^\top M_s x^{(i)} \end{aligned} \tag{c}$$

As previously, the consistency of $(\beta_t)_t$ ensures that (4.4) is correctly solved despite the approximation in $(\alpha_t)_t$ computation. With the partial dictionary update step we propose, it is possible to maintain G_t at a cost proportional to q. The time complexity of the coding step is thus similarly reduced when replacing (b) or (c) estimators in (4.12), but the latter option has a memory usage in $\mathcal{O}(n\,k)$. Although estimators (c) are slightly less efficient in the first epochs, they are a good compromise between resource usage and convergence. We summarize the characteristics of the three estimators (a)–(c) in Table 4.1, anticipating their empirical comparison in Section 6.1.

1. It is also possible to efficiently swap the estimators $(G_t^{(i)})_i$ on disk, as they are only accessed for $i = i_t$ at iteration t.

Table 4.1 – Comparison of estimators used for code computation

Est.	β_t	G_t	Convergence	Extra mem. cost	1st epoch perform.
(a)	Masked	Masked			✓
(b)	Averaged	Averaged	✓	$n\,k^2$	✓
(c)	Averaged	Exact	✓	$n\,k$	

4.5.3 *Dictionary update*

In the original online algorithm, the whole dictionary $\mathbf{D}_{t-1}$ is updated at iteration t. To reduce the time complexity of this step, we add a "freezing" constraint to the minimization (4.6) of $\bar{g}_t$. Every row r of $\mathbf{D}$ that corresponds to an *unseen* row r at iteration r (such that $\mathbf{M}_t[r, r] = 0$) remains unchanged. This casts the problem (4.6) into a lower dimensional space. Formally, the freezing operation comes out as a additional constraint in (4.6):

$$\mathbf{D}_t = \underset{\substack{\mathbf{D}\in\mathcal{C} \\ \mathbf{P}_t^\perp \mathbf{D} = \mathbf{P}_t^\perp \mathbf{D}_{t-1}}}{\operatorname{argmin}} \frac{1}{2}\operatorname{Tr}\left(\mathbf{D}^\top \mathbf{D}\bar{\mathbf{C}}_t\right) - \operatorname{Tr}\left(\mathbf{D}^\top \bar{\mathbf{B}}_t\right). \tag{4.15}$$

The constraints are separable into two blocks of rows. Recalling the notations of (4.2), for each atom $\mathbf{d}^{(j)}$, the rules $\|\mathbf{d}^{(j)}\| \leqslant 1$ and the freezing constraint $\mathbf{P}_t^\perp \mathbf{d}^{(j)} = \mathbf{P}_t^\perp \mathbf{d}_{t-1}^{(j)}$ can indeed be rewritten

$$\begin{cases} \|\mathbf{P}_t \mathbf{d}^{(j)}\| & \leqslant & 1 - \|\mathbf{d}_{t-1}^{(j)}\| + \|\mathbf{P}_t \mathbf{d}_{t-1}^{(j)}\| \triangleq r_t^{(j)} \\ \mathbf{P}_t^\perp \mathbf{d}^{(j)} & = & \mathbf{P}_t^\perp \mathbf{d}_{t-1}^{(j)}. \end{cases}$$

Solving (4.15) is therefore equivalent to solving the following problem in $\mathbb{R}^{q\times k}$, with $\mathbf{B}_t^r \triangleq \mathbf{P}_t \mathbf{B}_t$,

$$\mathbf{D}^r \in \underset{\mathbf{D}^r\in\mathcal{C}^r}{\operatorname{argmin}} \left(\frac{1}{2}\operatorname{Tr}\left(\mathbf{D}^{r\top}\mathbf{D}^r\bar{\mathbf{C}}_t\right) - \operatorname{Tr}\left(\mathbf{D}^{r\top}\bar{\mathbf{B}}_t^r\right) \triangleq \bar{g}_t^r(\mathbf{D}^r)\right) \tag{4.16}$$
$$\text{where } \mathcal{C}^r = \left\{\mathbf{D}^r \in \mathbb{R}^{q\times k} / \forall j \in [0, k-1], \|\mathbf{d}^{r(j)}\| \leqslant r_t^{(j)}\right\}.$$

The rows of $\mathbf{D}_t$ selected by $\mathbf{P}_t$ are then replaced with $\mathbf{D}^r$, while the other rows of $\mathbf{D}_t$ are unchanged from iteration $t-1$. Formally, $\mathbf{P}_t\mathbf{D}_t = \mathbf{D}^r$ and $\mathbf{P}_t^\perp \mathbf{D}_t = \mathbf{P}_t^\perp \mathbf{D}_{t-1}$. We solve (4.16) by a projected Block coordinate descent (BCD) similar to the one used in the original algorithm, but performed in a subspace of size q. We compute each column j of the gradient that we use in the block coordinate descent loop with $q \times k$ operations:

$$(\nabla \bar{g}_t^r(\mathbf{D}^r))^{(j)} = \mathbf{D}^r \bar{\mathbf{c}}_t^{(j)} - \bar{\mathbf{b}}_t^{r(j)} \in \mathbb{R}^q,$$

where $\bar{\mathbf{c}}_t^{(j)}$ and $\bar{\mathbf{b}}_t^{r(j)}$ are the j-th columns of $\bar{\mathbf{C}}_t$ and $\bar{\mathbf{B}}_t^r$. Each reduced atom $\mathbf{d}^{r(j)}$ is projected onto the elastic-net ball of radius $r_t^{(j)}$, at an average cost in $\mathcal{O}(q)$ following (Duchi et al., 2008; Mairal et al., 2010). This makes the complexity of a single-column update proportional to q. Performing the projection requires to keep in memory the values $\{n_t^{(j)} \triangleq 1 - \|\mathbf{d}_t^{(j)}\|\}_j$, which can be updated online at a negligible cost.

We provide the reduced dictionary update step in Algorithm 3, where we use the function `enet_projection`$(\mathbf{u}, r)$ that performs the orthogonal projection of $\mathbf{u} \in \mathbb{R}^q$ onto the elastic-net ball of radius r. As in the original algorithm, we perform a single pass over columns to solve (4.16). Dictionary update is now performed with a number of operations proportional to q, instead of p in the original algorithm. Thanks to the random nature of $(\mathbf{M}_t)_t$, updating $\mathbf{D}_{t-1}$ into $\mathbf{D}_t$ reduces $\bar{g}_t$ enough to ensure convergence.

Algorithm 3 Partial dictionary update

Input: Dictionary $\mathbf{D}_{t-1}$, projector $\mathbf{P}_t$, statistics $\bar{\mathbf{C}}_t$, $\bar{\mathbf{B}}_t$, norms $(n_{t-1}^{(j)})_{0\leqslant j<k}$, Gram matrix $\mathbf{G}_t$ (optional).
$\mathbf{D}_t \leftarrow \mathbf{D}_{t-1}$, $\quad \mathbf{G}_t \leftarrow \mathbf{G}_t - \mathbf{D}_{t-1}^\top \mathbf{P}_t \mathbf{D}_{t-1}$.
for $j \in$ permutation$([1,k])$ **do**
  $r_t^{(j)} \leftarrow n_{t-1}^{(j)} + \|\mathbf{P}_t \mathbf{d}_{t-1}^{(j)}\|$.
  $\mathbf{u} \leftarrow \mathbf{P}_t \mathbf{d}_{t-1}^{(j)} + \frac{1}{\bar{\mathbf{C}}_t[j,j]}(\mathbf{P}_t \bar{\mathbf{b}}_t^{(j)} - \mathbf{P}_t \mathbf{D}_t \bar{\mathbf{c}}_t^{(j)})$. ▷ in $\mathbb{R}^q$
  $\mathbf{P}_t \mathbf{d}_t^{(j)} \leftarrow$ `enet_projection`$(\mathbf{u}, r_t^{(j)})$. ▷ in $\mathbb{R}^q$
  $n_t^{(j)} \leftarrow r_t^{(j)} - \|\mathbf{P}_t \mathbf{d}_t^{(j)}\|$.
$\mathbf{G}_{t+1} \leftarrow \mathbf{G}_t + \mathbf{D}_t^\top \mathbf{P}_t \mathbf{D}_t$.
Output: Dictionary $\mathbf{D}_t$, norms $(n_t^{(j)})_j$, Gram matrix $\mathbf{G}_{t+1}$.

GRAM MATRIX COMPUTATION. Performing partial updates of $\mathbf{D}_t$ makes it possible to maintain the full Gram matrix $\mathbf{G}_t = \mathbf{G}_t^\star$ with a cost in $\mathcal{O}(q\,k^2)$ per iteration, as mentioned in Section 4.5.2.3. It is indeed enough to compute the reduced Gram matrix $\mathbf{D}^\top \mathbf{P}_t \mathbf{D}$ before and after the dictionary update:

$$\mathbf{G}_{t+1} = \mathbf{D}_t^\top \mathbf{D}_t = \mathbf{G}_t - \mathbf{D}_{t-1}^\top \mathbf{P}_t \mathbf{D}_{t-1}^\top + \mathbf{D}_t^\top \mathbf{P}_t \mathbf{D}_t^\top.$$

4.5.4 *Surrogate computation*

The computation of $\boldsymbol{\alpha}_t$ using one of the estimators above defines a surrogate $g_t(\mathbf{D}) \triangleq \frac{1}{2}\|\mathbf{x}_t - \mathbf{D}\boldsymbol{\alpha}_t\|_2^2 + \lambda\Omega(\boldsymbol{\alpha})$, which we use to update the aggregated surrogate $\bar{g}_t \triangleq (1-w_t)\bar{g}_{t-1} + w_t g_t$, as in online matrix factorization. We follow (4.7) (with weights $(w_t)_t$) to update the matrices $\bar{\mathbf{B}}_t$ and $\bar{\mathbf{C}}_t$, which define $\bar{g}_t$ up to constant factors. The update of $\bar{\mathbf{B}}_t$ requires a number of operations proportional to p, which we want to avoid. Fortunately, it is possible to leverage the use of two threads to circumvent this issue.

4.5.4.1 *Parallel parameter update*

Performing block coordinate descent on $\bar{g}_t^r$ indeed requires to access $\bar{\mathbf{B}}_t^r = \mathbf{P}_t \bar{\mathbf{B}}_t$ only. Assuming we may use use more than two threads, this allows to parallelize the dictionary update step with the update of $\mathbf{P}_t^\perp \bar{\mathbf{B}}_t$. In the main thread, we compute $\mathbf{P}_t \bar{\mathbf{B}}_t$ following

$$\mathbf{P}_t \bar{\mathbf{B}}_t \leftarrow (1-w_t)\mathbf{P}_t \bar{\mathbf{B}}_{t-1} + w_t \mathbf{P}_t \mathbf{x}_t \boldsymbol{\alpha}_t^\top.$$

which has a cost proportional to q. Then, we update in parallel the dictionary and the rows of $\bar{\mathbf{B}}_t$ that are not selected by $\mathbf{M}_t$:

$$\mathbf{P}_t^\perp \bar{\mathbf{B}}_t \leftarrow (1-w_t)\mathbf{P}_t^\perp \bar{\mathbf{B}}_{t-1} + w_t \mathbf{P}_t^\perp \mathbf{x}_t \boldsymbol{\alpha}_t^\top.$$

This update requires $k(p-q)\eta$ operations (one matrix-matrix product) for a mini-batch of size η. In contrast, with appropriate implementation, the dictionary update step requires $4\,k\,q^2$ to $6\,k\,q^2$ operations, among which $2\,k\,q^2$ come from slower matrix-vector products.

Assuming $k \sim \eta$, updating $\bar{\mathbf{B}}_t$ is faster than updating the dictionary up to $r \sim 10$, and performing (4.11) on a second thread is seamless in term of wall-clock time. More threads may be used for larger reduction or batch size.

4.5.4.2 *Truly partial update*

It is in fact possible to replace the parallel updates of $\bar{\mathbf{B}}_t$ with the following asynchronous updates:

$$\begin{aligned} \mathbf{P}_t^{\perp} \bar{\mathbf{B}}_t &\triangleq \mathbf{P}_t^{\perp} \bar{\mathbf{B}}_{t-1} \\ \mathbf{P}_t \bar{\mathbf{B}}_t &\triangleq (1 - \frac{p}{q} w_t) \mathbf{P}_t \bar{\mathbf{B}}_{t-1} + \frac{p}{q} w_t \mathbf{P}_t \mathbf{x}_t \boldsymbol{\alpha}_t^\top, \end{aligned} \tag{4.19}$$

while maintaining the convergence guarantees presented in Chapter 5. This was the approach we proposed in a first version of the SOMF algorithm (Mensch et al., 2016a). (4.19) has the advantage of updating only q rows of $\bar{\mathbf{B}}_t$ at each iteration. On the other hand, it introduces an extra source of perturbation to the original OMF algorithm. The effect of the perturbation is unfortunately too strong compared to the computational speed-up provided by (4.19). Theoretical analysis bears resemblance to a recent work of Leblond et al. (2017) proposed for the SAGA algorithm. Note that we will reuse (4.19) in the empirical adaptation of SOMF for matrix completion, in Section 6.2.

4.5.5 *Weight sequences*

Algorithm 2 require to specify $(w_t)_t$ and $(\gamma_c)_c$. We provide usable form for those sequences in Assumption **(B)** of the analysis: $w_t = \frac{1}{t^u}$ and $\gamma_c = \frac{1}{c^v}$, where $u \in (\frac{11}{12}, 1)$ and $v \in \left(\frac{3}{4}, 3u - 2\right)$ to ensure convergence. Weights have little impact on convergence speed in practice.

4.5.6 *Subsampling and time complexity*

Subsampling may be used in only *some* of the steps of Algorithm 2, with the other steps following Algorithm 1. Whether to use subsampling or not in each step depends on the trade-off between the computational speed-up it brings and the approximations it makes. It is useful to understand how complexity of OMF evolves with p. We write s the average number of non-zero coefficients in $(\boldsymbol{\alpha}_t)_t$ ($s = k$ when $\Omega = \|\cdot\|_2^2$). OMF complexity has three terms:

(i) $\mathcal{O}(p\,k^2)$: computation of the Gram matrix $\mathbf{G}_t$, update of the dictionary $\mathbf{D}_t$ with block coordinate descent,

(ii) $\mathcal{O}(p\,k\,\eta)$: computation of $\boldsymbol{\beta}_t = \mathbf{D}_{t-1}^\top \mathbf{x}_t$ and of $\bar{\mathbf{B}}_t$ using $\mathbf{x}_t \boldsymbol{\alpha}_t^\top$,

(iii) $\mathcal{O}(k\,s^2\,\eta)$: computation of $\boldsymbol{\alpha}_t$ using $\mathbf{G}_t$ and $\boldsymbol{\beta}_t$, using matrix inversion or elastic-net regression.

Using subsampling turns p into $q = \frac{p}{r}$ in the expressions above. It improves single iteration time when the cost of regression $\mathcal{O}(k\,s^2\,\eta)$ is dominated by another term. This happens whenever $\frac{p}{r} > s^2$, where r is the reduction factor used in the algorithm. Subsampling can

bring performance improvement up to $r \sim \frac{p}{s^2}$. It can be introduced in either computations from (i) or (ii), or both. When using small batch size, i.e., when $\eta < k$, computations from (i) dominates complexity, and subsampling should be first introduced in dictionary update (i), and for code computation (ii) beyond a certain reduction ratio. On the other hand, with large batch size $\eta > k$, subsampling should be first introduced in code computation, then in the dictionary update step. The reasoning above ignore potentially large constants. The best trade-offs in using subsampling must be empirically determined, which we do in Chapter 6.

CONCLUSION

This chapter introduced the SOMF algorithm in detail and provided the intuitions that guided its design. SOMF possess asymptotic almost-sure convergence guarantees, that may be stated in an optimization framework adapted to a wider set of problems than matrix factorization. Chapter 5 provides a theoretical analysis of SOMF.

5

ALGORITHM PROPERTIES VIA STOCHASTIC APPROXIMATE MAJORIZATION-MINIMIZATION

In this chapter, we present the *stochastic approximate majorization-minimization* (SAMM) framework and the convergence analysis of SOMF. At its core, our analysis controls the perturbations that approximate code computation and approximate surrogate minimization introduce in the sequence of iterate $(\mathbf{D}_t)_t$. We establish that the iterate sequence converges toward a critical point of the objective (4.3), *i. e.*

$$\min_{\mathbf{D}\in\mathcal{C}} \bar{f}(\mathbf{D}) \triangleq \frac{1}{n}\sum_{i=1}^{n} \min_{\boldsymbol{\alpha}\in\mathbb{R}^k} \left(\frac{1}{2}\|\mathbf{x}-\mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda\,\Omega(\boldsymbol{\alpha})\right),$$

with positive directional derivatives.

To better understand the mechanisms at stake, and make our approach modular, we work in the more general framework of stochastic majorization minimization (Mairal, 2013b), which abstracts the steps of online matrix factorization. We start by recalling what is stochastic majorization-minimization and how it encompass the OMF algorithm. We then turn to analyse the convergence properties of SAMM algorithms. We use these to obtain guarantees on SOMF convergence.

5.1 PRIOR ART: STOCHASTIC MAJORIZATION-MINIMIZATION

Online matrix factorization belongs to a wider category of algorithms introduced by Mairal (2013b), that minimize locally tight upper bounding surrogates instead of a more complex objective, in order to solve an expected risk minimization problem

$$\min_{\theta\in\Theta} \bar{f}(\theta) \triangleq \mathbb{E}_{\mathbf{x}}[f(\theta,\mathbf{x})].$$

Generalizing online matrix factorization, we introduce in Algorithm 4 the *stochastic majorization-minimization* (SMM) algorithm, which is at the core of our theoretical contribution. SMM algorithms extends the popular class of majorization-minimization algorithms (Ortega and Rheinboldt, 1970), of which gradient descent (Cauchy, 1847) and batch alternated minimization for matrix factorization are instances.

In online matrix factorization, the true empirical risk functions $\bar{f}_t$ and their *surrogates* $\bar{g}_t$ follow the update rules, with generalized weight $(w_t)_t$ set to $(\frac{1}{t})_t$ in (4.6) – (4.8):

$$\bar{f}_t \triangleq (1-w_t)\bar{f}_{t-1} + w_t f_t, \quad \bar{g}_t \triangleq (1-w_t)\bar{g}_{t-1} + w_t g_t, \tag{5.1}$$

Algorithm 4 Stochastic majorization-minimization (Mairal, 2013b)

Input: Initial iterate θ_0, weight sequence $(w_t)_{t>0}$, sample stream $(x_t)_{t>0}$, number of iteration T.
for t from 1 to T **do**
Draw $x_t \sim \mathcal{P}$, get $f_t : \theta \in \Theta \to f(x_t, \theta)$.
Construct a surrogate of f_t near θ_{t-1}, that meets

$$g_t \geqslant f_t, \quad g_t(\theta_{t-1}) = f_t(\theta_{t-1}).$$

Update the aggregated surrogate:

$$\bar{g}_t = (1 - w_t)\bar{g}_{t-1} + w_t g_t.$$

Compute

$$\theta_t = \operatorname*{argmin}_{\theta \in \Theta} \bar{g}_t(\theta).$$

Output: Final iterate θ_T.

where the *pointwise* loss function and its surrogate are

$$\begin{aligned} f_t(\mathbf{D}) &\triangleq \min_{\alpha \in \mathbb{R}^k} \frac{1}{2}\|x_t - \mathbf{D}\alpha\|_2^2 + \lambda\Omega(\alpha), \\ g_t(\mathbf{D}) &\triangleq \frac{1}{2}\|x_t - \mathbf{D}\alpha_t\|_2^2 + \lambda\Omega(\alpha_t). \end{aligned} \tag{5.2}$$

The function g_t is a majorizing surrogate of f_t: $g_t \geqslant f_t$, and g_t is tangent to f_t in $\mathbf{D}_{t-1}$, i.e, $g_t(\mathbf{D}_{t-1}) = f_t(\mathbf{D}_{t-1})$ and $\nabla(g_t - f_t)(\mathbf{D}_{t-1}) = 0$. Recall that at each step of online matrix factorization:

— The surrogate g_t is computed along with α_t, using (4.5).
— The parameters $\bar{\mathbf{B}}_t, \bar{\mathbf{C}}_t$ are updated following (4.7). They define the *aggregated* surrogate $\bar{g}_t$ up to a constant.
— The quadratic function $\bar{g}_t$ is minimized efficiently by block coordinate descent, using parameters $\bar{\mathbf{B}}_t$ and $\bar{\mathbf{C}}_t$ to compute its gradient.

The SMM framework simply formalizes the three steps above, for a larger variety of loss functions $f_t(\theta) \triangleq f(\theta, x_t)$, where θ is the parameter we want to learn ($\mathbf{D}$ in the online matrix factorization setting). At iteration t, a surrogate g_t of the loss f_t is computed to update the aggregated surrogate $\bar{g}_t$ following (5.1). The surrogate functions $(g_t)_t$ should upper-bound the loss functions $(f_t)_t$ and be tight in the current iterate θ_{t-1} (*e.g.*, the dictionary $\mathbf{D}_{t-1}$). This simply means that $f_t(\theta_{t-1}) = g_t(\theta_{t-1})$ and $\nabla(f_t - g_t)(\theta_{t-1}) = 0$. Computing $\bar{g}_t$ can be done if g_t is defined simply; in OMF, it is linearly parametrized by the couple of matrices $(\alpha_t\alpha_t^\top, x_t\alpha_t^\top)$. $\bar{g}_t$ is then minimized to obtain a new iterate θ_t — this is summarized in Algorithm 4.

SURROGATE EXAMPLES. Online matrix factorization uses a variational form for g_t, that involves the computation of a minimizer.

When f_t is L-smooth, we obtain another well known tight upper-bound surrogate by setting, for all $\theta \in \Theta$,

$$g_t(\theta) \triangleq f_t(\theta_{t-1}) + \langle \theta - \theta_{t-1}, \nabla f_t(\theta_{t-1}) \rangle + \frac{L}{2}\|\theta - \theta_{t-1}\|_2^2.$$

Using this class of surrogates in SMM exactly amount to perform stochastic gradient descent with step-sizes $\frac{w_t}{L}$.

It can be shown following Mairal (2013b) that stochastic majorization minimization algorithms asymptotically find stationary points of the expected risk $\mathbb{E}_x[f(\theta, x)]$ under mild assumptions recalled in Section 5.3. SMM admits the same mini-batch extensions as OMF.

We now propose an extension of the SMM framework that allows both the majorization and minimization steps to be approximated. We will show that convergence guarantees may be maintained despite those approximations.

5.2 STOCHASTIC APPROXIMATE MAJORIZATION-MINIMIZATION

The SOMF algorithm can be understood within the stochastic majorization minimization framework. The modifications that we propose are indeed perturbations to the first and third steps of the SMM presented in Algorithm 4:

— The code is computed approximately: the surrogate is only an *approximate* majorizing surrogate of f_t near $\mathbf{D}_{t-1}$.

— The surrogate objective is only *reduced* and not minimized, due to the added constraint and the fact that we perform only one pass of block coordinate descent.

We propose a new *stochastic approximate majorization-minimization* (SAMM) framework handling these perturbations:

— A majorization step (4 – Algorithm 4), computes an *approximate surrogate* of f_t near θ_{t-1}: $g_t \approx g_t^\star$, where g_t is a true upper-bounding surrogate of $\bar{f}_t$.

— A minimization step (6 – Algorithm 4), finds θ_t by reducing *enough* the objective $\bar{g}_t$: $\theta_t \approx \theta_t^\star \triangleq \operatorname{argmin}_{\theta \in \Theta} \bar{g}_t(\theta)$, which implies $\bar{g}_t(\theta_t) \gtrsim \bar{g}_t(\theta_t^\star)$.

The SAMM framework is general, in the sense that approximations are not specified. The next section provides a theoretical analysis of the approximation of SAMM and establishes how SOMF is an instance of SAMM. Its main practical result is Proposition 5.1, that provides convergence guarantees for SOMF, under the same assumptions made for OMF in Mairal et al. (2010).

5.3 CONVERGENCE ANALYSIS

We establish the convergence of SOMF under reasonable assumptions. For the sake of clarity, we first state our principal result (Proposition 5.1), that guarantees SOMF convergence. It is a corollary of a

more general result on SAMM algorithms. To present this broader result, we recall the theoretical guarantees of the stochastic majorization-minimization algorithm from Mairal (2013b) (Proposition 5.2); then, we show how the algorithm can withstand pertubations (Proposition 5.3). Proofs are reported in Section A.1. SAMM convergence is proven before establishing SOMF convergence as a corollary of this broader result. As a side contribution, our extension proves that performing a single pass of block coordinate descent to update the dictionary, an important heuristic introduced by Mairal et al. (2010), is indeed correct.

5.3.1 *Convergence of SOMF*

Similar to Mairal et al. (2010) and Mardani et al. (2015), we show that the sequence of iterates $(\mathbf{D}_t)_t$ asymptotically reaches a critical point of the empirical risk (4.3). We introduce the same hypothesis on the code covariance estimation $\bar{\mathbf{C}}_t$ as in Mairal et al. (2010) and a similar one on $\mathbf{G}_t$ — they ensure strong convexity of the surrogate and boundedness of $(\boldsymbol{\alpha}_t)_t$. They do not cause any loss of generality as they are met in practice after a few iterations, if r is chosen reasonably low, so that $q > k$. The following hypothesis can also be guaranteed by adding small ℓ_2 regularizations to $\bar{f}$.

(A) There exists $\rho > 0$ such that for all $t > 0$, $\bar{\mathbf{C}}_t, \mathbf{G}_t \succ \rho\mathbf{I}$.

We further assume, that the weights $(w_t)_t$ and $(\gamma_c)_c$ decay at specific rates. We specify simple weight sequences, but the proofs can be adapted for more complex ones.

(B) There exists $u \in (\frac{11}{12}, 1)$ and $v \in (\frac{3}{4}, 3u - 2)$ such that, for all $t > 0$, $c > 0$, $w_t = t^{-u}$, $\gamma_c = c^{-v}$.

The following convergence result then applies to any sequence $(\mathbf{D}_t)_t$ produced by SOMF, using estimators (b) or (c). $\bar{f}$ is the empirical risk defined in (4.3).

Proposition 5.1 (SOMF convergence)**.** *Under assumptions **(A)** and **(B)**, $\bar{f}(\mathbf{D}_t)$ converges with probability one and every limit point $\mathbf{D}_\infty$ of $(\mathbf{D}_t)_t$ is a stationary point of $\bar{f}$: for all $\mathbf{D} \in \mathcal{C}$*

$$\nabla \bar{f}(\mathbf{D}_\infty, \mathbf{D} - \mathbf{D}_\infty) \geqslant 0$$

This result applies for any positive subsampling ratio r, which may be set arbitrarily high. However, selecting a reasonable ratio remains important for performance, as we will discuss in Chapter 6.

Proposition 5.1 is a corollary of a stronger result on SAMM algorithms. As it provides insights on the convergence mechanisms, we formalize this result in the following.

5.3.2 *Basic assumptions and results on SMM convergence*

We first recall the main results on stochastic majorization minimization algorithms, established in Mairal (2013b), under assumptions

that we slightly tighten for our purpose. In our setting, we consider the empirical risk minimization problem

$$\min_{\theta\in\Theta}\Big(\bar{f}(\theta)\triangleq\frac{1}{n}\sum_{i=1}^{n}f(\theta,x^{(i)})\Big),\tag{5.3}$$

where $f:\mathbb{R}^K\times\mathcal{X}\to\mathbb{R}$ is a loss function and

(C) $\Theta\subset\mathbb{R}^K$ and the support $\mathcal{X}$ of the data are compact.

This is a special case of (4.4) where the samples $(x_t)_t$ are drawn uniformly from the set $\{x^{(i)}\}_i$. The loss functions $f_t\triangleq f(\cdot,x_t)$ defined on $\mathbb{R}^K$ can be non-convex. We instead assume that they meet reasonable regularity conditions:

(D) $(f_t)_t$ is uniformly R-Lipschitz continuous on $\mathbb{R}^K$ and uniformly bounded on Θ.

(E) The directional derivatives $\nabla f_t(\theta,\theta'-\theta)$, as defined by Borwein and Lewis (2010) and $\nabla\bar{f}(\theta,\theta'-\theta)$ exist for all θ and θ' in $\mathbb{R}^K$.

Assumption **(E)** allows to characterize the *stationary points* of problem (5.3), namely $\theta\in\Theta$ such that $\nabla\bar{f}(\theta,\theta'-\theta)\geqslant 0$ for all $\theta'\in\Theta$ — intuitively a point is stationary when there is no local direction in which the objective can be improved.

Let us now recall the definition of first-order surrogate functions used in the SMM algorithm. $(g_t)_t$ are selected in the set $\mathcal{S}_{\rho,L}(f_t,\theta_{t-1})$, hereby introduced.

Definition 5.1 (First-order surrogate function)**.** *Given a function* $f:\mathbb{R}^K\to\mathbb{R}$, $\theta\in\Theta$ *and* $\rho,L>0$, *we define* $\mathcal{S}_{\rho,L}(f,\theta)$ *as the set of functions* $g:\mathbb{R}^K\to\mathbb{R}$ *such that*

- g *is majorizing* f *on* Θ *and* g *is* ρ*-strongly convex,*
- g *and* f *are tight at* θ — *i.e.,* $g(\theta)=f(\theta)$, $g-f$ *is differentiable,* $\nabla(g-f)$ *is* L*-Lipschitz,* $\nabla(g-f)(\theta)=0$, *where* ∇ *is the classical differential operator.*

In OMF, g_t defined in (5.2) is a *variational* surrogate[1] of f_t. We refer the reader to Mairal (2013a) for further examples of first-order surrogates. We also ensure that $\bar{g}_t$ should be *parametrized* and thus representable in memory. The following assumption is met in OMF, as $\bar{g}_t$ is parametrized by the matrices $\bar{C}_t$ and $\bar{B}_t$.

(F) *Parametrized surrogates.* The surrogates $(\bar{g}_t)_t$ are parametrized by vectors in a compact set $\mathcal{K}\subset\mathbb{R}^P$. Namely, for all $t>0$, there exists $\kappa_t\in\mathcal{K}$ such that $\bar{g}_t$ is unequivocally defined as $g_t\triangleq\bar{g}_{\kappa_t}$.

Finally, we ensure that the weights $(w_t)_t$ used in Algorithm 4 decrease at certain rates, slightly less stringent than our assumption **(B)**.

(G) There exists $u\in(\frac{3}{4},1)$ such that $w_t=t^{-u}$.

1. In this case as in SOMF, g_t is not ρ-strongly convex but $\bar{g}_t$ is, thanks to assumption **(A)**. This is sufficient in the proofs of convergence.

When $(\theta_t)_t$ is the sequence yielded by Algorithm 4, the following result (Proposition 3.4 in Mairal, 2013b) establishes the convergence of $(\bar{f}(\theta_t))_t$ and states that θ_t is asymptotically a stationary point of the finite sum problem (5.3), as a special case of the *expected* risk minimization problem (4.4).

Proposition 5.2 (Convergence of SMM, from Mairal, 2013b)**.** *Under assumptions* ***(C)*** *–* ***(G)****,* $(\bar{f}(\theta_t))_{t\geqslant 1}$ *converges with probability one. Every limit point* θ_∞ *of* $(\theta_t)_t$ *is a stationary point of the risk* $\bar{f}$ *defined in* (5.3)*. That is,*

$$\forall \theta \in \Theta, \quad \nabla \bar{f}(\theta_\infty, \theta - \theta_\infty) \geqslant 0.$$

The correctness of the online matrix factorization algorithm can be deduced from this proposition.

5.3.3 *Convergence of SAMM*

We now introduce assumptions on the approximations made in SAMM, before extending the result of Proposition 5.2. We make hypotheses on both the surrogate computation (*majorization*) step and the iterate update (*minimization*) step. The principles of SAMM are illustrated in Figure 5.1, which provides a geometric interpretation of the approximations introduced in the following assumptions **(H)** and **(I)**.

5.3.3.1 *Approximate surrogate computation*

The SMM algorithm selects a surrogate for f_t at point θ_{t-1} within the set $\mathcal{S}_{\rho,L}(f_t, \theta_{t-1})$. Surrogates within this set are *tight* at θ_{t-1} and greater than f_t everywhere. In SAMM, we allow the use of surrogates that are only *approximately majorizing* f_t and *approximately tight* at θ_{t-1}. This is indeed what SOMF does when using estimators in the code computation step. For that purpose, we introduce the set $\mathcal{T}_{\rho,L}(f, \theta, \epsilon)$, that contains all functions ϵ-close of a surrogate in $\mathcal{S}_{\rho,L}(f, \theta)$ for the ℓ_∞-norm:

Definition 5.2 (Approximate first-order surrogate function)**.** *Given a function* $f : \mathbb{R}^K \to \mathbb{R}$*,* $\theta \in \Theta$ *and* $\epsilon > 0$*,* $\mathcal{T}_{\rho,L}(f, \theta, \epsilon)$ *is the set of* ρ*-strongly convex functions* $g : \mathbb{R}^K \to \mathbb{R}$ *such that*

- g *is* ϵ*-majorizing* f *on* Θ*:* $\forall\, \kappa \in \Theta$*,* $g(\kappa) - f(\kappa) \geqslant -\epsilon$*,*
- g *and* f *are* ϵ*-tight at* θ *— i.e.,* $g(\theta) - f(\theta) \leqslant \epsilon$*,* $g - f$ *is differentiable,* $\nabla(g - f)$ *is* L*-lipschitz.*

We assume that SAMM selects an approximative surrogate in the set $\mathcal{T}_{\rho,L}(f_t, \theta_{t-1}, \epsilon_t)$ at each iteration, where $(\epsilon_t)_t$ is a deterministic or random non-negative sequence that vanishes at a sufficient rate.

(H) For all $t > 0$, there exists $\epsilon_t > 0$ such that $g_t \in \mathcal{T}_{\rho,L}(f_t, \theta_{t-1}, \epsilon_t)$. There exists a constant $\eta > 0$ such that $\mathbb{E}[\epsilon_t] \in \mathcal{O}(t^{2(u-1)-\eta})$ and $\epsilon_t \to_\infty 0$ almost surely.

As illustrated in Fig. 5.1, given the OMF surrogate $g_t^\star \in \mathcal{S}_{\rho,L}(f_t, \theta_{t-1})$ defined in (5.2), any function g_t such that $\|g_t - g_t^\star\|_\infty < \epsilon$ is in $\mathcal{T}_{\rho,L}(f_t, \theta_{t-1}, \epsilon)$ — *e.g.*, where g_t uses an approximate $\boldsymbol{\alpha}_t$ in (5.2). This assumption can also be met in matrix factorization settings with difficult code regularizations, that *require* to make code approximations.

5.3.3.2 *Approximate surrogate minimization*

We do not require θ_t to be the minimizer of $\bar{g}_t$ any longer, but ensure that the surrogate objective function $\bar{g}_t$ decreases "fast enough". Namely, θ_t obtained from partial minimization should be closer to a minimizer of $\bar{g}_t$ than θ_{t-1}. We write $(\mathcal{F}_t)_t$ and $(\mathcal{F}_{t-\frac{1}{2}})_t$ the filtrations induced by the past of the algorithm, respectively up to the end of iteration t and up to the beginning of the minimization step in iteration t. Then, we assume

(I) For all $t > 0$, $\bar{g}_t(\theta_t) < \bar{g}_t(\theta_{t-1})$. There exists $\mu > 0$ such that, for all $t > 0$, where $\theta_t^\star = \operatorname{argmin}_{\theta\in\Theta} \bar{g}_t(\theta)$,

$$\mathbb{E}\left[\bar{g}_t(\theta_t) - \bar{g}_t(\theta_t^\star) | \mathcal{F}_{t-\frac{1}{2}}\right] \leqslant (1-\mu)(\bar{g}_t(\theta_{t-1}) - \bar{g}_t(\theta_t^\star)). \tag{5.4}$$

Assumption **(I)** is met by choosing an appropriate method for the inner $\bar{g}_t$ minimization step — a large variety of gradient-descent algorithms indeed have convergence rates of the form (5.4). In SOMF, the block coordinate descent with frozen coordinates indeed meet this property, relying on results from Wright (2015). When both assumptions are met, SAMM enjoys the same convergence guarantees as SMM.

5.3.3.3 *Asymptotic convergence guarantee*

The following proposition guarantees that the stationary point condition of Proposition 5.2 holds for the SAMM algorithm, despite the use of approximate surrogates and approximate minimization.

Proposition 5.3 (Convergence of SAMM)**.** *Under assumptions **(G)** – **(F)**, the conclusion of Proposition 5.2 holds for SAMM.*

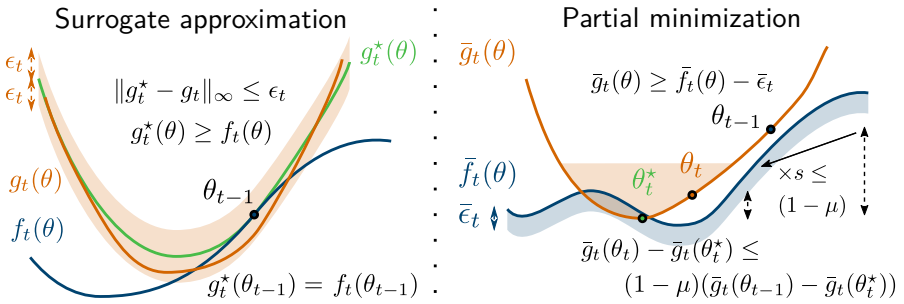


Figure 5.1 – Both steps of SAMM make well-behaved approximations. The operations that are performed in exact SMM are in green and superscripted by *, while the actual computed values are in orange. Light bands recall the bounds on approximations assumed in **(H)** and **(I)**.

Assumption **(H)** is essential to bound the errors introduced by the sequence $(\epsilon_t)_t$ in the proof of Proposition 5.3, while **(I)** is the key element to show that the sequence of iterates $(\theta_t)_t$ is stable enough to ensure convergence. The result holds for any subsampling ratio r, provided that **(A)** remains true. Full proofs are provided in Section A.1.2.

5.3.3.4 *Proving SOMF convergence*

Assumptions **(A)** and **(B)** readily implies **(C)**–**(G)**. With Proposition 5.3 at hand, proving Proposition 5.1 reduces to ensure that the surrogate sequence of SOMF meets **(H)** while its iterate sequence meets assumption **(I)**. Full proofs are provided in Section A.1.3.

5.3.4 *Discussion and variants*

The SOMF algorithm relies on a somewhat complicated assumption **(B)** on the learning weights $(w_t)_t$ and on a reduction mechanism that we further discuss.

WEIGHT DECAY. The original OMF algorithm is provably convergent for $w_t = t^{-u}$, with $u \in (\frac{3}{4}, 1]$. Decreasing u below 1 allows the algorithm to forget about past iterates and may slightly increase convergence speed. This assumption **(G)** is already stronger than the one required by SGD for convex objectives (Bottou, 1999) and online expectation-minimization for exponential families (Cappé and Moulines, 2009), for which we may set $u \in (\frac{1}{2}, 1]$. The convergence of SOMF relies on a variance reduction mechanism, and therefore demands a slightly more stringent condition that arises from the proofs: $u \in (\frac{11}{12}, 1)$, as stated in assumption **(B)**. This may be informally understood as follow: controlling the variance induced by the stochastic subsampling mechanism requires *not to forget about the past too quickly*.

REDUCING THE SUBSAMPLING RATIO. Instead of using the variance reduction mechanism formalised in equation (b) and (c), we may simply gradually reduce the subsampling ratio $r = \frac{p}{q}$, so as to meet assumption **(H)**. It if enough to define a sequence of subsampling sizes $(q_t)_t \in [1, p]$ so that

$$1 - \frac{q_t}{p} \in o(t^{(2(u-1)-\eta)}), \tag{5.5}$$

with $\eta > 0$ and $u \in (\frac{3}{4}, 1]$ such that $w_t = \frac{1}{t^u}$ for all $t > 0$. We then perform the simpler update (a), namely solve the *masked* elastic-net/ridge problem (4.13), recalled here:

$$\boldsymbol{\alpha}_t = \min_{\boldsymbol{\alpha} \in \mathbb{R}^k} \frac{1}{2} \|\mathbf{M}_t(\mathbf{x}_t - \mathbf{D}_{t-1}\boldsymbol{\alpha})\|_2^2 + \Omega(\boldsymbol{\alpha}),$$

where $\mathbf{M}_t$ is a random Bernouilli matrix with parameter $\frac{q_t}{p}$. Of course, (5.5) assumes that $q_t \to p$, *i.e.* that subsampling is no longer used asymptotically. The convergence of the resulting algorithm may

therefore appears less surprising than the guarantees obtained on SOMF with updates (b) and (c). However, (5.5) tells us that q_t may increase *slowly*: typically, we may choose $q_t = p - q_1/m_t^{2(u-1)+\eta}$, where m_t is the number of the current epoch and q_0 is the subsampling size at the first epoch. For $u = 1$, q_t can even remain approximately constant, as $q_t = p - q_1/m_t^{\eta}$ is valid for any positive η. This further justifies the use of the update (4.13).

5.4 CONCLUSION

We have established the convergence of SOMF and provided modular properties to analyse algorithms that perform perturbed stochastic majorization-minimization. Due to the non-convexity of the problem, our analysis does not provide rates of convergence in Proposition 5.1 and 5.3. This calls for a strong empirical validation of the method. We present it in Chapter 6, to which it is possible to move directly.

6

SUBSAMPLED ONLINE MATRIX FACTORIZATION IN PRACTICE

In this chapter, we experiment the performance of SOMF algorithm on various problems (dictionary learning, sparse component analysis, non-negative matrix factorization) and various datasets (two datasets of fMRI and one dataset from hyperspectral imaging). We demonstrate the usefulness of subsampling, and of the various details of the SOMF algorithm. We show quantitatively and qualitatively that the speed-up provided by SOMF makes huge matrix factorization amenable to practitioners. Finally, we consider an extension of the SOMF algorithm, that makes it usable for matrix completion. We assess the performance of this extension on explicit collaborative filtering.

6.1 EXPERIMENTS WITH SOMF

The SOMF algorithm is designed for datasets with large number of samples n and large dimensionality p. Indeed, as detailed in Section 4.5, subsampling removes the computational bottlenecks that arise from high dimensionality. Proposition 5.1 of Chapter 5 establishes that the subsampling used in SOMF is safe, as it enjoys the same guarantees as OMF. However, as with OMF, no convergence rate is provided. We therefore perform a strong empirical validation of subsampling.

We tackle two different problems, in functional Magnetic Resonance Imaging (fMRI) and hyperspectral imaging. Both involve the factorization of very large matrices $\mathbf{X}$ with sparse factors. As the data we consider are huge, subsampling reduces the time of a single iteration by a factor close to $\frac{p}{q}$. Yet it is also much redundant: SOMF makes little approximations and accessing only a fraction of the features per iteration should not hinder much the refinement of the dictionary. Hence high speed-ups are expected — and indeed obtained. All experiments can be reproduced using open-source code.

6.1.1 *Problems and datasets*

6.1.1.1 *Functional MRI*

As discussed in Chapter 3, matrix factorization has long been used in fMRI, since the seminal work of McKeown et al. (1998). Data are temporal series of 3D images of brain activity and are decomposed into spatial modes capturing regions that activate synchronously. They form a matrix $\mathbf{X}$ where columns are the 3D images, and rows corresponds to voxels. Interesting dictionaries for neuroimaging capture spatially-localized components, with a few brain regions. This can

be obtained by enforcing sparsity on the dictionary: we use an ℓ_2 penalty and the elastic-net constraint. SOMF streams subsampled 3D brain records to learn the sparse dictionary $\mathbf{D}$. Data can be huge: we use the whole HCP500 release (Van Essen et al., 2012), with $n = 2.4 \cdot 10^6$ (2000 records, 1 200 time points) and $p = 2 \cdot 10^5$, totaling 2 TB of dense data. For comparison, we also use a smaller public dataset (ADHD200, M. P. P. D. Milham et al., 2012) with 40 records, $n = 7000$ samples and $p = 6 \cdot 10^4$ voxels. Importantly, we seek a *low-rank* factorization, to keep the decomposition interpretable — $k = 70 \ll p$.

6.1.1.2 *Hyperspectral imaging*

Hyperspectral cameras acquire images with many channels that correspond to different spectral bands. They are used heavily in remote sensing (satellite imaging), and material study (microscopic imaging). They yield digital images with around 1 million pixels, each associated with hundreds of spectral channels. Sparse matrix factorization has been widely used on these data for image classification (Chen et al., 2011; Soltani-Farani et al., 2015) and denoising (Maggioni et al., 2013; Peng et al., 2014). All methods rely on the extraction of full-band patches representing a local image neighborhood with all channels included. These patches are very high dimensional, due to the number of spectral bands. From one image of the Aviris project (Vane, 1987), we extract $n = 2 \cdot 10^6$ patches of size 16×16 with 224 channels, hence $p = 6 \cdot 10^4$. A dense dictionary is learned from these patches. It should allow a sparse representation of samples: we either use the classical dictionary learning setting (ℓ_1/elastic-net penalty), or further add positive constraints to the dictionary and codes: both methods may be used and deserved to be benchmarked. We seek a dictionary of reasonable size: we use $k \sim 256 \ll p$.

6.1.2 *Experimental design*

To validate the introduction of subsampling and the usefulness of SOMF, we perform two major experiments.

— We measure the performance of SOMF when increasing the reduction factor, and show benefits of stochastic dimension reduction on all datasets.

— We assess the importance of subsampling in each of the steps of SOMF. We compare the different approaches proposed for code computation.

VALIDATION. We compute the objective function (4.3) over a test set to rule out any overfitting effect — a dictionary should be a good representation of unseen samples. This criterion is always plotted against wall-clock time, as we are interested in the performance of SOMF for practitioners.

TOOLS. To perform a valid benchmark, we implement OMF and SOMF using Cython (Behnel et al., 2011). We use coordinate descent (Fried-

Table 6.1 – Summary of experimental settings

Field	Functional MRI		Hyperspectral imaging
Dataset	ADHD	HCP	Patches from Aviris
Factors	**D** sparse, **A** dense		**D** dense, **A** sparse
# samples n	$7 \cdot 10^3$	$2 \cdot 10^6$	$2 \cdot 10^6$
# features p	$6 \cdot 10^4$	$2 \cdot 10^5$	$6 \cdot 10^4$
X size	2 GB	2 TB	103 GB
Use case ex.	Extracting predictive feature		Recognition / denoising

Table 6.2 – CPU time to reach convergence (< 1% test objective)

Dataset	ADHD		Aviris (NMF)		Aviris (DL)		HCP	
Algorithm	OMF	SOMF	OMF	SOMF	OMF	SOMF	OMF	SOMF
Conv. time	6 min	**28 s**	2 h 30	**43 min**	1 h 16	**11 min**	3 h 50	**17 min**
Speed-up	11.8		3.36		6.80		13.31	

man et al., 2007) to solve Lasso problems with optional positivity constraints. Code computation is parallelized to handle mini-batches. Experiments use *scikit-learn* (Pedregosa et al., 2011) for numerics, and *nilearn* (Abraham et al., 2014) for handling fMRI data. We have released the code in an open-source Python package. Experiments were run on 3 cores of an Intel Xeon 2.6GHz, in which case computing $\mathbf{P}_t^{\perp}\bar{\mathbf{B}}_t$ is faster than updating $\mathbf{P}_t\mathbf{D}_t$.

PARAMETER SETTING. Setting the number of components k and the amount of regularization λ is a hard problem in the absence of ground truth. Those are typically set by cross-validation when matrix factorization is part of a supervised pipeline. For fMRI, we set $k = 70$ to obtain interpretable networks, and set λ so that the decomposition approximately covers the whole brain (i.e., every map is $\frac{k}{70}$) sparse). For hyperspectral images, we set $k = 256$ and select λ to obtain a dictionary on which codes are around 3% sparse. We cycle randomly through the data (fMRI records, image patches) until convergence, using mini-batches of size $\eta = 200$ for HCP and Aviris, and $\eta = 50$ for ADHD (small number of samples). Hyperspectral patches are normalized in the dictionary learning setting, but not in the non-negative setting — the classical pre-conditioning for each case. We use $u = 0.917$ and $v = 0.751$ for weight sequences.

6.1.3 *Reduction brings speed-up at all data scales*

We benchmark SOMF for various reduction factors against the original online matrix factorization algorithm OMF Mairal et al. (2010), on the three presented datasets. We stream data in the same order for all reduction factors. Using variant (c) (true Gram matrix, averaged $\boldsymbol{\beta}_t$) performs slightly better on fMRI datasets, whereas (b) (averaged

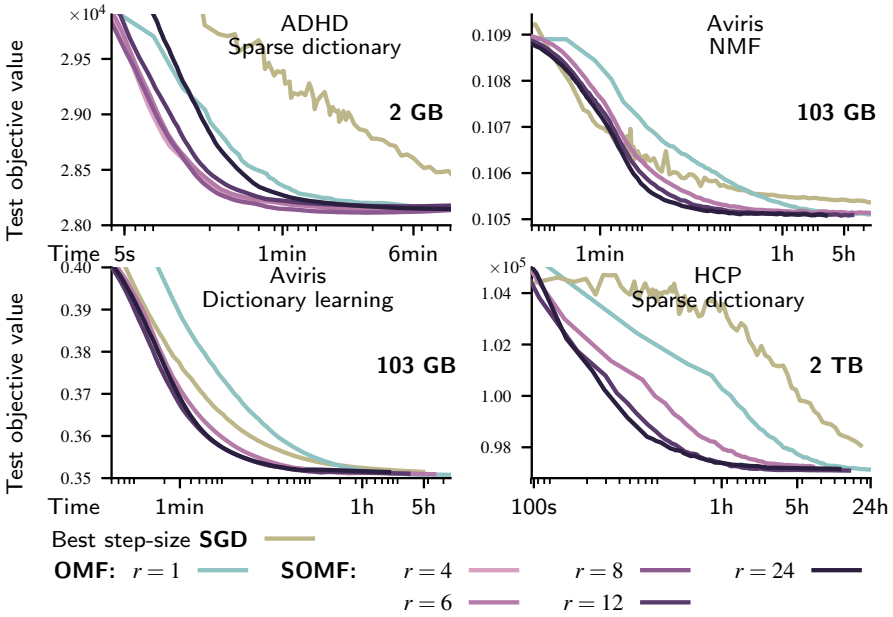


Figure 6.1 – Subsampling provides significant speed-ups on all fMRI and hyperspectral datasets. A reduction factor of 12 is a good overall choice. With larger data, larger reduction factors can be used for better performance — convergence is reached 13× faster than state-of-the-art methods on the 2TB HCP dataset.

Gram matrix and β_t) is slightly faster for hyperspectral decomposition. For comparison purpose, we display results using estimators (b) only.

6.1.3.1 *Benchmarking SOMF*

Figure 6.1 plots the test objective against CPU time. First, we observe that all algorithms find dictionaries with very close objective function values for all reduction factors, on each dataset. This is not a trivial observation as the matrix factorization problem (4.3) is not convex and different runs of OMF and SOMF may converge towards minima with different values. Second, and most importantly, SOMF provides significant improvements in convergence speed for three different sizes of data and three different factorization settings. Both observations confirm the relevance of the subsampling approach. Quantitatively, we summarize the speed-ups obtained in Table 6.2. On fMRI data, on both large and medium datasets, SOMF provides more than an order of magnitude speed-up. Practitioners working on datasets akin to HCP can decompose their data in 20 minutes instead of 4 h previously, while working on a single machine. We obtain the highest speed-ups for the largest dataset — accounting for the extra redundancy that usually appears when dataset size increase. Up to $r \sim 8$, speed-up is of the order of r — subsampling induces little noise in the iterate sequence, compared to OMF. Hyperspectral decomposition is performed near 7× faster than with OMF in the classical dictionary learning setting, and 3× in the non-negative setting, which further demonstrates the versatility of SOMF.

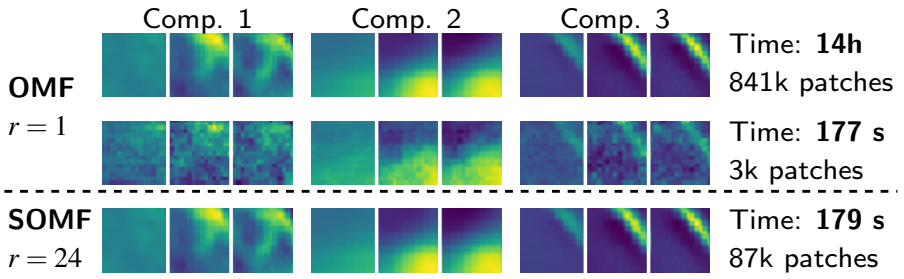


Figure 6.2 – Given a 3 minute time budget, the atoms learned by SOMF are more focal and less noisy that those learned by OMF. They are closer to the dictionary of first line, for which convergence has been reached.

6.1.3.2 *Comparison with stochastic gradient descent*

It is possible to solve (4.3) using projected stochastic gradient descent (SGD, Duchi and Singer (2009)). We use the gradient of $\mathbf{D} \to f_t(\mathbf{D})$ evaluated in $\mathbf{D}_{t-1}$ to compute $\mathbf{D}_t$ with fixed step-size η:

$$\nabla_D f_t(\mathbf{D}_{t-1}) = (\mathbf{x}_t - (\mathbf{D}_{t-1}\boldsymbol{\alpha}_t^\star)\boldsymbol{\alpha}_t^\star,$$
$$\mathbf{D}_t \leftarrow \mathbf{D}_{t-1} - \eta\nabla_D f_t(\mathbf{D}_{t-1})$$

where $\boldsymbol{\alpha}_t^\star$ is defined in (4.5). Computation of $\nabla_D f_t(\mathbf{D}_{t-1})$ is derived from Danskin theorem (1966). Its form when $\boldsymbol{\alpha}_t^\star$ is the solution of a Lasso regression is due to Mairal et al. (2009).

On all tested settings, for high precision convergence, SGD (with the best step-size among a grid) is slower than OMF and even slower than SOMF. In the dictionary learning setting, SGD is somewhat faster than OMF but slower than SOMF in the first epochs. SGD further requires to select the step-size by grid search. In contrast, SOMF and OMF performance little depends on the parameters u and v, and do not require hyper-parameter search for solver parameters.

6.1.3.3 *Qualitative results on dictionaries*

Qualitatively, given a certain time budget, we compare the dictionaries obtained on different datasets.

HYPERSPECTRAL IMAGES. Figure 6.2 compares the results of OMF and the results of SOMF with a subsampling ratio $r = 24$, in the non-negative setting. Our algorithm yields a valid smooth bank of filters much faster. [9]

FUNCTIONAL MRI. Figure 6.3 shows that with the same time budget, the proposed reduction approach with $r = 12$ on half of HCP data (500 subjects in this experiment) gives better results than processing a small fraction of the data without reduction: segmented regions are less noisy and closer to processing the full data. Practitioners are thus able to derive a usable dictionary from one of the largest fMRI dataset available in less than half a day. This was out-of-reach using existing techniques.

[9] *As expected, OMF and SOMF does not output the same final dictionaries, but some atoms remains very close along the updates of both algorithms, when streaming data in the same order. We select such atoms in Fig 6.2.*

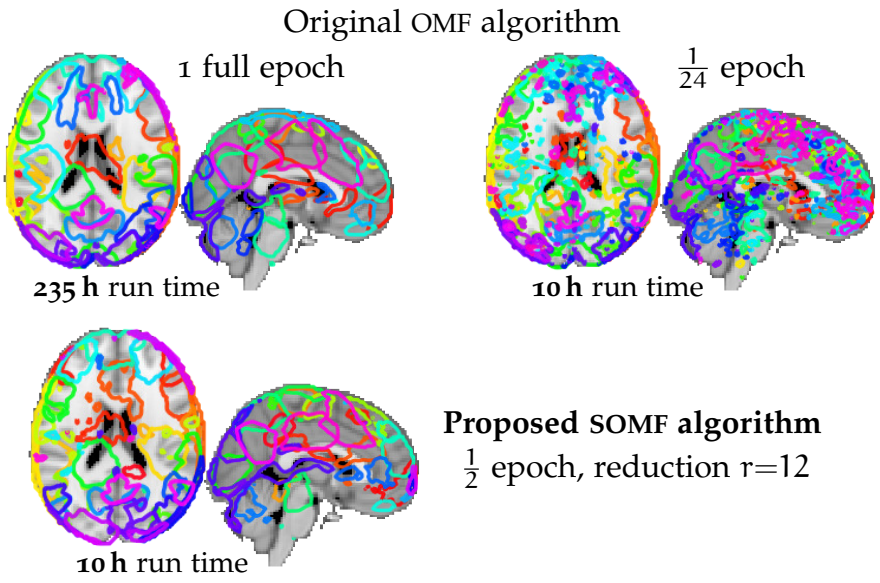


Figure 6.3 – **Brain atlases**: outlines of each map at half the maximum value ($\lambda = 10^{-4}$). **Top left**: the reference OMF algorithm on the full dataset. **Top right**: the reference algorithm on a twentieth of the dataset. **Bottom**: the proposed SOMF algorithm with a similar run time: half the dataset and $r = 12$. Compared to a full run of the baseline algorithm, the figure explore two possible strategies to decrease computation time: processing less data (top right), or our approach (bottom). Our approach achieves a result closer to the gold standard in a given time budget.

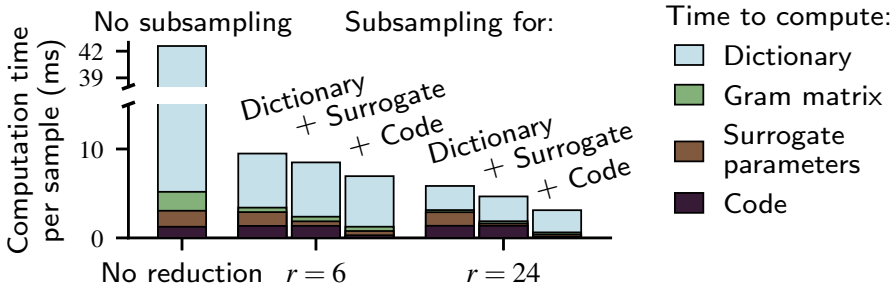


Figure 6.4 – Profiling OMF and SOMF for HCP decomposition. Partial dictionary update removes the major bottleneck of online matrix factorization for small reductions. For higher reduction, parameter update and code computation must be subsampled to further reduce the iteration time.

6.1.3.4 *Finding the right subsampling ratio*

Table 6.2 reports convergence time within 1%, which is enough for application in practice. SOMF is less beneficial when setting very high precision: for convergence within 0.01%, speed-up for HCP is 3.4. This is expected as SOMF trades speed for approximation. For high precision convergence, the reduction ratio can be reduced after a few epochs. As expected, there exists an *optimal* reduction ratio, depending on the problem and precision, beyond which performance reduces: $r = 12$ yields better results than $r = 24$ on Aviris (dictionary learning) and ADHD (sparse components), for 1% precision.

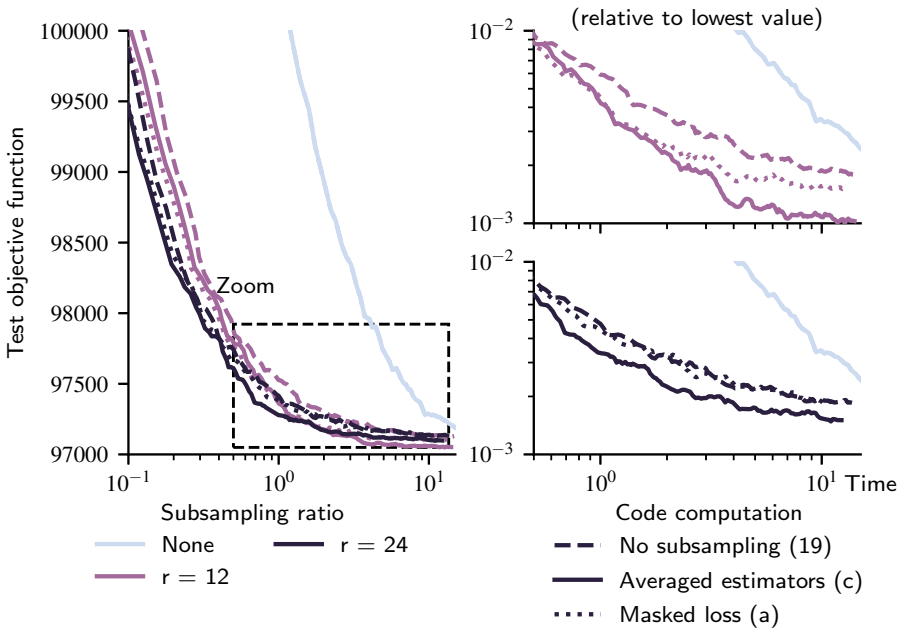


Figure 6.5 – Approximating code computation with the proposed subsampling method further accelerates the convergence of SOMF. Refining code computation using past iterations (averaged estimates) performs better than simply performing a subsampled linear regression.

Our first experiment establishes the power of stochastic subsampling as a whole. In the following two experiments, we refine our analysis to show that subsampling is indeed useful in the three steps of online matrix factorization.

6.1.4 *For each step of SOMF, subsampling removes a bottleneck*

In Section 4.4, we have provided theoretical guidelines on when to introduce subsampling in each of the three steps of an iteration of SOMF. This analysis predicts that, for $\eta \sim k$, we should first use partial dictionary update, before using approximate code computation and asynchronous parameter aggregation. We verify this by measuring the time spent by SOMF on each of the updates for various reduction factors, on the HCP dataset. Results are presented in Figure 6.4. We observe that block coordinate descent is indeed the bottleneck in OMF. Introducing partial dictionary update removes this bottleneck, and as the reduction factor increases, code computation and surrogate aggregation becomes the major bottlenecks. Introducing subsampling as described in SOMF overcomes these bottlenecks, which rationalizes all steps of SOMF from a computational point of view.

6.1.5 *Code subsampling is useful for high reduction*

It remains to assess the performance of approximate code computation and averaging techniques used in SOMF. Indeed, subsampling for code computation introduces noise that may undermine the com-

putational speed-up. To understand the impact of approximate code computation, we compare three strategies to compute $(\alpha_t)_t$ on the HCP dataset. First, we compute $(\alpha_t^\star)_t$ from $(x_t)_t$ using (4.12). Subsampling is thus used only in dictionary update. Second, we rely on masked, non-consistent estimators (a), as in Mensch et al. (2016a) — this breaks convergence guarantees. Third, we use averaged estimators (β_t, G_t) from (c) to reduce the variance in $(\alpha_t)_t$ computation.

Figure 6.5 compares the three strategies for $r \in \{12, 24\}$. Partial minimization at each step is the most important part to accelerate convergence: subsampling the dictionary updates already allows to outperforms OMF. This is expected, as dictionary update constitutes the main bottleneck of OMF in large-scale settings. Yet, for large reduction factors, using subsampling in code computation is important to further accelerate convergence. This clearly appears when comparing the plain and dashed black curves. Using past estimates to better approximate $(\alpha_t)_t$ yields faster convergence than the non-converging, masked loss strategy (a). Note that the latter one remains a good option as it is simpler to implement and almost as efficient.

Before concluding this chapter, we introduce an extension of the SOMF algorithm that allows it to handle missing values. Although the proposed algorithm is not provably convergent, it can be used to perform fast *collaborative filtering*.

6.2 EXTENSION TO MATRIX COMPLETION

SOMF algorithm may be adapted to handle a different kind of matrix factorization problem, known as low-rank matrix completion. In this setting, we have only access to a masked data matrix $M * X$, where $X, M \in \mathbb{R}^{n \times p}$ are the data matrix and masking binary matrix, and $*$ denote the elementwise product between two matrices. Formally, we want to find $D \in \mathbb{R}^{p \times k}$ and $A \in \mathbb{R}^{k \times n}$ so that DA is low-rank and $M * X \approx M * (DA)$. We then predict the unknown values of X as $\hat{X} \triangleq DA$.

One of the best known application of low-rank matrix completion is explicit *collaborative filtering*. In this setting, every user from a pool of n individuals ranks a subset of p items (*e.g.*, movies). We want to gather all the ratings to predict the ratings that each individual would make for each of the p items. This amounts to complete a user-item rating matrix X, of which we only observe the ratings that were provided by the users.

6.2.1 *Problem setting*

Low-rank matrix factorization is traditionally stated as the following empirical minimization problem, known as maximum margin matrix factorisation (Srebro et al., 2004):

$$\min_{\substack{D \in \mathbb{R}^{p \times k} \\ A \in \mathbb{R}^{k \times n}}} \sum_{i=1}^{n} \left\| m^{(i)} * (x^{(i)} - D\alpha^{(i)}) \right\|_2^2 + \frac{\lambda}{2}(\|D\|_2^2 + \|A\|_2^2), \qquad (6.1)$$

where $\mathbf{m}^{(i)}$ is the i-th column of $\mathbf{M}$. The joint ℓ_2 penalty on $\mathbf{D}$ and $\mathbf{A}$ enforce $\mathbf{DA}$ to be low-rank (Fazel et al., 2001). The low property is typically also ensured by hard setting k to be lower than $\min(p, n)$. This setting was successfully used by the winners of the Netflix challenge (R. M. Bell and Koren, 2007).

To adapt SOMF to solve a problem similar to (6.1), we write, for all $i \in [n]$, $\mathbf{M}^{(i)} \triangleq \frac{p}{q^{(i)}}\mathrm{Diag}(\mathbf{m}^{(i)})$, where $q^{(i)} \triangleq \|\mathbf{m}^{(i)}\|_0$ is the number of observed coefficient in sample $\mathbf{x}^{(i)}$. Every mask $\mathbf{M}^{(i)}$ have the same diagonal form as in (4.9), as if it were sampled from a 1-centered Bernouilli distribution of parameter $\frac{q^{(i)}}{p}$. In the matrix completion case, the masks are *provided*: coefficients of $\mathbf{x}^{(i)}$ masked out by $\mathbf{M}^{(i)}$ are *unknown* and can never be accessed during training. We propose to solve the following objective

$$\min_{\mathbf{D}\in\mathcal{C}} \frac{1}{n}\sum_{i=1}^{n} \min_{\boldsymbol{\alpha}\in\mathbb{R}^k} \Big(\frac{1}{2}\big\|\mathbf{M}^{(i)}(\mathbf{x}^{(i)} - \mathbf{D}\boldsymbol{\alpha})\big\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_2^2\Big), \tag{6.2}$$

where $\mathbf{C}$ constrains $\mathbf{D}$ atoms to live in ℓ_2 unit balls. Objective (6.2) is highly similar to (6.1), expect for the penalty on $\mathbf{D}$ that has been replaced by a constraint. Importantly, it focuses on the left-side factor $\mathbf{D}$ only, as does the matrix factorization objective (4.3) central to this chapter. This will allow to reuse the principles of SOMF.

RELATED WORK. Szabó et al. (2011) proposed an algorithm similar to OMF to solve an objective akin to (6.2). Unfortunately, the single iteration complexity of their algorithm is proportional to p and not to the effective size $q^{(i)}$ of sample $\mathbf{x}_t = \mathbf{x}^{(i)}$. This makes it unusable for large-scale matrices with few non-zero coefficients. In contrast, the algorithm we now propose achieves the appropriate complexity.

6.2.2 *Proposed algorithm*

To solve (6.2), we propose the following algorithm. At iteration t, we sample $\mathbf{x}_t = \mathbf{x}^{(i)}$ and $\mathbf{M}_t = \mathbf{M}^{(i)}$, and

— Compute $\boldsymbol{\alpha}_t$ solving (4.13), *i. e.*

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

— Update $\bar{\mathbf{C}}_t$ as in SOMF and $\bar{\mathbf{B}}_t$ using the partial update equation (4.19), *i. e.*

$$\begin{aligned}\mathbf{P}_t^\perp\bar{\mathbf{B}}_t &\triangleq \mathbf{P}_t^\perp\bar{\mathbf{B}}_{t-1}\\ \mathbf{P}_t\bar{\mathbf{B}}_t &\triangleq (1 - \frac{p}{q^{(i)}}w_t)\mathbf{P}_t\bar{\mathbf{B}}_{t-1} + \frac{p}{q^{(i)}}w_t\mathbf{P}_t\mathbf{x}_t\boldsymbol{\alpha}_t^\top,\end{aligned}$$

— Update $\mathbf{D}$ to solve the "frozen" problem (4.15), *i. e.* changes the coefficients of $\mathbf{P}_t\mathbf{D}$ only:

$$\mathbf{D}_t \triangleq \operatorname*{argmin}_{\substack{\mathbf{D}\in\mathcal{C}\\ \mathbf{P}_t^\perp\mathbf{D}=\mathbf{P}_t^\perp\mathbf{D}_{t-1}}} \frac{1}{2}\mathrm{Tr}\,(\mathbf{D}^\top\mathbf{D}\bar{\mathbf{C}}_t) - \mathrm{Tr}\,(\mathbf{D}^\top\bar{\mathbf{B}}_t).$$

Compared to SOMF algorithm, two aspects have changed. First, we do not use the mechanisms (b) or (c) to reduce the variance that stochastic subsampling introduces. This is no longer possible as we always look at sample $x^{(i)}$ with the same mask $M^{(i)}$. Secondly, for the same reason, we are not able to update the full statistic $\bar{B}_t$ at each iteration, as we can only the coefficients gathered in $P_t x_t$. We therefore perform *partial updates* of $\bar{B}_t$, using a scaling coefficient $\frac{p}{q^{(i)}}$ to compensate for the different frequencies at which rows appear in the streaming process.

COMPLETION AT TEST TIME. Past the first epoch, at iteration t, every column i of X can be predicted using the last code $\alpha_t^{(i)} \triangleq \alpha_s$ that was computed for this column, *i.e.* the largest $s \leqslant t$ such that $x_s = x^{(i)}$. At iteration t, for all i in $[n]$, we set $\hat{x}^{(i)} \triangleq D_t \alpha_s$. Prediction thus only requires a single additional matrix computation using the recorded parameters D_t and $A_t \triangleq (\alpha_t^{(i)})_{i \in [n]}$.

6.2.3 *Experiments*

We validate the performance of the proposed algorithm on explicit collaborative filtering.

6.2.3.1 *Setting*

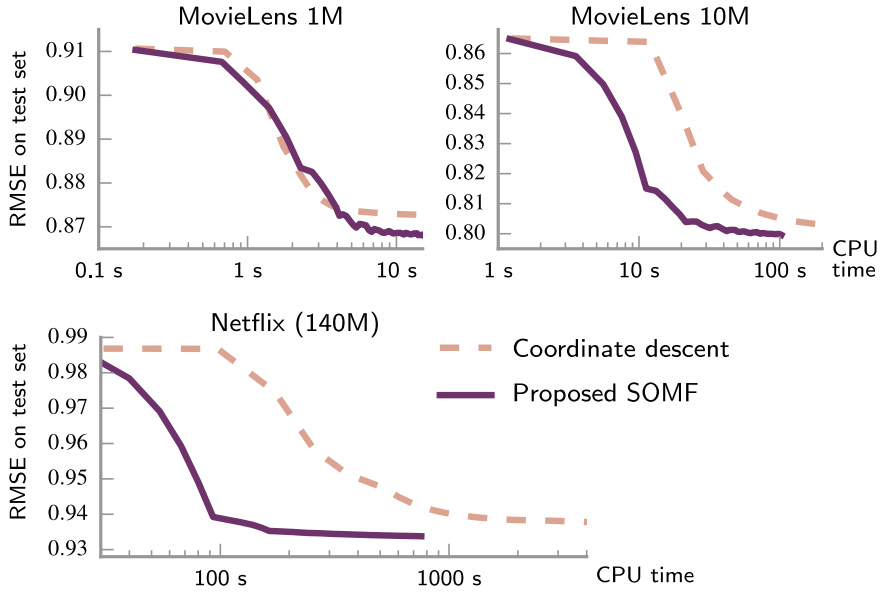


Figure 6.6 – **Learning speed for collaborative filtering** for datasets of different size: the larger the dataset, the greater our speed-up.

We evaluate the scalability of our method on datasets of different dimension: MovieLens 1M, MovieLens 10M, and 140M ratings Netflix dataset. We stream user ratings to our algorithm: p is the number of movies and n is the number of users. As $n \gg p$ on Netflix dataset, this increases the benefit of using an online method. We have observed comparable prediction performance streaming item ratings.

Table 6.3 – **Comparison of performance and convergence time** for online masked matrix factorization and coordinate descent method.

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

BASELINE. We compare our algorithm to a coordinate descent based method (Yu et al., 2012), that provides state-of-the art convergence time performance on our largest dataset. Although stochastic gradient descent methods for matrix factorization can provide slightly better single-run performance (Takács et al., 2009), these are notoriously hard to tune and require a precise grid search to uncover a working schedule of learning rates. In contrast, coordinate descent methods do not require any hyper-parameter setting and are therefore more efficient in practice. We benchmarked various recommender-system codes (*MyMediaLite*, *LibFM*, *SoftImpute*, *spira*), and chose coordinate descent algorithm from *spira* as it was by far the fastest.

PREPROCESSING. Successful prediction should take into account the biases associated to users and items. We compute these biases on train data following Hastie et al. (2015) (alternated debiasing). We use them to center the samples $(x_t)_t$ that are streamed to the algorithm, and to perform final prediction.

DETAILS. Both baseline and proposed algorithm are implemented in a computationally optimal way, enabling fair comparison based on CPU time. Benchmarks were run using a single 2.7 GHz Xeon CPU, with $k = 30$ components in the dictionary. For Movielens datasets, we use a random 25% of data for test and the rest for training. We average results on five train/test split for MovieLens in Table 6.3. On Netflix, the probe dataset is used for testing. Regularization parameter λ is set by cross-validation on the training set: the training data is split 3 times, keeping 33% of Movielens datasets for evaluation and 1% for Netflix, and grid search is performed on 15 values of λ between 10^{-2} and 10. We assess the quality of obtained decomposition by measuring the Root mean square error (RMSE) between prediction on the test set and ground truth. We use mini-batches of size $\frac{n}{100}$.

6.2.3.2 *Performance benchmark*

We report the evolution of test RMSE across time in Figure 6.6. Convergence is virtually achieved, despite the lack of theoretical guarantees. We report test RMSE at convergence and wall-clock convergence time in Table 6.3. Benchmarks are performed on the final run, after selecting the regularization parameter λ.

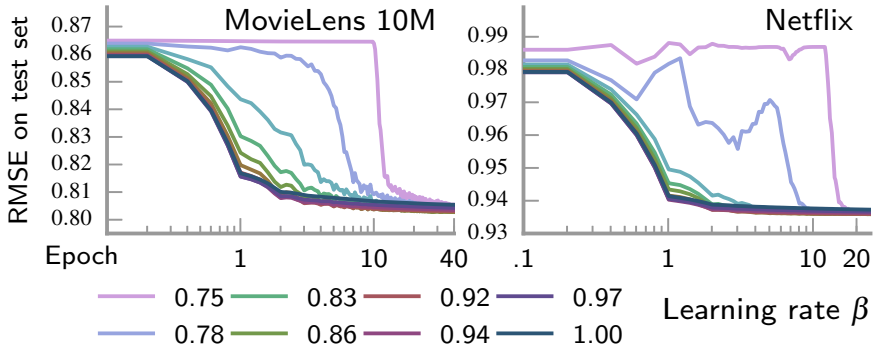


Figure 6.7 – **Learning weights**: on two different datasets, optimal convergence is obtained for $\beta \in [.85,\ .95]$, predicted by theory.

The proposed method converge toward a solution that is at least as good as that of coordinate descent, and slightly better on Movielens 10M and Netflix (140M ratings). Our algorithm brings a substantial performance improvement on medium and large scale datasets. On Netflix, convergence is almost reached in 4 minutes (score under 0.1% deviation from final RMSE), which makes our method 6.8 times faster than coordinate descent. Moreover, the relative performance of our algorithm increases with dataset size. Indeed, as datasets grow, less epochs are needed for our algorithm to reach convergence (Fig. 6.6). This is a significant advantage over coordinate descent, that requires a stable number of cycle on coordinates to reach convergence, regardless of dataset size.

6.2.3.3 *Learning weights*

Unlike SGD, and similar to the vanilla online dictionary learning algorithm, our method does not critically suffer from hyper-parameter tuning. We tried weights $w_t = \frac{1}{t^\kappa}$ as described in Section 4.5.5, and observed that a range of κ yields fast convergence. Theoretically, from **(G)**, κ must be in $(.75, 1]$ to ensure convergence of stochastic majorization minimization algorithms. Although the matrix completion algorithm adapted from SOMF is not provably convergent, we obtain optimal accuracy decrease for $\kappa \in [.85, 0.95]$, as observable in Figure 6.7. We report results for $\beta = 0.91$ in Figure 6.6 and Table 6.3.

6.3 CONCLUSION OF PART II

In the previous three chapters, we introduced SOMF, a matrix factorization algorithm that can handle input data with very large number of rows and columns. It leverages subsampling within the inner loop of a streaming algorithm to make iterations faster and accelerate convergence. We show that SOMF provides a stationary point of the non-convex matrix factorization problem. To prove this result, we extend the stochastic majorization-minimization framework to two major approximations. We assess the performance of SOMF on real-world large-scale problems, with different sparsity/positivity requirements on learned factors. In particular, on fMRI and hyperspectral data de-

composition, we show that the use of subsampling can speed-up decomposition up to 13 times. SOMF may also be adapted to perform explicit collaborative filtering, with very good performance. The larger the dataset, the more SOMF outperforms state-of-the art techniques, which is very promising for future applications. This call for adaptation of our stochastic subsampling approach to learn more complex models.

Part III

DEEPER MODELS FOR MULTI-STUDY COGNITIVE MAPPING

7

LEARNING MULTI-STUDY NEURAL REPRESENTATIONS OF COGNITION FOR COMPREHENSIVE INTER-SUBJECT DECODING

In this chapter, we consider the problem of *inter-subject decoding* in fMRI, and specifically address the following question: can we gather the many publicly available task fMRI studies to learn new cognitive models that are both more interpretable and more accurate than existing methods, that have been shown to be fragile due to the small sample sizes (Button et al., 2013) ?

We build upon the work from Chapter 4–6 to learn functional networks from a large repository of resting-state data. We inject these into *supervised* models that classify statistical maps from many studies into the stimuli/tasks used in these studies. More precisely, we resort to three-layer linear models: the first two layers incorporate information from resting-state data and perform successive dimension reductions; the last layer is constituted of one classification head per study. The dimension reduction of input brain images is *jointly* learned with the many classification tasks, so as to allow *transfer learning* across tasks.

Our multi-study model is thus *deeper* that the usual decoding models, although it remains linear. We show that it performs quantitatively better than the usual models used for decoding, and that it may be used to uncover interesting cognitive networks in the brain. Interpreting statistical learning models with multiple layers is challenging: we propose a new approach based on matrix factorization and ensembling to produce models with interpretable layers from equivalent non-interpretable models.

This chapter is a substantial extension of the work

> Mensch, A., Mairal, J., Bzdok, D., Thirion, B., & Varoquaux, G. (2017). Learning neural representations of human cognition across many fMRI studies. *Advances in Neural Information Processing Systems*,

and has been uploaded as a preprint under the title

> Mensch, A., Mairal, J., Thirion, B., & Varoquaux, G. (2018a). Extracting universal representations of cognition across brain-imaging studies. *arXiv:1809.06035 [stat.ML]*.

It is written with the cognitive neuroscience community as a target — we slightly tone down on mathematical formalism and give a stronger cognitive perspective to our approach.

7.1 INTRODUCTION

Cognitive neuroscience is progressively accumulating records of neural activity responses to specific tasks or stimuli, and the num-

ber of publicly available fMRI records is increasing in two major directions. First, relatively generic task data are being acquired on cohorts of subjects in the thousands (Sudlow et al., 2015; Van Essen et al., 2012), along with a large amount of resting-state data. Secondly, the conclusions obtained on smaller-cohort task studies (with dozens of subjects) are being made available in standard format and common repositories (Poldrack et al., 2013). Within this positive context, the conclusions of Button et al. (2013) brought to the light the low statistical power of analyzing these small studies with standard methods. Since those still form the majority of data in neuro-imaging, this lack of power is a central challenge for cognitive neuroimaging. As stressed by Poldrack et al. (2017), increasing the number of training samples in cognitive inference may therefore be the only way to at last provide strong conclusions on functional localization — and the community should endeavor to work with much larger cohorts.

Yet, large-scale studies are costly and there is little hope to acquire many single-subject records for the multitude of precise psychological processes that are of interest when studying the brain. Despite the richness of their data, *task-broad* fMRI initiatives, also known as deep phenotyping (Nooner et al., 2012; Pinho et al., 2018), are doomed to remain limited in number of subjects, and therefore hard to exploit to produce inter-subject cognitive atlases. This may appear somewhat of a dead-end, *unless* we leverage the aforementioned accumulation of functional data across studies. Combining many large and small-cohort studies to learn common cognitive models would indeed drastically increase their observed evidence.

A major obstacle against this endeavor lies in the heterogeneity of the protocols used to produce statistical maps of brain activity. As stressed by Newell (1973), aggregating knowledge across cognitive neuroscience experiments is intrinsically difficult due to the diverse nature of the hypotheses and conclusions of the investigators. Concretely, every task fMRI study aims at isolating brain effects underlying some study-specific psychological processes. The conclusions it provides are statistical maps that correspond to carefully designed but study-exclusive stimuli, that seldom have any exact counterpart in other studies.

These statistical maps will typically be used to learn inter-subject decoding models, that predict stimuli from statistical maps acquired on *new* subjects (Poldrack et al., 2009). As a modern consequence of Newell's curse, taking advantage of using different studies to learn brain map decoders requires to circumvent the undocumented nature of protocols' relationships, or in other word the absence of known connections between the different cognitive labels we wish to predict.

To address this issue, several works (Koyejo and Poldrack, 2013; Schwartz et al., 2013; T. D. Wager et al., 2013) rely on cognitive ontologies (*e. g.,* Turner and Laird, 2012) to decompose psychological manipulations onto common meaningful cognitive concepts, that they predict using a single model. Although it proved successful in providing well defined region atlases, this approach hardly scales up to the current size of public repositories as it requires a high level

of supervision: a human must classify every condition from every study into a common nomenclature. It is also prone to be biased toward specific understanding of cognition. On the other side of the supervision spectrum, large-scale meta-analysis initiatives (Yarkoni et al., 2011) relates key coordinates extracted from statistical maps to a summarized description of the scientific papers they appear in. Although quantitative meta-analysis techniques provide useful summaries of the existing literature, they are hindered by label noise in the experiment descriptions, and the weak information on brain activation provided by author-selected coordinates (Salimi-Khorshidi et al., 2009).

In this chapter, we show how to learn multi-study decoding models from full statistical maps, without preliminary labelling of any sort. That is, we give up on defining ad-hoc cognitive ontologies, which is a fundamental problem in psychology (Uttal, 2001), and let interesting cognitive directions be extracted from data. For this, we start from the minimal hypothesis that activation maps may be described on atomic basis functions that captures the neural building blocks underlying cognitive processes (Barrett, 2009). Leveraging advances in multi-task learning (Ando and T. Zhang, 2005; Y. Xue et al., 2007) with deep models (*e.g.*, Collobert and Weston, 2008; LeCun et al., 2015), we learn these functions in a fully data-driven way, so that they are fit for decoding every study of our corpus. We argue that our starting point hypothesis and approach constitute a sound direction to overcome the known limitations (Poldrack and Yarkoni, 2016) of single-study cognitive subtraction models. Our model indeed 1) extracts *interpretable* task-optimized spatial networks, that constitute a valid approximation of basic cognitive directions and 2) significantly improves decoding performance for a vast majority of studies, as the information provided by every statistical image helps decoding left-out subject images *across paradigms*.

7.2 RESULTS

We begin with a concise overview of our methodological approach, that will be further described in Section 7.4.

7.2.1 *Method overview*

Our approach of multi-study inter-subject decoding has three major aspects, that we summarize in Figure 7.1. First, as made possible by the increasing availability of public task functional MRI data, we aggregate (Figure 7.1a) statistical maps from many task studies and the BOLD time-series from one or several large resting-state studies, to serve as input to the proposed model. Statistical maps are obtained by standard analysis, computing z-statistics maps for either base conditions or contrasts of interest when those are publicly specified.

We cast inter-subject decoding as a machine-learning classification problem, where models predict the contrast/condition class from an

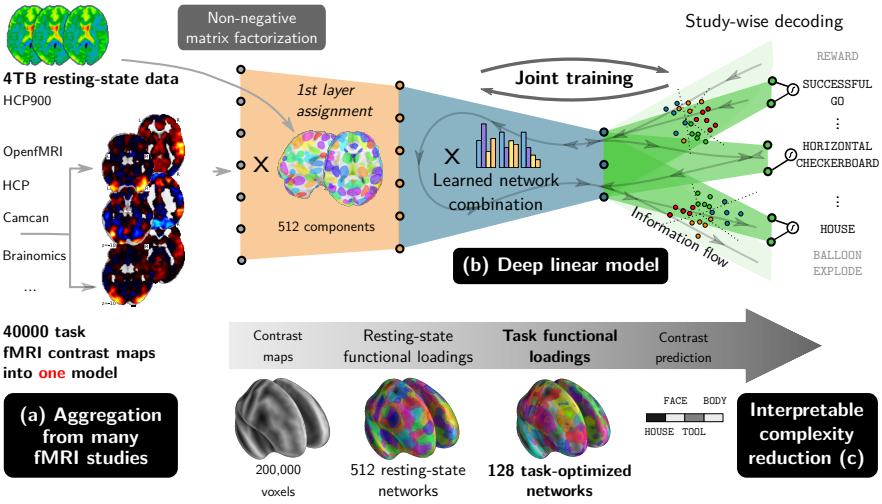
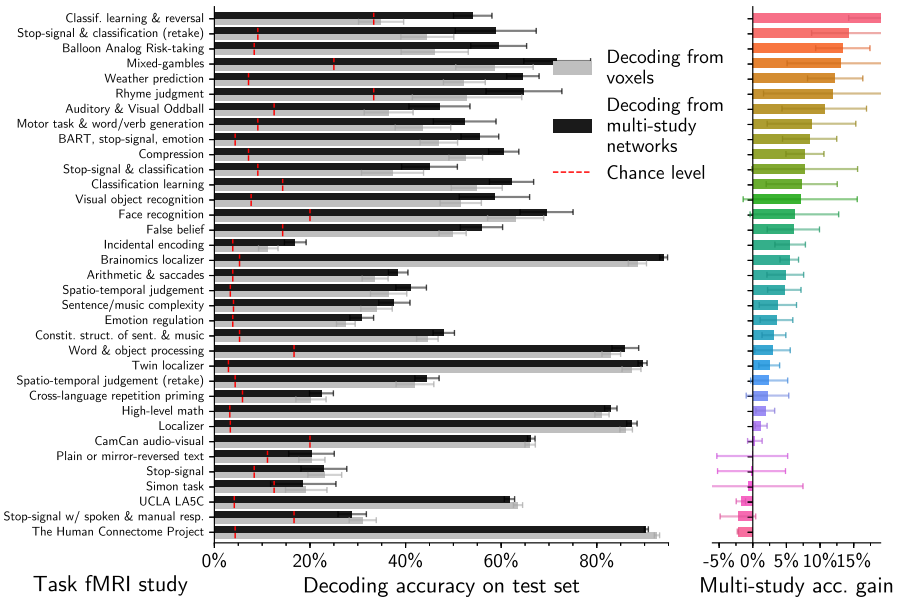


Figure 7.1 – We perform inter-subject decoding using a shared three-layer model trained on multiple studies. An initial layer projects the input images from all studies onto functional networks learned on resting-state data. Then, a second layer combines the functional networks loadings into common meaningful cognitive directions, that are used to perform decoding for each study in a third-layer. The second and third layer are trained jointly, fostering transfer learning across studies.

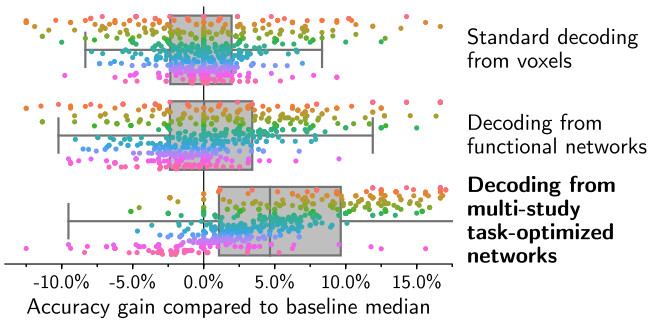
input z-map. The proposed linear classification model features *three* layers of transformation (Figure 7.1b). The first layer projects input z-maps onto functional components (*e. g.*, 512) that are learned from resting-state data. The second layer performs a further dimension reduction (*e. g.*, with 128 output features) and outputs a *common embedding* of all input data; the embedded data from each study are then classified into their respective contrast/condition classes by a third study-specific layer. The second layer and the many classification heads of the third layer are jointly learned using regularized stochastic optimization. Overall, this approach reflects our starting point cognitive hypothesis: cognition may be represented on basic cognitive functions distributed spatially in the brain. On the other hand, we expect that, for all studies, projecting on this basis should improve or at least preserve across-subject predictive accuracy, by removing confounds while keeping intact the cognitive signal. We should therefore be able to label input brain maps from a shared universal low-dimensional brain representation, that we assume to be a combination (described in the second layer) of resting-state functional networks (assigned to the first layer). Our approach simultaneously learns to compute and to decode from this representation, for every study of our corpus.

As our model suffers from parametrization invariances, we perform a post-hoc linear transformation of the second and third layer, based on an ensembling method, to uncover an interpretable representation (Figure 7.1c) of the learned dimension reduction. Together, the first two layers project input data onto *multi-study task-optimized networks* (MSTON), whose loadings offer a general multi-study and

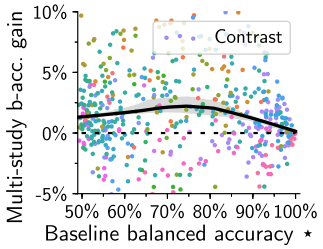
multi-subject representation of the cognitive signal contained in statistical maps.



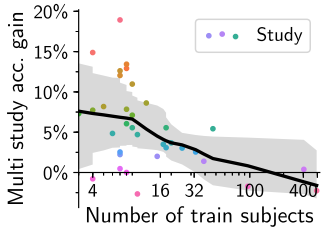
(a) Using the multi-study model improves performance for 28 / 35 studies



(b) Performance details for 20 half-split evaluations



(c) Gain vs. baseline score



(d) Gain vs. study size

Figure 7.2 – Performing joint training improves the performance of inter-subject decoding for most studies **(a)**. Overall, decoding from task-optimized networks leads to a median improvement accuracy of 5%; improvement is skewed across studies **(b)**. Studies of typical size strongly benefits **(d)** from transfer learning, whereas little information is to be gained for larger or easier to decode studies **(c)**

7.2.2 *Data and performance metrics*

We present the results obtained by applying our method on a set of 35 publicly available task fMRI studies, listed in Table (7.1); a few are performed on cohorts of hundreds of subjects (*e. g.,* HCP, Camcan, LA5C), but most feature more common cohorts of 10 to 20 subjects. These studies all follow different experimental protocols, although those are known to recruit related aspects of cognition (*e. g.,* motor, attention, judgement tasks, object recognition). We measure *accuracy* on left-out subjects for each study, and compare the scores obtained by our model to results obtained by simpler baseline decoders, that classify z-maps separately from each study, and directly from voxels. To analyse more specifically the impact of our method on the prediction accuracy for each contrast/condition, we also discuss *balanced accuracy*, that is computed for each predicted class. Details on data and metrics are reported in the detailed method section 7.4.

7.2.3 *Multi study training inter-subject decoding*

Figure 7.2 summarizes our quantitative results. For 28 out of the 35 task fMRI studies that we consider (Figure 7.2a), following our training procedure and thereby decoding contrast/condition from multi-study task-optimized networks brings a significant improvement in test prediction accuracy. It reaches +17% for the most sensitive studies, with a median of 4.9% across studies and cross-validation splits.

We explain our model performance from the *transfer learning* it permits across the many study decoding tasks. By minimizing a *joint* objective that combines training losses from every study, we learn a second-layer representation that is efficient for many study-specific decoding tasks; the second layer parameters therefore incorporate information from all studies; the joint objective further permits information transfer among the many classification heads of the third layer. Although we have no knowledge on how experiments are effectively related from a cognitive point of view, our quantitative results show that some of these relations can be *learned* during training to improve decoding performance.

Studies that benefit from using multi-study training have diverse cognitive focuses. Among the highest accuracy gains, we find cognitive control (stop-signal), classification studies, and localizer-like protocols. Our corpus contains many of such studies: the number of samples that brings information on the associated cognitive networks is substantially increased from single-study to multi-study decoding. Our model thus learns to capture the activation of these networks across subjects more efficiently, thereby leading to the observed improvement. In contrast, for a few studies, among which HCP and LA5C, we observe a slight negative transfer effect. This is not surprising: as HCP holds 900 subjects, it may not benefit from the aggregation of much smaller studies; LA5C focuses on higher-level cognitive pro-

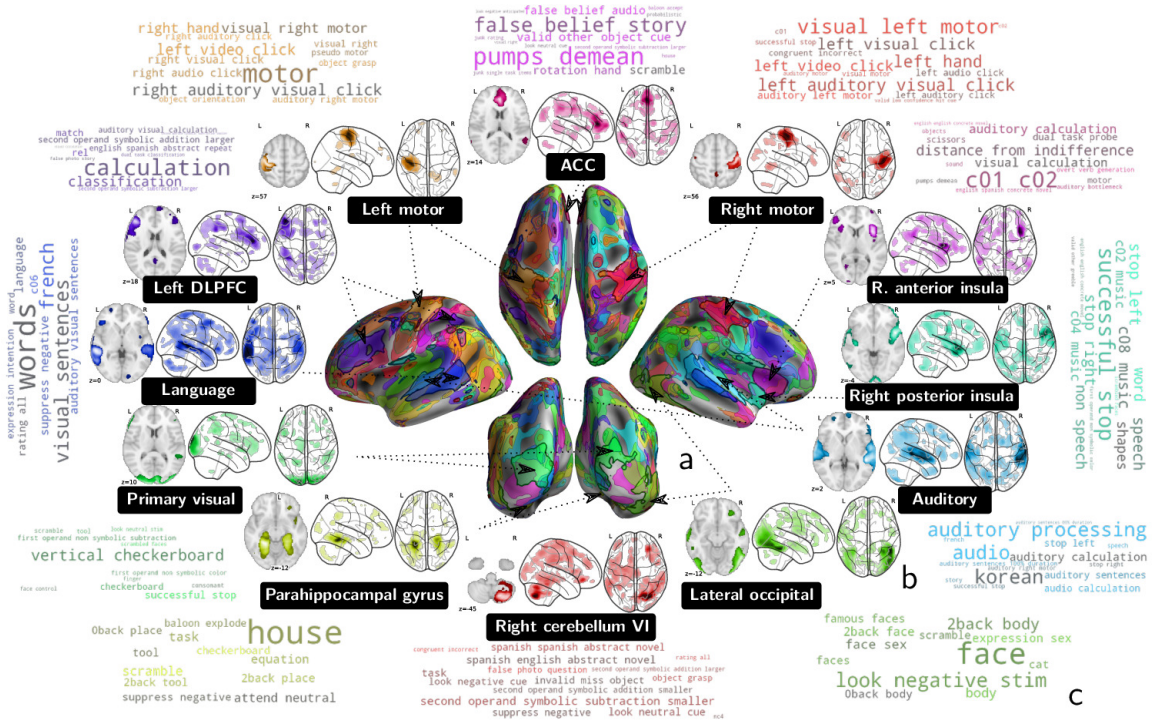


Figure 7.3: Visualization of some of task-optimized networks. Our approach allows to learn networks which are important for inter-subject decoding across studies. These networks, individually focal and collectively well spread across the cortex, are readily associated with the cognitive tasks they contribute to predict. We display a selection of these networks, named with the salient anatomical brain region they recruit, along with word clouds representation of the stimuli each network pushes to predict.

cesses with limited counterparts in the other studies, which prevents effective transfer.

Figure 7.2b shows that simply projecting data onto resting-state functional networks instead of using our three layer model does not significantly improve decoding, although the net effect varies from study to study. Appending a further *supervised* dimension reduction is thus necessary to improve overall decoding accuracy. As expected (Figure 7.2c), easy and hard-to-classify contrast classes little benefit from multi-study training, whereas classes whose balanced accuracy is around 80% profit from the highest balanced accuracy improvement. Figure 7.2d shows that the benefit of multi-study training is higher for smaller studies, confirming that out method can be seen as a regularization from external data. To further outline the benefits of multi-study training for small datasets, we show how it affects learning curves in Section 7.4: gain increases as training size is reduced.

7.2.4 *Multi-study task-optimized networks*

Training the second and third layer of our model using stochastic gradient descent identifies a *subspace* of the brain images onto which projecting helps decoding. Found subspaces prove remarkably stable across runs (see Section 7.4.6). Performing non-negative matrix factorization over the parameters of the second layer across multiple runs finds interpretable directions in a "mean" subspace. In voxel space,

these directions form multi-study task optimized networks, which constitutes the support of the learned low-dimensional representation of input z-maps.

We outline the contours of the extracted MSTON in Figure 7.3a. The networks cover the entire cortex, an expected design consequence, fostered by the broad coverage of cognition of the studies we gathered. Task optimized networks should indeed capture information for discriminating in between cognitive classes with very diverse localizations. Overall, the activations associated with the 545 contrasts of our analysis cover the entire cortex, which pushes MSTON to be well spread over the brain. Brain regions that are systematically involved and studied in task fMRI protocols, *e. g.*, motor cortex, auditory cortex and primary visual cortex are over-segmented by MSTON, *i. e.* appear in several different networks. As capturing information in these regions is crucial for decoding many contrasts in our corpus, our model dedicates a large part of its representation capability for it. Decoding requires to compare distributed activation: as an apparent consequence, MSTON are non-connected networks, as outlined in Figure 7.3b. For instance, both fusiform gyri appears together in the yellow network.

Most importantly, Figure 7.3b-c shows that our method singles out networks with cognitive meaning. Every network is important for classifying z-maps into a few classes, whose names are represented in word-clouds (Figure 7.3c). Our method finds cognitive networks at different levels. At a lower level, it identifies the primary visual cortex, associated with contrasts such as checkerboard stimuli, and both hand motor cortices, associated with many tasks demanding motor functions. At a higher level, it identifies the left DLPFC and the IPS in a single network, which is recruited by decoding tasks related to calculation and comparison. It successfully delineates the language network and the right posterior insula, which is detected to be important in decoding tasks involving music. Several found networks involve regions of the brains recruited by wide range of tasks, such as the cerebellum, the anterior insula, and the ACC, a part of the salience network.

7.2.5 *Impact of multi-study training on classification maps*

To better understand how the use of multi-study training and layered architecture improve decoding performance, we compare classification maps obtained using our model to baseline classification maps in Figure 7.4a. Those are simple to obtain, as our model eventually remains a simple linear classifier from voxels to classes. For contrasts for which balanced accuracy gain is significant, the classification maps are less noisy and more focal. They single out determinant regions more clearly, *e. g.*, the fusiform face area (FFA, row 1) in classification maps for the face-vs-house contrast, or the left motor cortex in maps (row 2) predicting pumping action in BART tasks. The language network is typically better delineated by our model (row 3),

and so is the posterior insula in music related contrasts (row 4). These improvements are due to two aspects: first, projecting onto a lower dimension subspace has a denoising effects on z-maps, that is already at play when projecting onto simple resting-state functional networks. Next, using multi-study task-optimized networks contribute to finding sharper images. Our method slightly decreases performance for a small fraction of contrasts: typically, maps associated vertical checkerboard (row 5), an easy-to-decode and very localized condition. Our model renders them as more distributed, a consequence of multi-study training that has here a negative effect.

In a dual perspective, we display in Figure 7.4b the representation of input z-maps that the projection on task-optimized networks brings. Projected data are more focal, *i. e.* spatial variations that are unlikely to be related to cognition are smoothed. It is therefore less confounded, which allows decoders to generalize better across subjects than when classifying raw input directly. This is once again a combined effect of the first layer (projection on functional networks) and of the trained second layer.

In Figure 7.5, we compare correlation between classification maps obtained with our model and the baseline decoder. The absolute correlation between classification maps within and across studies is higher on average. Th is is because the whole classification matrix is low-rank and influenced by the many studies we consider — the classification maps of our model are supported by networks relevant for cognition. As a consequence, it is easier to cluster maps into meaningful groups using hierarchical clustering based on cosine distances. For instance, we outline inter-study groups of maps related to left-motor functions, or calculation tasks. Hierarchical clustering on baseline maps is less successful: the associated dendrogram is less structured, and the distortion introduced by clusters is higher (as suggested by the smaller cophenetic coefficient). Clusters are harder to identify, due to a smaller contrast in the correlation matrix. Multi-study training thus acts as a regularizer, by forcing maps from each study to be more correlated to maps from other studies.

7.3 DISCUSSION

Our approach shows that using hierarchical models trained end-to-end can be successful in functional neuroimaging. This is interesting in several aspects. First, in practice, our approach can be seen as a universal way to improve the accuracy of decoding in a new study. Many task fMRI experiments are still performed on cohorts of less than 30 subjects. In this regime,it is highly likely that decoding performance improves when aggregating existing studies to the new one in a factored, multi-study model (Figure 7.2a,d). As the repositories of publicly available data are progressively getting normalized and accessible, our model provides an easy-to-deploy upgrade over simple decoders. We have shown that improvements are also qualitative, as the interpretation of decoding maps is made easier (Figure 7.4).

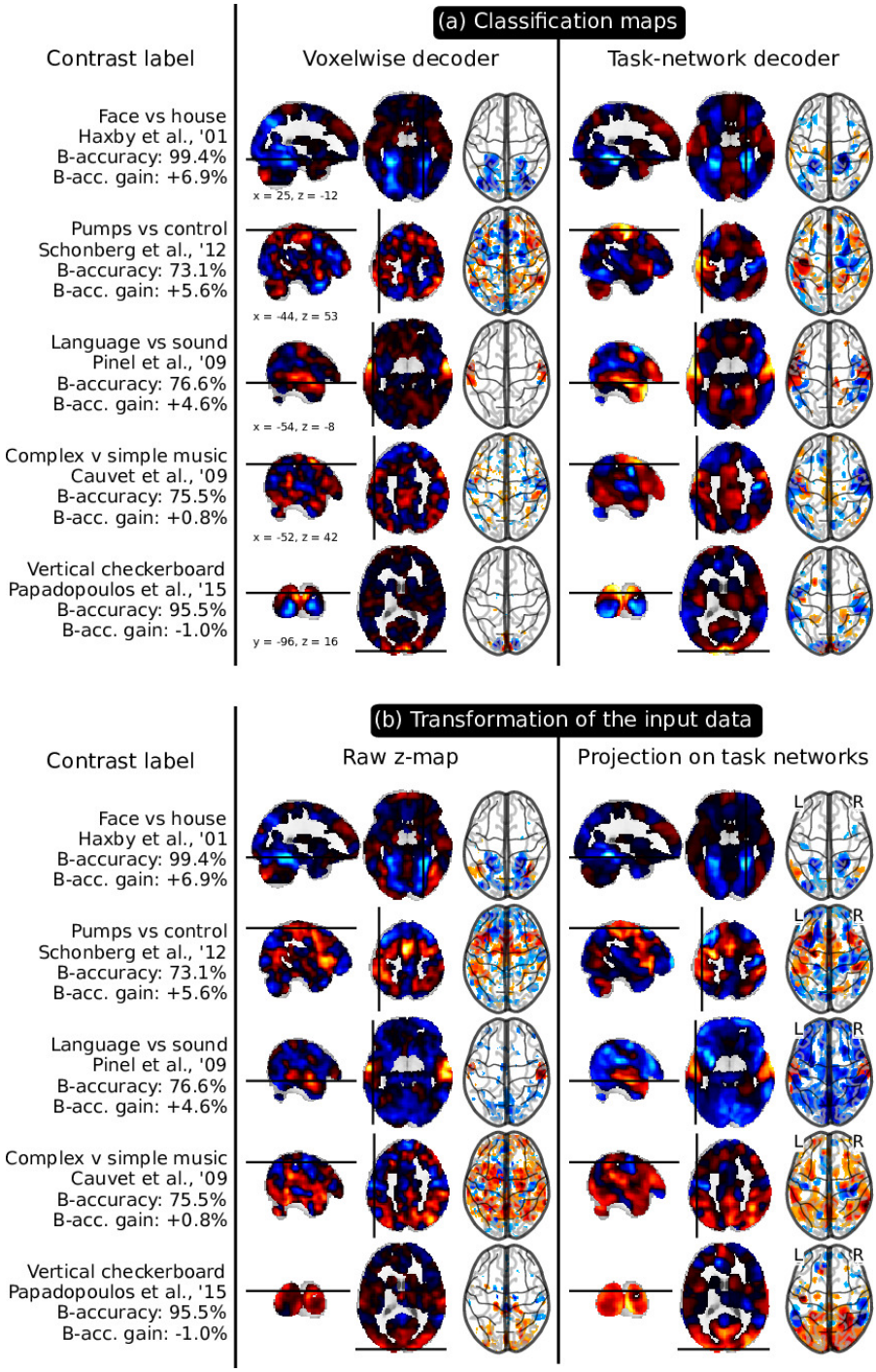


Figure 7.4 – Classification maps **(a)** obtained from multi-study training of decoding models are smoother and more focal. Relevant brain regions are often better underlined. In a dual perspective, the representation of input data **(b)** on task-optimized networks is simpler and therefore easier to classify.

Secondly, our approach shows the benefits of uncovering interpretable cognitive networks that capture information relevant for many decoding tasks. This provides quantitative evidence of the structuring of the human mind in various basic networks. Capturing all these networks is beyond the scope of any single fMRI study. Yet

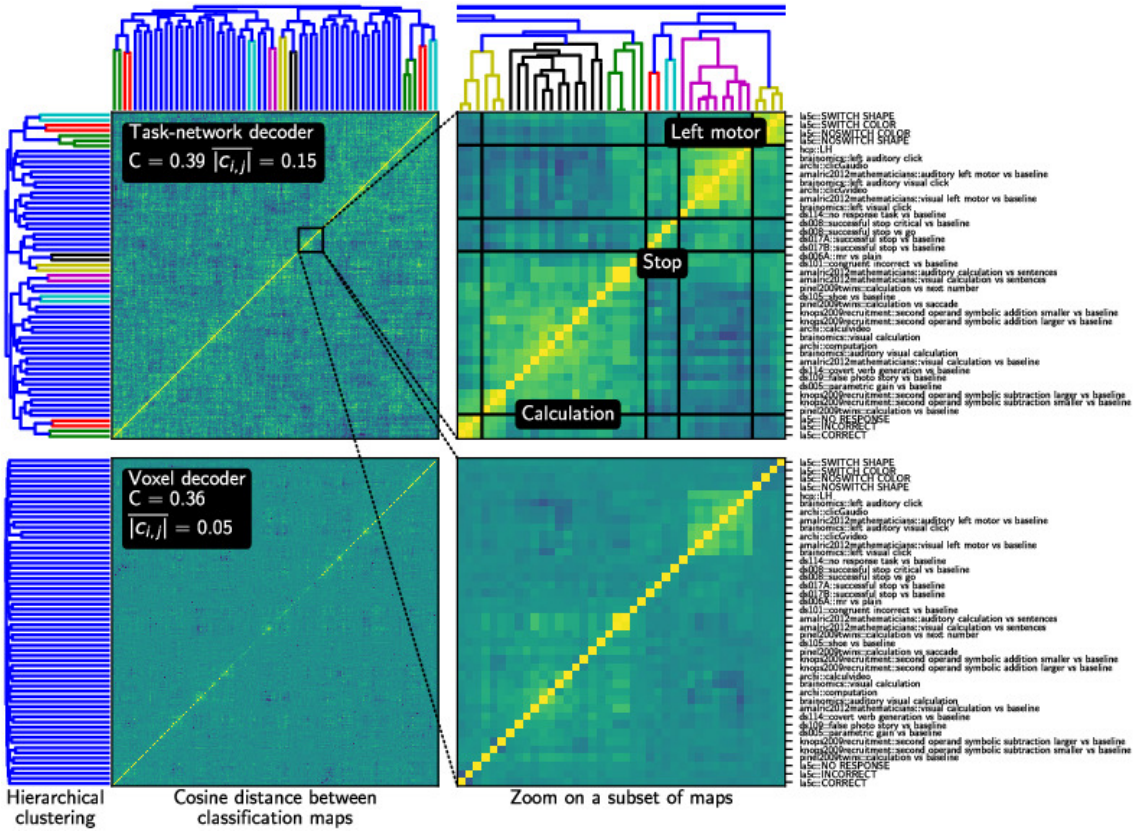


Figure 7.5: Cosine distances between classification maps, obtained with our multi-study decoder (top) and with decoders learned separately (bottom), clustered using average-linkage hierarchical clustering. The classification maps obtained when decoding from task-optimized networks are more easily clustered into cognitive-meaningful groups using hierarchical clustering — the cophenetic coefficient of the top clustering is thus higher.

aggregation of many cognitive studies allows to find interesting approximations, that we call MSTON and study in Figure 7.3.

Our approach was driven by the recent successes of deep non-linear models in computer vision and medical imaging. Although this may seem disappointing, we were not able to increase performance by departing from linear models: any introduction of non linearities in our models leads to a drastic increase of overfitting and does not improve left-out accuracy. On the other hand, the principle of using a layered model proves successful: having more fMRI data at our disposal allows to learn "deeper" models, although those should remain linear. Sticking to linear models has the further advantage of allowing easy interpretation of decoding models. Techniques issued from the deep learning communities prove very useful to train models that generalize well across subjects: we used dropout regularization, batch normalization and advanced stochastic gradient technique as those are essential for successful transfer learning and good generalization performance. We departed from the traditional convex models used in neuro-imaging: for this reason, we had to resort to post-hoc analysis of learned models, as detailed in Section 7.4.6, to uncover MSTON, inspired by methods for interpreting neural network predictions.

We suggest that widening our model by aggregating more studies using a systematic pipeline will allow us to find better descriptions

of task-optimized networks. These could in turn be used in subsequent analysis. For example They may be tuned at the individual level using recent methods akin to Tavor et al. (2016), or leveraged to compute biomarkers in place of resting-state functional networks (used in *e. g.,* Abraham et al., 2017; Greicius, 2008).

7.4 DETAILED METHOD

We describe in mathematical terms the multi-layer decoder at the center of our method. We start by formalizing the joint objective loss and the model training process.

7.4.1 *Inter-subject decoding setting*

We consider N task functional MRI studies, on which we perform inter-subject decoding, as formalized in Chapter 2. In each study j, n^j subjects are made to perform several tasks. Acquired BOLD time-series are registered to a common template using non-linear registration, after motion and slice-timing corrections. BOLD time-series are then fed to standard analysis, that fits a linear model relating the design matrix of each experiment to the activation of every voxel. From the obtained beta maps, we compute z-statistics maps, either associated with each of the base conditions (stimulus or task) of the experiments, or to contrasts defined by the study's authors. In both case, z-maps are labeled with a number $1 \leqslant k \leqslant c^j$ that corresponds to k-th contrast/base condition (called contrast in the following). Overall, this produced a set of z-maps $(\mathbf{x}_i^j)_{i \in [c^j n^j]}$ living in $\mathbb{R}^p$, where p is the number of voxels, associated with a sequence of contrast $(k_i)_{i \in [c^j n^j]}$. Inter-subject decoding proposes a model $f_\theta : \mathbb{R}^p \to [1, c^j]$ that predicts contrast from z-maps, *i. e.* $\hat{k}_i^j \triangleq f_\theta(\mathbf{x}_i^j)$, where θ is learned from training data, and the performance of the model is assessed on left-out subjects.

7.4.2 *Baseline voxel-space decoder*

Baseline decoders are linear classifier models defined separately for each study j, that take full brain images as input. For every input map $\mathbf{x}_i \in \mathbb{R}^p$, we compute the logits $\mathbf{l}_i$ in $\mathbb{R}^c$ as

$$\mathbf{l}_i \triangleq \boldsymbol{W}\mathbf{x}_i + \mathbf{b},$$

where $\boldsymbol{W} \in \mathbb{R}^{c \times p}$ and $\mathbf{b} \in \mathbb{R}^c$ are the parameters of the linear model to be learned for study j — we drop the exponent j in this paragraph. Logits yield a classification probability vector using the softmax operator. At test time, we predict the label corresponding to the maximal logit, *i. e.* $\hat{y}_i = \operatorname{argmax}_{1 \leqslant y \leqslant c} l_{i,y}$. The model is trained on the data

$(x_i, y_i)_{i\in[n]}$ by minimizing the regularized multinomial classification problem

$$\min_{\substack{W\in\mathbb{R}^{c\times p}\\ b\in\mathbb{R}^c}} -\frac{1}{n}\sum_{i=1}^{n}\left(l_{i,y_i}(W,b)+\log(\sum_{k=1}^{c}\exp l_{i,k}(W,b))\right)+\lambda\|W\|_F^2. \tag{7.1}$$

7.4.3 *Baseline dimension reduced decoder*

A variant of the voxel-based decoders is obtained by introducing a first-layer dimension reduction learned from resting-state data. This amounts to computing

$$l_i \triangleq VDx_i + b,$$

where $V \in \mathbb{R}^{c\times k}$ form the classifying weights of the model, and the matrix $D \in \mathbb{R}^{k\times p}$ is *assigned* during training to functional networks learned on resting-state data, as detailed in Section 7.4.5. Multiplying input data by D projects statistical images onto meaningful resting-state components, in an attempt to improve classification performance and reduce computation cost, akin to the methods proposed in S. M. Smith et al. (2009) and Yeo et al. (2011). The model is trained solving the convex objective (7.1) separately for each study, replacing W by $V \in \mathbb{R}^{c\times k}$:

$$\min_{\substack{V\in\mathbb{R}^{c\times k}\\ b\in\mathbb{R}^c}} -\frac{1}{n}\sum_{i=1}^{n}\left(l_{i,y_i}(V,b,D)+\log(\sum_{k=1}^{c}\exp l_{i,k}(V,b,D))\right)+\lambda\|V\|_F^2. \tag{7.2}$$

Our results (Figure 7.2c) show that decoding from functional networks is not significantly better than decoding from voxels directly. For both baselines, the parameter λ is found by half-split cross-validation [1]. Training is performed using a L-BFGS solver. We use non standardized maps $(x_i)_i$ as input as we observed that standardization hinders performance.

7.4.4 *Three-layer model description*

Our three-layer model adds a second shared linear layer in between the projection on functional networks and the classification models. We still have

$$l_i^j \triangleq W^j x_i^j + b^j$$

for every z-map i and study j. However, in this case, we introduce a coupling in between the various parameters $(W^j)_{j\in[N]}$ of each study, that should decompose on on common basis LD, where L is estimated from the whole corpus of data. Formally, we assume that

1. over the values $(10^i)_{i=\{-3,-2\ldots,3\}}$

there exist matrices $\mathbf{L} \in \mathbb{R}^{l \times k}$, $(\mathbf{U}^j)_{j \in [N]}$, so that $l < k < p$, and for all $j \in [N]$,

$$\mathbf{W}^j \triangleq \mathbf{U}^j \mathbf{L} \mathbf{D}, \quad \text{where} \quad \mathbf{U}^j \in \mathbb{R}^{c^j \times l}. \tag{7.3}$$

The matrix $\mathbf{D}$ corresponds to the first-layer weights pictured in Figure 7.1, $\mathbf{L}$ to the second's, and $(\mathbf{U}^j, \mathbf{b}^j)_j$ to the various classification heads of the third. In this work, we choose $k \approx 512$, $l = 128$. While $\mathbf{D}$ remains fixed, the second-layer matrix $\mathbf{L}$ and the N classification heads $(\mathbf{U}^j)_{j \in [N]}$ are jointly learned during training, a necessary step toward improving decoding accuracy. The "shared-layer" parameterization (7.3) is a common approach in multi-task learning (Ando and T. Zhang, 2005; Y. Xue et al., 2007), and should allow *transfer learning* between decoding tasks, under certain conditions. In our setting, both the data distribution from the different studies and the classification task associated with each study differ — this is a particular case of *inductive transfer learning*, described by Pan and Yang (2010).[10]

[10] *less studied than the classical multi-task setting where input data are single-source but learning tasks are multiple.*

Without refinement nor regularization, we seek a local minimizer the following non-convex objective function, that combines the classification objectives (7.1) from all studies:

$$\min_{\substack{\mathbf{L} \in \mathbb{R}^{l \times k} \\ (\mathbf{U}^j, \mathbf{b}^j)_j}} - \sum_{j=1}^{N} \frac{(n^j)^\beta}{n^j} \sum_{i=1}^{n^j} \Big(l^j_{i, y_i}(\mathbf{U}^j, \mathbf{b}^j, \mathbf{L}) - \log\big(\sum_{k=1}^{c^j} \exp l^j_{i,k}(\mathbf{U}^j, \mathbf{b}^j, \mathbf{L})\big)\Big), \tag{7.4}$$

where the dependence on $\mathbf{D}$ is implicit. The scalar $\beta \in [0, 1]$ is a parameter that regulates the importance of each study in the joint objective, that we further discuss later. We solve the problem (7.4) using stochastic optimization. Namely, at each iteration, we compute an unbiased estimate of the objective (7.4) and its gradient w.r.t. the model parameters, to perform a stochastic gradient step. For this, we randomly choose study j with a probability proportional to $(n^j)^\beta$, and consider a mini-batch of z-maps $(\mathbf{x}^j_i)_{i \in B}$, that we use to compute the unbiased objective estimate

$$-\frac{1}{B} \sum_{i=1}^{n} -\Big(l^j_{i,k_i} \log\big(\sum_{k=1}^{c} \exp l^j_{i,k}\big)\Big), \tag{7.5}$$

from which we compute gradients w.r.t. $\mathbf{L}$, $\mathbf{U}^j$ and $\mathbf{b}^j$.

Minimizing (7.4) leads to strong overfitting and low performance on left-out data, with performance similar to fitting (7.1) without regularization, separately for each study. Adding ℓ_2 regularization to the second and third layer weights gives little benefit, as we discuss in Section 7.5.1 On the other hand, introducing *dropout* (Srivastava et al., 2014) during training alleviates the overfitting issue and allows transfer learning to occur. Dropout is a stochastic regularization method that is designed to prevent the weights from each layer to co-adapt, and ensure that the information is well spread across coefficients rows and columns (Neyshabur, 2017). In our case, this favors transfer learning, as it ensures that no single row of $\mathbf{L}$, or in

plain words no task-optimized network, becomes dedicated to a *single* study. We further discuss the different methods that foster transfer of information between studies in Section 7.5.1.

We use the variational flavor of dropout (Kingma et al., 2015) to make individual dropout rate for every study adaptive. This slightly improves performance compared to binary dropout: every decoding task requires a different level of regularization, depending on the size of the study and the hardness of the task, and it is beneficial to estimate it from data. During training, at every iteration, for every input sample i of a minibatch from study j, we randomly draw two multiplicative noise matrices

$$\mathbf{M}_D = \mathrm{Diag}([b_{D,t}]_{t\in[k]}), \quad \mathbf{M}_L^j = \mathrm{Diag}([b_{L,t}]_{t\in[l]}),$$

where $b_{D,t} \sim \mathcal{N}(1, \alpha)$ and $b_{L,t} \sim \mathcal{N}(1, \alpha^j)$, with α fixed and α^j estimated from data.[2] We then compute the noisy logits

$$l_i^j \triangleq \mathbf{U}^j \mathbf{M}_L^j \mathbf{L} \mathbf{M}_D \mathbf{D} x_i^j + \mathbf{b}^j,$$

and use these to compute the loss (7.5), to which we add a regularization term that regulates the learning of α^j, introduced by Molchanov et al. (2017). We compute the gradient w.r.t. $\mathbf{L}$, $\mathbf{U}^j$, $\mathbf{b}^j$ using the local reparametrization trick (Kingma et al., 2015). We refer to Molchanov et al. (2017) for more details on variational dropout and Bayesian grounding of this approach.

Optimization is performed using *Adam* (Kingma and Ba, 2015), a flavor of stochastic gradient descent that depends less on the step-size. We use batch normalization (Ioffe and Szegedy, 2015) between the second and third layer, as it slightly improves performance — it reduces potential negative transfer learning — and training speed.

7.4.5 *Resting-state data*

As mentioned above, we use resting-state data to compute the first-layer weights $\mathbf{D} \in \mathbb{R}^{k\times p}$, where $k = 512$. We consider data from the HCP900 release, and stack all records to obtain a data matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$. We then use the method proposed in Chapter 4– 6 to solve the sparse non-negative matrix factorization problem

$$\mathbf{A}, \mathbf{D} \triangleq \underset{\mathbf{D}\in\mathcal{C}, \mathbf{A}\in\mathbb{R}^{k\times n}}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{A}\mathbf{D}\|_F^2 + \lambda\|\mathbf{A}\|_F^2 \tag{7.6}$$

where the constraint $\mathcal{C} = \left\{\mathbf{D} \in \mathbb{R}^{k\times p}, \|\mathbf{d}_j\|_1 \leqslant 1, \mathbf{d}_j \geqslant 0\right\}$ enforces every dictionary component to live in the simplex of $\mathbb{R}^p$, ensuring sparsity and positivity of the functional networks. The sparsity level is chosen so that $\mathbf{D}$ covers the whole brain with as little overlap as possible.

2. This *Gaussian* dropout has a similar behavior to the more commonly used binary dropout with parameter $p = \frac{\alpha}{\alpha+1}$.

SECOND-LAYER INITIALIZATION. To initialize the weights of the second layer weights, we learn a smaller dictionary $\mathbf{D}_l \in \mathbb{R}^{l \times p}$ as in (7.6), where $l = 128$. We then compute the initial weights $\mathbf{L}_l$ so that $\mathbf{D}_l \approx \mathbf{L}_l \mathbf{D}$ using least-square regression. This way, the two first layers of the initial layer initially behave as a projection on $l = 128$ larger functional networks, which is a reasonable prior for reducing brain statistical maps. This initialization slightly improves performance, as we discuss in Section 7.5.3.

GREY MATTER RESTRICTION. To help interpreting the obtained model, we found it helpful to remove from $\mathbf{D}$ the fraction (9%) of the functional networks components located in the white matter and the cerebrospinal fluid areas, turning $k = 512$ into $k = 465$. We discuss the effect of this restriction in Section 7.5.3.1.

7.4.6 *Post-hoc model transformation with ensembling*

Given any invertible matrix $\mathbf{M} \in \mathbb{R}^{l \times l}$, the non-regularized version of the objective (7.4) is left invariant when transforming $\mathbf{L}$ into $\mathbf{M}\mathbf{L}$ and each $\mathbf{U}^j$ into $\mathbf{U}^j \mathbf{M}^{-1}$. This prevents us from interpreting the coefficients of $\mathbf{L}$ at the end of the training procedure, and to retrieve relevant networks by reading the weights of the second weight. The only aspect of $\mathbf{L}$ that remains unchanged after a linear parameter transformation is its span. Dropout regularization, that favors the canonical directions in matrix space (Srivastava et al., 2014), should break this symmetry, but does not help to uncover meaningful directions in the span of $\mathbf{L}$ in practice.

On the other hand, we found that this span was remarkably stable across runs on the same data, whether when varying initialization or simply the order in which data are streamed during stochastic gradient descent. More precisely, we trained our model 100 times with different seeds, and concatenated the weights $(\mathbf{L}_r)_r$ of the second-layer into a big matrix $\bar{\mathbf{L}}$. We performed a SVD on this matrix, and observed that the first $l = 128$ components captured 98% of the variance of $\bar{\mathbf{L}}$ when using the same initialization but different streaming order, and 96% when also using a different random initialization. Despite the many local minima that objective (7.4) admits, the span of $\mathbf{L}$ thus remains close to some reference span that we can extract with a matrix factorization method.

The above remark suggested the following ensemble method. We run the learning algorithm $r = 100$ times, and store the weights $(\mathbf{L}_r)_r$ of the second layer for each run, along with the average matrices and biases

$$\bar{\mathbf{W}}^j = \frac{1}{r} \sum_{n=1}^{r} \mathbf{U}_s^j \mathbf{L}_s \qquad \bar{\mathbf{b}}^j = \frac{1}{r} \sum_{n=1}^{r} \mathbf{b}_r^j, \quad \forall j \in [N]$$

that combine the second and third-layer weights and biases for each study j and run n, and average them across runs. We then stack the second-layer weights $(\mathbf{L}_r)_r$ into a fat matrix $\tilde{\mathbf{L}} \in \mathbb{R}^{l\,r \times k}$ on which we

perform sparse non-negative matrix factorization. Namely, we compute $\bar{\mathbf{L}} \in \mathbb{R}^{l \times k}$, the new weight matrix for the second layer, solving

$$\min_{\mathbf{L} \in \mathcal{C}} \min_{\mathbf{K} \in \mathbb{R}^{l \times l_r}} \frac{1}{2} \|\tilde{\mathbf{L}} - \mathbf{K}\mathbf{L}\|_F^2 + \lambda \|\mathbf{K}\|_2^2,$$

where $\mathcal{C} = \left\{\mathbf{L} \in \mathbb{R}^{l \times k}, \|l^j\|_1 \leqslant 1, l^j \geqslant 0\right\}$ and λ regulates the sparsity of $\bar{\mathbf{L}}$. Finally, we compute new weights $\bar{\mathbf{U}}^j$ for all the classification heads of the third layer, so that $\bar{\mathbf{W}}^j \approx \bar{\mathbf{U}}^j \mathbf{L}^\star$, from a least-square point of view, for each study j. The new model is then formed of parameters $\mathbf{D}, \bar{\mathbf{L}}, (\bar{\mathbf{U}}^j, \bar{\mathbf{b}}^j)_{j \in [N]}$. In plain words, we obtain sparse non-negative second-layer weights $\mathbf{L}^\star$, and define from these weights a new model that is as close as possible to the ensemble of all learned models $\left\{\mathbf{D}, \mathbf{L}_s, (\mathbf{U}_s^j, \mathbf{b}_s^j)_j\right\}_{N \in [r]}$.

The columns of $\bar{\mathbf{L}}$ are now interpretable separately, as the non-negative and sparse constraints have broken the inherent parameter invariance of the original model. The columns of $\bar{\mathbf{L}}$ hold the coefficients for combining resting-state networks held in $\mathbf{D}$ into l multi-study task-optimized networks $\bar{\mathbf{L}}\mathbf{D} \in \mathbb{R}^{l \times p}$. We initialize the sparse NMF algorithm with the weights $\mathbf{L}_l$ computed in Section 7.4.5, to inject a small prior regarding final MSTON distribution: before running NMF, those are set to $\mathbf{L}_l \mathbf{D} \approx \mathbf{D}_l$, *i.e.* are close to large resting-state functional networks.

We observed that directly enforcing negativity/sparsity over $\mathbf{L}$ during the training of the model led to a strong loss in accuracy. Finding a consensus model through a post-hoc ensembling transformation thus proves to be the right solution for obtaining both performance improvement *and* interpretability.

7.5 DESIGN DISCUSSION

In this section, we discuss the various choices when made for designing our model and training procedures. To this end, we perform diverse quantitative and qualitative comparisons of model variants.

7.5.1 *How to induce transfer learning ?*

We start by discussing the various way in which we can force information sharing across studies in training our multi-layer model.

7.5.1.1 *The need for objective coupling*

Without modification nor constraint on the second layer output size l, we cannot expect to observe any transfer learning by solving the joint objective (7.4). Indeed, in the general case where we allow $l \geqslant c \triangleq \sum_{j=1}^{N} c^j$, we let $(\tilde{\mathbf{V}}^j, \mathbf{b}^j)_j \in \mathbb{R}^{c \times k}$ be the unique solutions of the non-regularized convex problems (7.1) and $\tilde{\mathbf{V}} \in \mathbb{R}^{c \times k}$ be their vertical concatenation. We then form the matrices

$$\mathbf{L} = \begin{bmatrix} \tilde{\mathbf{V}} \\ \mathbf{0} \in \mathbb{R}^{l-c\times k} \end{bmatrix} \in \mathbb{R}^{l\times k} \text{ and } \begin{bmatrix} \mathbf{U}^1 \\ \vdots \\ \mathbf{U}^N \end{bmatrix} \triangleq \left[\mathbf{I}_c \in \mathbb{R}^{c\times c}, \mathbf{0} \in \mathbb{R}^{l-c\times l}\right], \tag{7.7}$$

in which $\mathbf{L}$ is split into row-blocks $(\tilde{\mathbf{V}}^j)_j$, dedicated to and learned on *single studies*. It follows from elementary considerations that the matrices $(\mathbf{L}, (\mathbf{U}^j, \mathbf{b}^j)_j)$ form a global minimizer of (7.4), that is formed from the solutions of the *separated* problems (7.2). It is therefore possible to find solutions of (7.4) for which no transfer occur. Two modifications of the objective (7.4) allows to enforce transfer: dropout regularization and low-rank constraints, that we present and compare.

7.5.1.2 *Dropout as a transfer incentive*

First, as presented in the method section (7.4), we can use dropout in between the second layer weight L and the third layer head weights $\mathbf{U}^j$. Dropout prevents constructions of block-separated solution of objective (7.4) similar to the one proposed in (7.7). Indeed, every reduced sample $\mathbf{L}\mathbf{D}\boldsymbol{x}_i^j$ fed to the third layer classification head j can see any of his features corrupted by multiplicative noise $\mathbf{M}_L$ during training. This pushes the model to capture information relevant for all studies in every activation of the second layer. In other word, the projection performed on any task-optimized network $\mathbf{l}_h\mathbf{D}$, for $h \in [l]$ should be relevant for decoding every study. This fosters transfer learning as $\mathbf{L}$ carry multi-study aggregated information at the end of training, unlike in (7.7).

7.5.1.3 *Transfer through low-rank constraints/penalty*

A second approach to transfer is to force the matrices

$$\mathbf{V} \triangleq \begin{bmatrix} \mathbf{V}^1 \\ \vdots \\ \mathbf{V}^N \end{bmatrix} \triangleq \begin{bmatrix} \mathbf{U}^1 \\ \vdots \\ \mathbf{U}^N \end{bmatrix} \mathbf{L},$$

formed of the parameters of the joint objective (7.4) to be *low-rank*. In this case, the subspace of $\mathbb{R}^{c\times k}$ in which $\mathbf{V}$ evolves is strictly smaller than $\mathbb{R}^{c\times k}$, and we cannot always find a global minimum of the joint objective (7.4) formed with the solutions $\tilde{\mathbf{V}}$ of the separate objectives (7.2), as we did in the construction (7.7). As a consequence, the data from studies truly influence the solutions $(\mathbf{L}, (\mathbf{U}^j, \mathbf{b}_j)_j)$ of (7.4), and transfer is theoretically possible.

The low-rank property may be enforced in two ways. First, we may set it as a hard constraint, setting $l < c$ in the joint objective (7.4). This is in practice what we do when selecting $l = 128$, as $c = 545$ in our experiments.

Alternatively, following Srebro et al. (2004), we may resort to a convex objective function parametrized by $\mathbf{V} \in \mathbb{R}^{c\times k}$, that penalizes

$\mathbf{V}$ rank. We learn $\mathbf{V}^j \in \mathbb{R}^{c^j \times k}$ for all study $j \in [N]$ solving the joint objective

$$\min_{(\mathbf{V}^j, \mathbf{b}^j)_j} -\sum_{j=1}^{N} \frac{(n^j)^\beta}{n^j} \sum_{i=1}^{n^j} \left(l^j_{i,y_i}(\mathbf{V}^j, \mathbf{b}^j) - \log\left(\sum_{k=1}^{c^j} \exp l^j_{i,k}(\mathbf{V}^j, \mathbf{b}^j)\right)\right) + \lambda \left\| \left[\mathbf{V}^{1\top} \dots \mathbf{V}^{N\top}\right] \right\|_\star, \tag{7.8}$$

where $\|\mathbf{V}\|_\star$ is the nuclear norm of $\mathbf{V}$, defined as $\sum_{i=1}^{\min(c,k)} \sigma_i(\mathbf{V})$, with $(\sigma_i(\mathbf{V}))_i$ singular values of $\mathbf{V}$. The nuclear norm is a convex proxy for the rank of matrix $\mathbf{V}$. As a consequence, the rank of the solution decreases from $\min(c, k)$ to 0 as λ increases. The objective (7.8) is solvable using proximal-methods, *e. g.*, using FISTA (Beck and Teboulle, 2009). However, these methods become impractical when c becomes large — it requires to perform a $c \times c$ SVD at each iteration. Fortunately, there exists a non-convex objective (Rennie and Srebro, 2005), amenable to stochastic gradient descent (R. M. Bell and Koren, 2007), that includes the solution of (7.8) as a minimizer. It is obtained setting $l = \max(x, k)$ and adding ℓ_2^2 penalties to the objective (7.4):

$$\min_{\substack{\mathbf{L} \in \mathbb{R}^{l \times k} \\ (\mathbf{U}^j, \mathbf{b}^j)_j}} -\sum_{j=1}^{N} \frac{(n^j)^\beta}{n^j} \sum_{i=1}^{n^j} \left(l^j_{i,y_i}(\mathbf{U}^j, \mathbf{b}^j, \mathbf{L}) - \log\left(\sum_{k=1}^{c^j} \exp l^j_{i,k}(\mathbf{U}^j, \mathbf{b}^j, \mathbf{L})\right)\right)$$

$$+ \frac{\lambda}{2}\left(\|\mathbf{L}\|_F^2 + \sum_{j=1}^{N} \|\mathbf{U}^j\|_F^2 \right), \quad \text{where} \quad \mathbf{U}^j \in \mathbb{R}^{c^j \times l} \; \forall j \in [N].$$

We solve this objective using *Adam*, similarly to the main method. It is possible to continue using dropout in between the first and second layer while enforcing $\mathbf{V}$ to be low-rank — this can then be understood as a regularization technique through feature noising (S. Wager et al., 2013).

7.5.1.4 *Empirical comparison*

Both the dropout and low-rank approaches are competitive a priori to foster transfer learning. Our final method uses a combination of both, as it enforces a hard low-rank constraint and uses dropout. This choice was motivated by the comparison displayed in Figure 7.6. We compare three regularization variants, measuring the improvement due to hard low-rank constraints and the difference between dropout and ℓ_2. ℓ_2 accuracy gain is an upper-bound of its actual performance in practice, as we take the highest performing λ on the test sets. The three estimators uses input dropout ($p = 0.25$), while dropout between layer 2 and 3 is initialized to $p = 0.75$ when used. We observe that forcing $\mathbf{V}$ to be low-rank is beneficial, and that dropout regularization performs significantly better than low-rank inducing ℓ_2 penalties, which justifies using dropout and hard-rank constraints for regularization.

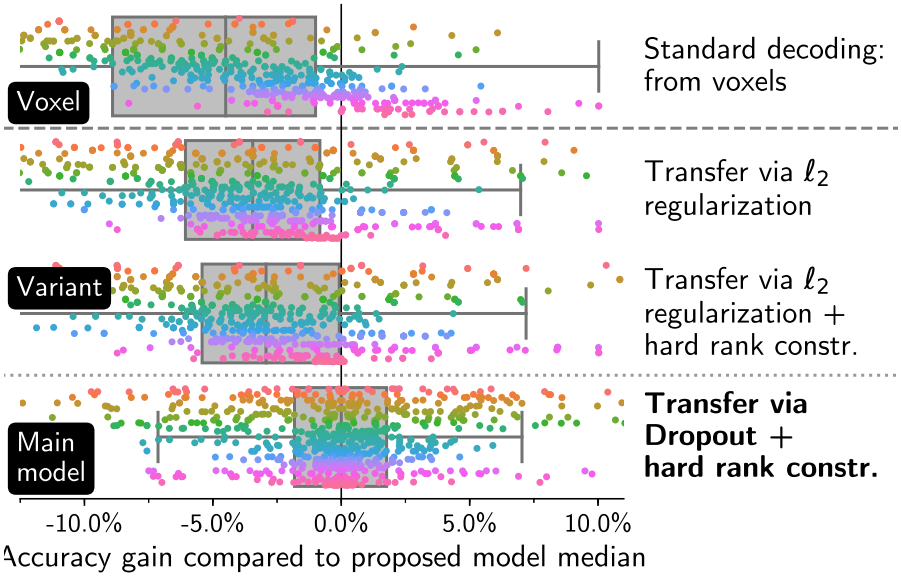


Figure 7.6 – Quantitative comparison of regularization techniques: dropout with hard-rank constraints outperform ℓ_2 regularization with and without hard-rank constraints.

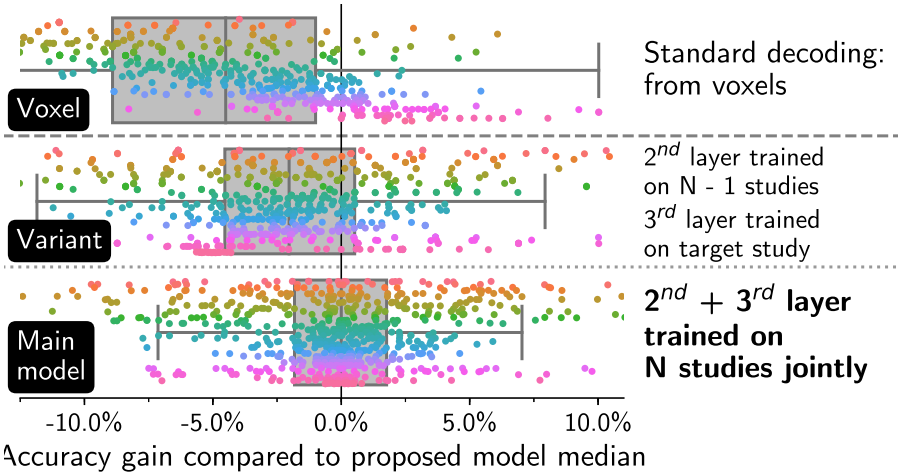


Figure 7.7 – Quantitative improvement linked to training the model on the join objective (7.4), versus improvement linked to transfer in the second-layer only.

7.5.2 *Performance without joint training*

We have argued that training the joint objective (7.4) improves decoding performance as the data from every study influences the learned weights in bot the second *and* third layer. This can be measured by comparing the performance of learning task-optimized networks on all studies but a target one, and freezing these networks (*i. e.* use them for a fixed dimension reduction) to learn the target decoder. We observe in Figure (7.7) that this approach, that allows transfer through the second layer only, performs better than the baseline, but worse than our approach. This shows that using a joint objective also foster transfer in the classification heads of the third layer.

7.5.3 *Interpretability incentives*

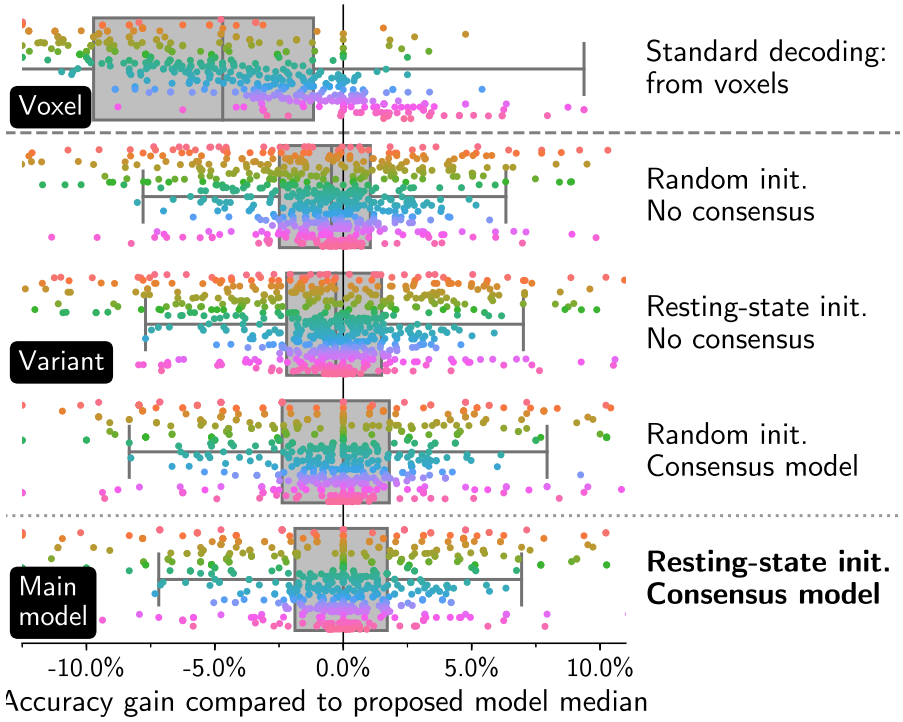


Figure 7.8 – Quantitative improvement linked to ensembling and resting-state initialization in our model.

Our approach involves model interpretability as a core requirement. Three factors contribute to output cognitive meaningful task-optimized networks. First, the initial first layer, learned on resting-state data, coarsens the resolution of networks in a way adapted to typical brain signal. Second, we compute a consensus model, so that the task-optimized network loadings held in $\mathbf{L}$ are non-negative and interpretable. Third, we initialize the second-layer weights so that $\mathrm{L}_{\text{init}}\mathbf{D}$ corresponds to resting-state functional networks. This initialization is used both during the training phase and the consensus phase.

CONSENSUS MODEL. In Figure 7.8, we measure the quantitative effects of the two later factors on decoder accuracy. Learning a consensus model using sparse NMF is crucial for finding interpretable direction in the span of $\mathbf{L}$. Without this refinement, the directions we obtain are similar to the one displayed in Figure 7.9a, and are not interpretable. The consensus phase also contributes positively to the model decoding performance (+.5% overall accuracy). We attribute this improvement to an ensembling effect similar to the benefits of bagging (Breiman, 1996), as the final model summarizes 100 training runs on the same data, with different random seeds.

RESTING-STATE INITIALIZATION. Figure 7.8 shows that resting-state based initialization of the second-layer has no measurable impact on performance, and thus provides more interpretable compo-

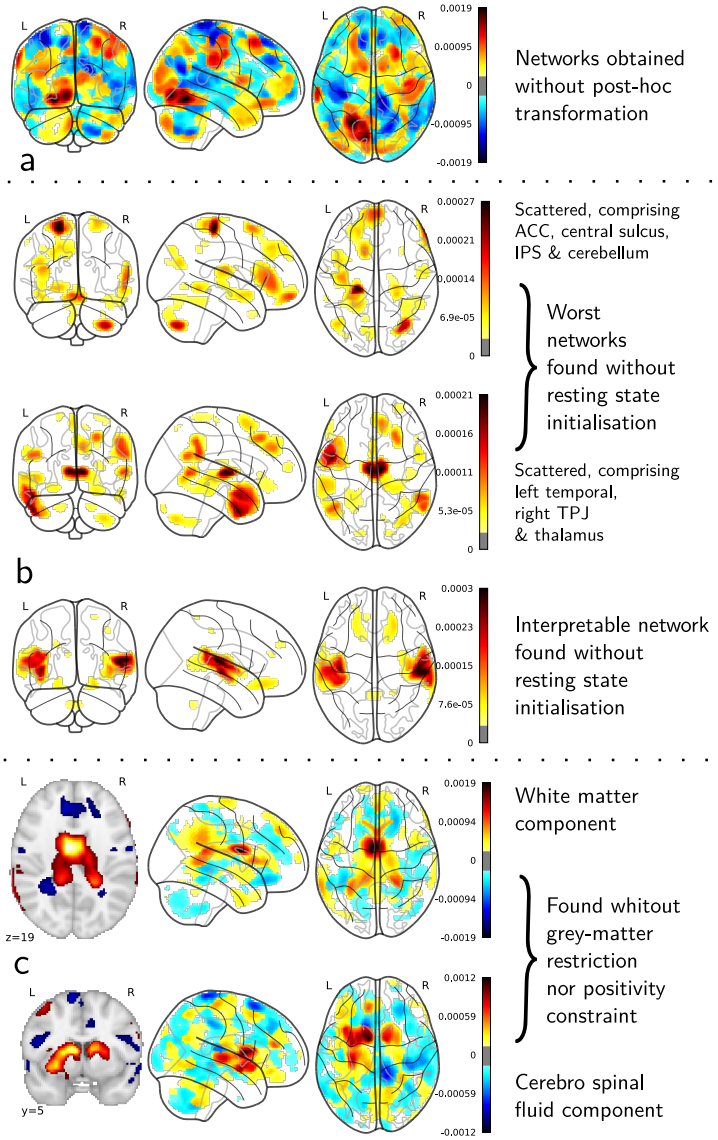


Figure 7.9 – Without interpretability refinements **(a)**, resting-state based initialization **(b)** and grey matter components selection **(c)**, some task-optimized networks may be hard to interpret/ not relevant from a cognitive perspective.

nents with no accuracy cost, as the qualitative discussion will show. Qualitatively, we show examples of three components found without resting-state initialization in Figure 7.9b. Two of those are scattered networks, that capture various connected components whose co-occurrence is not interpretable: those are likely artifacts due to random initialization. Using resting-state initialization finds such networks much more rarely. It remains interesting to note that most of the components found without resting-state based prior bear cognitive meaning, similar to the third components displayed in Figure 7.9b.

7.5.3.1 *Effect of selecting grey-matter components*

We project data onto a subset of 465 out of 512 functional networks learned on HCP resting-state data, selecting the networks that inter-

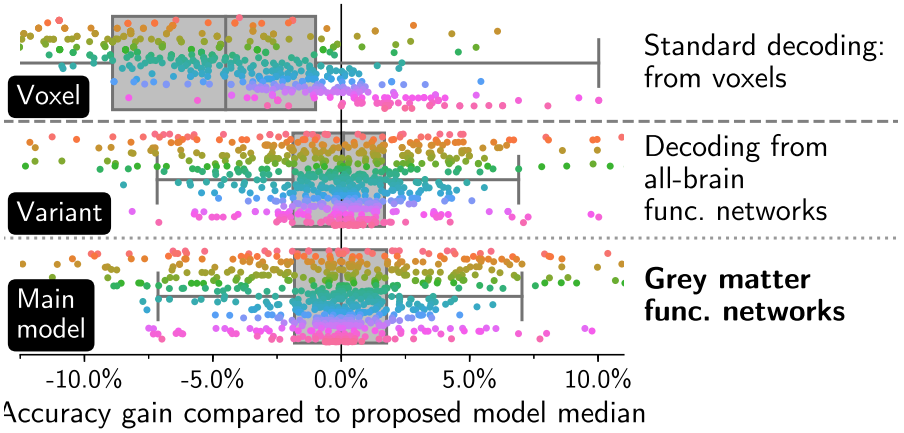


Figure 7.10 – Working with functional networks located in the grey matter only has not significant impact on performance.

sect with an anatomical grey-matter mask. This avoids finding MSTON that that are distributed or formed with non grey-matter regions. In Figure 7.9c, we show that without those precautions, our model finds networks located in the white matter and the neuro-spinal fluid zones. Quantitatively, as expected, performing classification from grey-matter components only brings a non-significant performance loss (Figure 7.10).

7.5.4 *Effect of variational dropout and batch normalization*

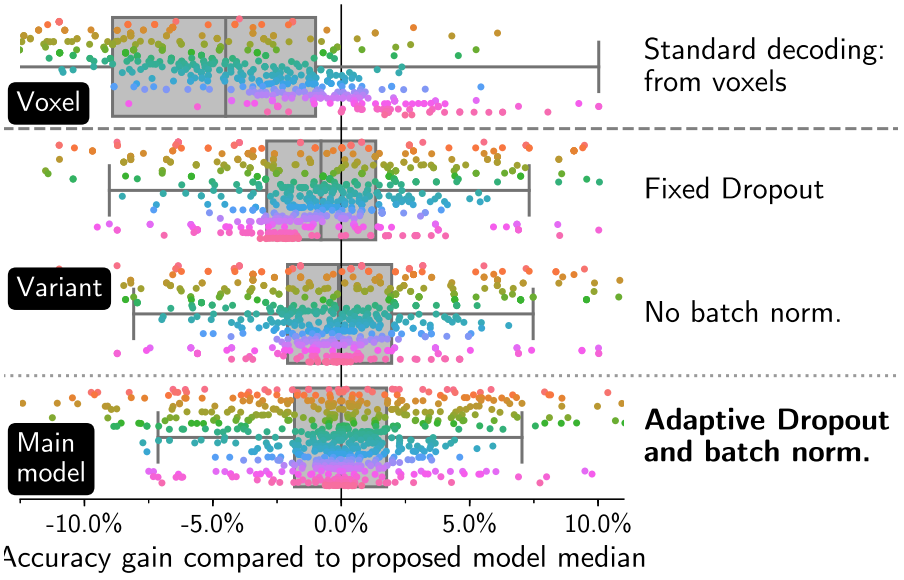


Figure 7.11 – Batch normalization and adaptive variational dropout both have a beneficial impact on classification accuracy of the final learned decoder.

We introduced variational dropout and batch normalization in the training procedure of our algorithm. Figure 7.11 shows that is indeed beneficial. Variational dropout allows a gain of +1% compared to baseline; batch normalization benefit is smaller but positive, and

allows faster training — in line with its original purpose (Ioffe and Szegedy, 2015).

7.5.5 *Study weights*

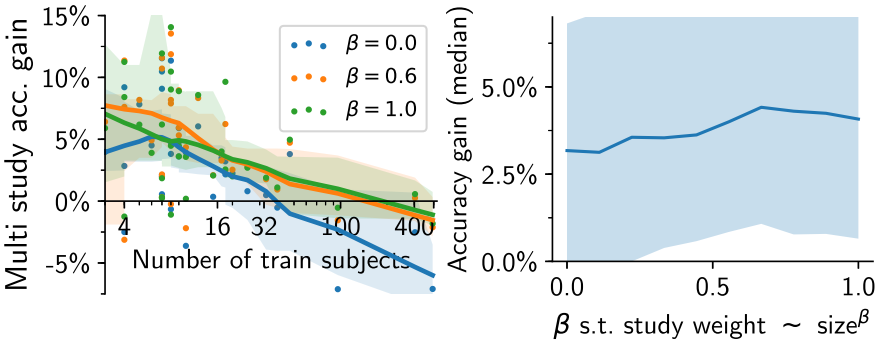


Figure 7.12 – Impact of changing the study weight in the joint objective. Large studies should be given more weights ($\beta \to 1$) to prevent negative transfer learning. Yet using an intermediary $\beta \approx 0.6$ (*i. e.* giving more weight to samples from small studies) is beneficial for performance on studies with less than 10 subjects.

Our model learns the second and third layer weights by solving

$$\min_{\substack{L\in\mathbb{R}^{l\times k}\\(U^j,b^j)_j}} -\sum_{j=1}^{N}\frac{(n^j)^\beta}{n^j}\sum_{i=1}^{n^j}\Big(l^j_{i,y_i}(U^j,b^j,L)-\log\big(\sum_{k=1}^{c^j}\exp l^j_{i,k}(U^j,b^j,L)\big)\Big),$$

in which the many studies can be given various weights. At one extreme, we may consider that all studies of the corpus should be weighted the same, which amounts to setting $\beta = 0$ in (7.4). At the opposite, we can consider that each brain map from each study should have the same importance, which amount to set $\beta = 1$. As Figure 7.12b shows, it is beneficial to set an intermediary β, typically $\beta = 0.6$. On the one hand, we do want to give the smallest study of our corpus a non negligible importance; on the other hand, we wish large corpus, that provide a greater amount of information, to remain more weighted than smaller ones. Our amortized reweighting amounts to give every study j an "effective sample size"

$$n^j_{\text{eff}} = \sum_{i=1}^{N} n^i \frac{{n^j}^\beta}{\sum_{i=1}^{N} {n^i}^\beta},$$

that is larger than the true sample size for smaller studies and smaller for larger studies. We observe on Figure 7.12a that the negative transfer learning endured by large-study decoders such as HCP and LA5C reduces as these studies are given more weight ($\beta \to 1$). On the other hand, the performance on small datasets slightly reduces for $\beta > 0.6$. It also reduces for low β, hinting at the importance of large studies for improving small studies decoding.

We thus have provided justifications for all the technical design choices made in training our decoding model: regularization, joint training, training refinements, choice of study weights.

7.6 DATA CORPUS AND REFERENCES

In this last section, we detail our experiment pipeline, the numerical parameters needed for reproducing this study, and the sources from which we obtained our corpus of studies.

7.6.1 *Reproduction details*

We used *nilearn* (Abraham et al., 2014) and *scikit-learn* (Pedregosa et al., 2011) in our experiment pipelines, the SOMF algorithm (see Chapter 6) for learning resting state dictionary and *PyTorch* (Paszke et al., 2017) for model design and training. A Python package[3] is available for reproducibility and reuse. It provides the multi-scale resting-state dictionaries extracted from HCP, as those are costly to learn.

GENERAL CROSS-VALIDATION SCHEME. For every validation experiment and comparison, we perform 20 half-split of all data. Namely, we consider half of the subjects of every study for training, and test the decoder on the other half. As two studies (Rizk-Jackson et al., n.d.) share subjects, we also ensure that no single subject appears in both the training and the test sets across studies.

BASELINE PARAMETER SELECTION. We cross validate the λ parameter on a grid $\{10^i, i = [-3, 3]\}$.

DROPOUT RATE. We use a dropout rate of $p = 0.25$ in between the first and second layer and initialize study-specific dropout rates with $p = 0.75$ in between the second and third-layer classification heads (*i. e.* we set $\alpha = \frac{p}{1-p}$ in variational dropout).

RESTING-STATE DICTIONARIES. We obtain the 512-components and 128-components resting-state dictionaries by choosing λ on a grid $\{10^i, i = [-5, 1]\}$, so that we obtain components that cover the whole brain with minimal overlapping.

CONSENSUS PHASE. We run the training procedure 100 times with different random seeds. We set $\lambda = 10^{-4}$, so as to obtain 80% sparsity.Higher sparsity leads to a slight decrease in performance, lower sparsity is softer on symmetry breaking, which may reduce interpretability. This parameter has little influence.

3. github.com/arthurmensch/cogspaces

7.6.2 *Task fMRI studies*

Table 7.1 recapitulates the various studies used in our corpus and provide their sources. The names corresponds to the ones used in Figure 7.2.

Task name	Source study
High-level math	Amalric and Dehaene (2016)
Localizer	Pinel et al. (2007a)
Brainomics localizer	Papadopoulos Orfanos et al. (2017)
CamCan audio-visual	Shafto et al. (2014)
Constit. struct. of sent. & music	Cauvet (2012) and Hara et al. (2009)
Sentence/music complexity	Devauchelle et al. (2009)
Balloon Analog Risk-taking	Schonberg et al. (2012)
Classification learning	Aron et al. (2006)
Rhyme judgment	G. Xue and Poldrack (2007)
Mixed-gambles	Tom et al. (2007)
Plain or mirror-reversed text	Jimura et al. (2014)
Stop-signal w/ spoken & manual resp.	G. Xue et al. (2008)
Stop-signal	Aron et al. (2007)
BART, stop-signal, emotion	Cohen (2009)
Weather prediction	Foerde et al. (2006)
Stop-signal & classification	Rizk-Jackson et al. (n.d.)
Stop-signal & classification (retake)	Rizk-Jackson et al. (n.d.)
Cross-language repetition priming	Alvarez et al. (2002)
Classif. learning & reversal	Poldrack et al. (2001)
Simon task	Kelly and M. Milham (n.d.)
Flanker task (event-related)	Kelly et al. (2008)
Visual object recognition	Haxby et al. (2001a)
Word & object processing	Duncan et al. (2009)
Emotion regulation	T. D. Wager et al. (2008)
False belief	Moran et al. (2012)
Incidental encoding	Uncapher et al. (2011)
Motor task & word/verb generation	Gorgolewski et al. (2013)
Auditory & Visual Oddball	Collier et al. (2014)
Spatio-temporal judgement	Gauthier et al. (2012)
Spatio-temporal judgement (retake)	Gauthier et al. (2012)
The Human Connectome Project	Barch et al. (2013)
Face recognition	Henson et al. (2011)
Arithmetic & saccades	Knops et al. (2009)
UCLA LA5C	Poldrack et al. (2016)
Twin localizer	Pinel and Dehaene (2013)
Compression	Vagharchakian et al. (2012)

Table 7.1 – Studies used in our corpus

Part IV

NEW ALGORITHMIC LAYERS FOR DEEP STRUCTURE PREDICTION

8

DIFFERENTIABLE DYNAMIC PROGRAMMING FOR STRUCTURED PREDICTION AND ATTENTION

In this chapter, we depart from fMRI, the main domain of application of this thesis, and consider the problem of learning rich representations for *structured prediction*. We present a general approach for performing end-to-end training of (potentially deep) models that involves predicting structured entities, *i.e.* objects that belongs to combinatorial sets, through the application of dynamic programming algorithms. The work presented in this chapter is the result of an four months collaboration with Mathieu Blondel, in NTT Communication Laboratories, Kyoto, Japan. It was recently presented and published under the title

> Mensch, A., & Blondel, M. (2018). Differentiable dynamic programming for structured prediction and attention. *Proceedings of the International Conference on Machine Learning (ICML)*

It possesses some connections with concepts studied in the previous chapters. First, we show how to introduce *sparsity* in structured prediction, and transform single output prediction models into models that predict a small set of possible structures. Second, we show how to minimize a range of objective functions that involves computing *maximizers*, similar to what we did when designing the SOMF algorithm.

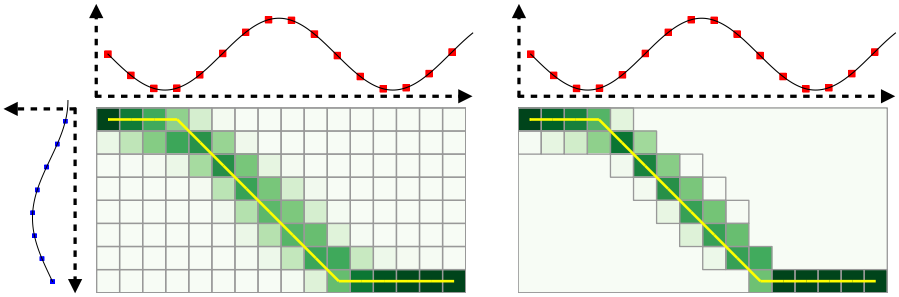
Dynamic programming (DP) constitutes the starting point of this chapter. It solves a variety of structured combinatorial problems by iteratively breaking them down into smaller sub-problems. In spite of their versatility, many DP algorithms are non-differentiable, which hampers their use as a layer in neural networks trained by backpropagation. To address this issue, we propose to smooth the max operator in the dynamic programming recursion, using a strongly convex regularizer. This allows to relax both the optimal value and solution of the original combinatorial problem, and turns a broad class of DP algorithms into differentiable operators. Theoretically, we provide a new probabilistic perspective on backpropagating through these DP operators, and relate them to inference in graphical models. We derive two particular instantiations of our framework, a smoothed Viterbi algorithm for sequence prediction and a smoothed DTW algorithm for time-series alignment. We showcase these instantiations on structured prediction (audio-to-score alignment, NER) and on structured and sparse attention for translation.

8.1 INTRODUCTION

Modern neural networks are composed of multiple layers of nested functions. Although layers usually consist of of elementary linear algebraic operations and simple non-linearities, there is a growing need for layers that output the *value* or the *solution* of an optimization problem. This can be used to design loss functions that capture relevant regularities in the input (Cuturi and Blondel, 2017; Lample et al., 2016) or to create layers that impose prior structure on the output (Amos and Kolter, 2017; Djolonga and Krause, 2017; Y. Kim et al., 2017; Niculae and Blondel, 2017).

Among these works, several involve a convex optimization problem (Amos and Kolter, 2017; Djolonga and Krause, 2017; Niculae and Blondel, 2017); others solve certain combinatorial optimization problems by dynamic programming (Cuturi and Blondel, 2017; Y. Kim et al., 2017; Nowak et al., 2018). However, because dynamic programs (Bellman, 1952) are usually non-differentiable, virtually all these works resort to the formalism of conditional random fields (CRFs) (Lafferty et al., 2001), which can be seen as changing the semiring used by the dynamic program — replacing all values by their exponentials and all $(\max, +)$ operations with $(+, \times)$ operations (Verdu and Poor, 1987). While this modification smoothes the dynamic program, it looses the sparsity of solutions, since hard assignments become soft ones. Moreover, a general understanding of how to relax and differentiate dynamic programs is lacking. We propose to do so by leveraging smoothing (Moreau, 1965; Nesterov, 2005) and backpropagation (Linnainmaa, 1970). We make the following contributions.

1. We present a unified framework for turning a broad class of dynamic programs into differentiable operators. Unlike existing works, we propose to change the semiring to use $(\max_\Omega, +)$ operations, where $\max_\Omega$ is a max operator smoothed with a strongly convex regularizer Ω (Section 8.2).
2. We show that the resulting DP operators, that we call DP_Ω, are smoothed relaxations of the original DP algorithm and satisfy several key properties, chief among them convexity. In addition, we show that their gradient, $\nabla\mathrm{DP}_\Omega$, is equal to the *expected trajectory* of a certain random walk and can be used as a sound relaxation to the original dynamic program's solution. Using negative entropy for Ω recovers existing CRF-based works from a different perspective — we provide new arguments as to why this Ω is a good choice. On the other hand, using squared ℓ_2 norm for Ω leads to new algorithms whose expected solution is *sparse*. We derive a clean and efficient method to backpropagate gradients, both through DP_Ω and $\nabla\mathrm{DP}_\Omega$. This allows us to define differentiable DP layers that can be incorporated in neural networks trained end-to-end (Section 8.3).
3. We illustrate how to to derive two particular instantiations of our framework, a smoothed Viterbi algorithm for sequence pre-



$\mathrm{DTW}_{-H}(\boldsymbol{\theta}) = -7.49$ $\mathrm{DTW}_{\|\cdot\|^2}(\boldsymbol{\theta}) = 9.61$

Figure 8.1 – $\mathrm{DTW}_\Omega(\theta)$ is an instantiation of the proposed smoothed dynamic programming operator, $\mathrm{DP}_\Omega(\theta)$, to the DTW computational graph. In this picture, θ is the squared Euclidean distance matrix between the observations of two time-series. The gradient $\nabla\mathrm{DTW}_\Omega(\theta)$ is equal to the expected alignment under a certain random walk characterized in Section 8.3.3 and is a sound continuous relaxation to the hard DTW alignment between the two time-series (here depicted with a yellow path). Unlike negentropy regularization (left), ℓ_2^2 regularization leads to exactly sparse alignments (right). Our framework allows to backpropagate through both $\mathrm{DTW}_\Omega(\theta)$ and $\nabla\mathrm{DTW}_\Omega(\theta)$, which makes it possible to learn the distance matrix θ end-to-end.

diction and a smoothed DTW algorithm for supervised time-series alignment (Section 8.4). The latter is illustrated in Figure 8.1. Finally, we showcase these two instantiations on structured prediction tasks (Section 8.5) and on structured attention for neural machine translation (Section 8.6).

Notation

We denote scalars, vectors and matrices using lower-case, bold lower-case and bold upper-case letters, *e.g.*, y, $\mathbf{y}$ and $\mathbf{Y}$. We denote the elements of $\mathbf{Y}$ by $y_{i,j}$ and its rows by $\mathbf{y}_i$. We denote the Frobenius inner product between $\mathbf{A}$ and $\mathbf{B}$ by $\langle \mathbf{A}, \mathbf{B} \rangle \triangleq \sum_{i,j} a_{i,j} b_{i,j}$. We denote the $(D-1)$-probability simplex by $\triangle^D \triangleq \{\boldsymbol{\lambda} \in \mathbb{R}_+^D : \|\boldsymbol{\lambda}\|_1 = 1\}$. We write $\mathrm{conv}(\mathcal{Y}) \triangleq \{\sum_{\mathbf{Y} \in \mathcal{Y}} \lambda_{\mathbf{Y}} \mathbf{Y} : \boldsymbol{\lambda} \in \triangle^{|\mathcal{Y}|}\}$ the convex hull of $\mathcal{Y}$, $[N]$ the set $\{1, \dots, N\}$ and $\mathrm{supp}(\mathbf{x}) \triangleq \{j \in [D] : x_j \neq 0\}$ the support of $\mathbf{x} \in \mathbb{R}^D$. We denote the Shannon entropy by $H(\mathbf{q}) \triangleq \sum_i q_i \log q_i$.

We have released an optimized and modular *PyTorch* implementation [1] for reproduction and reuse.

8.2 SMOOTHED MAX OPERATORS

In this section, we introduce smoothed max operators (Beck and Teboulle, 2012; Nesterov, 2005; Niculae and Blondel, 2017), that will serve as a powerful and generic abstraction to define differentiable dynamic programs in Section 8.3. Let $\Omega : \mathbb{R}^D \to \mathbb{R}$ be a strongly

1. https://github.com/arthurmensch/didyprog

convex function on $\triangle^D$ and let $\mathbf{x} \in \mathbb{R}^D$. We define the max operator smoothed by Ω as:

$$\max_\Omega(\mathbf{x}) \triangleq \max_{\mathbf{q} \in \triangle^D} \langle \mathbf{q}, \mathbf{x} \rangle - \Omega(\mathbf{q}). \tag{8.1}$$

In other words, $\max_\Omega$ is the convex conjugate of Ω, restricted to the simplex. From the duality between strong convexity and smoothness, $\max_\Omega$ is smooth: differentiable everywhere and with Lipschitz continuous gradient. Since the argument that achieves the maximum in (8.1) is unique, from Danskin's theorem (1966), it is equal to the gradient:

$$\nabla \max_\Omega(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{q} \in \triangle^D} \langle \mathbf{q}, \mathbf{x} \rangle - \Omega(\mathbf{q}).$$

The gradient is differentiable almost everywhere for any strongly-convex Ω (everywhere for negentropy). Next, we state properties that will be useful throughout this chapter.

Lemma 8.1. *Properties of* $\max_\Omega$ *operators*
Let $\mathbf{x} = (x_1, \ldots, x_D)^\top \in \mathbb{R}^D$.

1. *Boundedness: If* Ω *is lower-bounded by* $L_{\Omega,D}$ *and upper-bounded by* $U_{\Omega,D}$ *on the simplex* $\triangle^D$*, then*
   $\max(\mathbf{x}) - U_{\Omega,D} \leqslant \max_\Omega(\mathbf{x}) \leqslant \max(\mathbf{x}) - L_{\Omega,D}$.
2. *Distributivity of* $+$ *over* $\max_\Omega$*:*
   $\max_\Omega(\mathbf{x} + c\mathbf{1}) = \max_\Omega(\mathbf{x}) + c \quad \forall c \in \mathbb{R}$.
3. *Commutativity: If* $\Omega(\mathbf{P}\mathbf{q}) = \Omega(\mathbf{q})$*, where* $\mathbf{P}$ *is a permutation matrix, then* $\max_\Omega(\mathbf{P}\mathbf{x}) = \max_\Omega(\mathbf{x})$.
4. *Non-decreasingness in each coordinate:*
   $\max_\Omega(\mathbf{x}) \leqslant \max_\Omega(\mathbf{y}) \quad \forall \mathbf{x} \leqslant \mathbf{y}$
5. *Insensitivity to* $-\infty$*:* $x_j = -\infty \Rightarrow \nabla \max_\Omega(\mathbf{x})_j = 0$.

Proofs are given in Section B.1.1. In particular, property 3 holds whenever $\Omega(\mathbf{q}) = \sum_{i=1}^D \omega(q_i)$, for some function ω. In this chapter, we focus on two specific regularizers Ω: the negentropy $-H$ and the squared ℓ_2 norm. For these choices, all properties above are satisfied and we can derive closed-form expressions for $\max_\Omega$, its gradient and its Hessian — see Section B.2.1. When using negentropy, $\max_\Omega$ becomes the *log-sum-exp* and $\nabla \max_\Omega$ the *softmax*. The former satisfies associativity, which as we shall see, makes it natural to use in dynamic programming. With the squared ℓ_2 regularization, as observed by Martins and Astudillo (2016) and Niculae and Blondel (2017), the gradient $\nabla \max_\Omega$ is *sparse*. This will prove useful to enforce sparsity in the models we study.

8.3 DIFFERENTIABLE DYNAMIC PROGRAMMING LAYERS

DP is a generic way of solving combinatorial optimization problems by recursively solving problems on smaller sets. We first introduce this category of algorithms in a broad setting, then use smoothed max operators to define differentiable DP layers.

8.3.1 *Dynamic programming on a DAG*

Every problem solved by dynamic programming reduces to finding the highest-scoring path between a start node and an end node, on a weighted directed acyclic graph (DAG). We therefore introduce our formalism on this generic problem, and give concrete examples in Section 8.4.

Formally, let $G = (\mathcal{V}, \mathcal{E})$ be a DAG, with nodes $\mathcal{V}$ and edges $\mathcal{E}$. We write $N = |\mathcal{V}| \geqslant 2$ the number of nodes. Without loss of generality, we number the nodes in topological order, from 1 (start) to N (end), and thus $\mathcal{V} = [N]$. Node 1 is the only node without parents, and node N the only node without children. Every directed edge (i, j) from a parent node j to a child node i has a weight $\theta_{i,j} \in \mathbb{R}$. We gather the edge weights in a matrix $\theta \in \boldsymbol{\Theta} \subseteq \mathbb{R}^{N \times N}$, setting $\theta_{i,j} = -\infty$ if $(i, j) \notin \mathcal{E}$ and $\theta_{1,1} = 1$. We consider the set $\mathcal{Y}$ of all paths in G from node 1 to node N. Any path $\mathbf{Y} \in \mathcal{Y}$ can be represented as a $N \times N$ binary matrix, with $y_{i,j} = 1$ if the path goes through the edge (i, j) and $y_{i,j} = 0$ otherwise. In the sequel, paths will have a one-to-one correspondence with discrete structures such as sequences or alignments. Using this representation, $\langle \mathbf{Y}, \theta \rangle$ corresponds to the cumulated sum of edge weights, along the path $\mathbf{Y}$. The computation of the *highest score* among all paths amounts to solving the combinatorial problem

$$\mathrm{LP}(\theta) \triangleq \max_{\mathbf{Y} \in \mathcal{Y}} \langle \mathbf{Y}, \theta \rangle \in \mathbb{R}. \tag{8.2}$$

Although the size of $\mathcal{Y}$ is in general exponential in N, $\mathrm{LP}(\theta)$ can be computed in one topologically-ordered pass over G using *dynamic programming*. We let $\mathcal{P}_i$ be the set of parent nodes of node i in graph G and define recursively

$$\begin{aligned} v_1(\theta) &\triangleq 0 \\ \forall i \in [2, \dots, N] : \; v_i(\theta) &\triangleq \max_{j \in \mathcal{P}_i} \theta_{i,j} + v_j(\theta). \end{aligned} \tag{8.3}$$

This algorithm outputs $\mathrm{DP}(\theta) \triangleq v_N(\theta)$. We now show that this is precisely the highest score among all paths.

Proposition 8.1. *Optimality of dynamic programming*

$$\forall \theta \in \boldsymbol{\Theta} : \quad \mathrm{DP}(\theta) = \mathrm{LP}(\theta)$$

The optimality of recursion (8.3) is well-known (Bellman, 1952). We prove it again with our formalism in Section B.1.2, since it exhibits the two key properties that the max operator must satisfy to guarantee optimality: *distributivity* of $+$ over it and *associativity*. The cost of computing $\mathrm{DP}(\theta)$ is $\mathcal{O}(|\mathcal{E}|)$, which is exponentially better than $\mathcal{O}(|\mathcal{Y}|)$.

In many applications, we will often rather be interested in the *argument* that achieves the maximum, *i.e.*, one of the highest-scoring paths

$$\mathbf{Y}^\star(\theta) \in \operatorname*{argmax}_{\mathbf{Y} \in \mathcal{Y}} \langle \mathbf{Y}, \theta \rangle. \tag{8.4}$$

This argument can be computed by *backtracking*, that we now relate to computing subgradients of $\mathrm{LP}(\theta)$.

LINEAR PROGRAM, LACK OF DIFFERENTIALITY. Unfortunately, the linear program value $\mathrm{LP}(\theta)$ is not differentiable everywhere w.r.t. θ. To see why this is the case, notice that (8.2) can be rewritten as a linear program over the convex polytope $\mathrm{conv}(\mathcal{Y})$:

$$\mathrm{LP}(\theta) = \max_{Y \in \mathrm{conv}(\mathcal{Y})} \langle Y, \theta \rangle.$$

From the generalized Danskin theorem (Bertsekas, 1971),

$$Y^\star(\theta) \in \partial \mathrm{LP}(\theta) = \operatorname*{argmax}_{Y \in \mathrm{conv}(\mathcal{Y})} \langle Y, \theta \rangle,$$

where ∂ denotes the subdifferential of $\mathrm{LP}(\theta)$, *i.e.*, the set of subgradients. When $Y^\star(\theta)$ is unique, $\partial \mathrm{LP}(\theta)$ is a singleton and $Y^\star$ is equal to the gradient of $\mathrm{LP}(\theta)$, that we write $\nabla \mathrm{LP}(\theta)$. Unfortunately, $Y^\star(\theta)$ is not always unique, meaning that $\mathrm{LP}(\theta)$ is not differentiable everywhere. As we will show in Section 8.5.2, this hinders optimization as we can only train models involving $\mathrm{LP}(\theta)$ with *subgradient* methods. Worse, $Y^\star(\theta)$, a function from Θ to $\mathcal{Y}$, is discontinuous and has null or undefined derivatives. It is thus impossible to use it in a model trained by gradient descent.

8.3.2 *Smoothed max layers*

To address the lack of differentiability of dynamic programming, we introduce the operator $\max_\Omega$, presented in Section 8.2, and consider two approaches.

SMOOTHING THE LINEAR PROGRAM. Let us define the Ω-smoothed maximum of a function $f\colon \mathcal{Y} \to \mathbb{R}$ over a finite set $\mathcal{Y}$ using the following shorthand notation:

$$\max_{Y \in \mathcal{Y}}{}_\Omega \; f(Y) \triangleq \max_\Omega((f(Y))_{Y \in \mathcal{Y}}).$$

A natural way to circumvent the lack of differentiability of $\mathrm{LP}(\theta)$ is then to replace the *global* max operator by $\max_\Omega$:

$$\mathrm{LP}_\Omega(\theta) \triangleq \max_{Y \in \mathcal{Y}}{}_\Omega \; \langle Y, \theta \rangle \in \mathbb{R}. \tag{8.5}$$

From Section 8.2, $\mathrm{LP}_\Omega(\theta)$ is convex and, as long as Ω is strongly convex, differentiable everywhere. In addition, $\nabla \mathrm{LP}_\Omega(\theta)$ is Lipschitz continuous and thus differentiable almost everywhere. Unfortunately, solving (8.5) for general Ω is likely intractable when $\mathcal{Y}$ has an exponential size.

SMOOTHING THE DYNAMIC PROGRAM. As a tractable alternative, we propose an *algorithmic* smoothing. Namely, we replace max by $\max_\Omega$ *locally* within the DP recursion. Omitting the dependence on Ω, this defines a smoothed recursion over the new sequence $(v_i(\theta))_{i=1}^N$:

$$\begin{aligned} &v_1(\theta) \triangleq 0 \\ &\forall i \in [2, \dots, N] : \; v_i(\theta) \triangleq \max_{j \in \mathcal{P}_i}{}_\Omega \; \theta_{i,j} + v_j(\theta). \end{aligned} \tag{8.6}$$

The new algorithm outputs $\mathrm{DP}_\Omega(\theta) \triangleq v_N(\theta)$, the *smoothed highest score*. Smoothing the max operator locally brings the same benefit as before — $\mathrm{DP}_\Omega(\theta)$ is smooth and $\nabla\mathrm{DP}_\Omega(\theta)$ is differentiable almost everywhere. However, computing $\mathrm{DP}_\Omega(\theta)$ is now always tractable, since it simply requires to evaluate $(v_i(\theta))_{i=1}^N$ in topological order, as in the original recursion (8.3). Although $\mathrm{LP}_\Omega(\theta)$ and $\mathrm{DP}_\Omega(\theta)$ are generally different (in fact, $\mathrm{LP}_\Omega(\theta) \geqslant \mathrm{DP}_\Omega(\theta)$ for all $\theta \in \boldsymbol{\Theta}$), we now show that $\mathrm{DP}_\Omega(\theta)$ is a sensible approximation of $\mathrm{LP}(\theta)$ in several respects.

Proposition 8.2. *Properties of DP_Ω*

1. *$DP_\Omega(\theta)$ is convex*
2. *$LP(\theta) - DP_\Omega(\theta)$ is bounded above and below:*

$$(N-1)L_{\Omega,N} \geqslant LP(\theta) - DP_\Omega(\theta) \leqslant (N-1)U_{\Omega,N},$$

   with Lemma 8.1 notations.

3. *When Ω is separable, $DP_\Omega(\theta) = LP_\Omega(\theta)$ **if and only if** $\Omega = -\gamma H$, where $\gamma \geqslant 0$. In other word, the **only separable regularization** for which smoothing the dynamic program amounts to smoothing the whole associated linear program is the **negentropy**.*

Proofs are given in Section B.1.3. The first claim can be surprising due to the recursive definition of $\mathrm{DP}_\Omega(\theta)$. The second claim implies that $\mathrm{DP}_{\gamma\Omega}(\theta)$ converges to $\mathrm{LP}(\theta)$ when the regularization vanishes: $\mathrm{DP}_{\gamma\Omega}(\theta) \to_{\gamma\to 0} \mathrm{LP}(\theta)$; $\mathrm{LP}_{\gamma\Omega}(\theta)$ also satisfies this property. The "if" direction of the third claim follows by showing that $\max_{-\gamma H}$ satisfies associativity. This recovers known results in the framework of message passing algorithms for probabilistic graphical models (*e.g.*, M. J. Wainwright and Jordan, 2008, Section 4.1.3), with a more algebraic point of view. The key role that the distributive and associative properties play into breaking down large problems into smaller ones has long been noted (Aji and McEliece, 2000; Verdu and Poor, 1987). However, the "and only if" part of the claim is new to our knowledge. Its proof shows that $\max_{-\gamma H}$ is the only $\max_\Omega$ satisfying associativity, exhibiting a functional equation from information theory (Horibe, 1988). While this provides an argument in favor of entropic regularization, ℓ_2^2 regularization has different benefits in terms of sparsity of the solutions.

8.3.3 *Relaxed argmax layers*

It is easy to check that $\nabla\mathrm{LP}_\Omega(\theta)$ belongs to $\mathrm{conv}(\mathcal{Y})$ and can be interpreted as an expected path under some distribution induced by $\nabla\max_\Omega$, over all possible $\boldsymbol{Y} \in \mathcal{Y}$ — see Section B.1.4 for details. This makes $\nabla\mathrm{LP}_\Omega(\theta)$ interpretable as a *continuous relaxation* of the highest-scoring path $\boldsymbol{Y}^\star(\theta)$ defined in (8.4). However, like $\mathrm{LP}_\Omega(\theta)$, computing $\nabla\mathrm{LP}_\Omega(\theta)$ is likely intractable in the general case. Fortunately, $\nabla\mathrm{DP}_\Omega(\theta)$ is always easily computable by *backpropagation* and enjoys similar properties, as we now show.

COMPUTING $\nabla \mathrm{DP}_\Omega(\theta)$. Computing $\nabla \mathrm{DP}_\Omega(\theta)$ can be broken down into two steps. First, we compute and record the local gradients alongside the recursive step (8.6):

$$\forall\, i \in [N]: \quad q_i(\theta) \triangleq \nabla \max_\Omega(\theta_i + \nu(\theta)) \in \triangle^N,$$

where $\nu(\theta) \triangleq (\nu_1(\theta), \dots, \nu_N(\theta))$. Since we assume that $\theta_{i,j} = -\infty$ if $(i,j) \notin \mathcal{E}$, we have $\mathrm{supp}(q_i(\theta)) = \mathcal{P}_i$. This ensures that, similarly to $\nu_i(\theta)$, $q_i(\theta)$ exclusively depends on $(\nu_j(\theta))_{j \in \mathcal{P}_i}$. Let $\mathcal{C}_j$ be the children of node $j \in [N]$. A straighforward application of backpropagation (cf. Section B.1.5) yields a recursion run in *reverse-topological* order, starting from node $j = N - 1$ down to $j = 1$:

$$\forall\, i \in \mathcal{C}_j: \; e_{i,j} \leftarrow \bar{e}_i q_{i,j} \text{ then } \bar{e}_j \leftarrow \sum_{i \in \mathcal{C}_j} e_{i,j},$$

where $\bar{e}_N \leftarrow 1$ and $e_{i,j} \leftarrow 0$ for $(i,j) \notin \mathcal{E}$. The final output is $\nabla \mathrm{DP}_\Omega(\theta) = E$. Assuming $\max_\Omega$ can be computed in linear time, the total cost is $\mathcal{O}(|\mathcal{E}|)$, the same as $\mathrm{DP}(\theta)$. Pseudo-code is summarized in Section B.1.5.

ASSOCIATED PATH DISTRIBUTION. The backpropagation formulae we derived have a probabilistic interpretation. Indeed, $Q(\theta) \in \mathbb{R}^{N \times N}$ can be interpreted as a transition matrix: it defines a *random walk* on the graph G, *i.e.*, a finite Markov chain with states $\mathcal{V}$ and transition probabilities supported by $\mathcal{E}$. The random walk starts from node N and, when at node i, hops to node $j \in \mathcal{P}_i$ with probability $q_{i,j}$. It always ends at node 1, which is absorbing. The walk follows the path $Y \in \mathcal{Y}$ with a probability $p_{\theta,\Omega}(Y)$, which is simply the product of the $q_{i,j}$ of visited edges. Thus, $Q(\theta)$ defines a *path distribution* $p_{\theta,\Omega}$. Our next proposition shows that $\nabla \mathrm{DP}_\Omega(Y) \in \mathrm{conv}(\mathcal{Y})$ and is equal to the expected path $\mathbb{E}_{\theta,\Omega}[Y]$ under that distribution.

Proposition 8.3. *$\nabla DP_\Omega(\theta)$ as an expected path*

$$\forall\, \theta \in \Theta: \quad \nabla \mathrm{DP}_\Omega(\theta) = \mathbb{E}_{\theta,\Omega}[Y] = E \in \mathrm{conv}(\mathcal{Y}).$$

Proof is provided in Section B.1.5. Moreover, $\nabla \mathrm{DP}_\Omega(\theta)$ is a principled relaxation of the highest-scoring path $Y^\star(\theta)$, in the sense that it converges to a subgradient of $\mathrm{LP}(\theta)$ as the regularization vanishes: $\nabla \mathrm{DP}_{\gamma\Omega}(\theta) \xrightarrow[\gamma \to 0]{} Y^\star(\theta) \in \partial \mathrm{LP}(\theta)$. When $\Omega = -\gamma H$, the distributions underpinning $\mathrm{LP}_\Omega(\theta)$ and $\mathrm{DP}_\Omega(\theta)$ coincide and reduce to the Gibbs distribution $p_{\theta,\Omega}(Y) \propto \exp(\langle \theta, Y \rangle / \gamma)$. The value $\mathrm{LP}_\Omega(\theta) = \mathrm{DP}_\Omega(\theta)$ is then equal to the log partition. When $\Omega = \gamma \|\cdot\|^2$, some transitions between nodes have zero probability and hence some paths have zero probability under the distribution $p_{\theta,\Omega}$. Thus, $\nabla \mathrm{DP}_\Omega(\theta)$ is typically *sparse* — this will prove interesting to introspect the various models we consider (typically, the smaller γ, the sparser $\nabla \mathrm{DP}_\Omega(\theta)$).

8.3.4 *Multiplication with the Hessian $\nabla^2 DP_\Omega(\theta)Z$*

Using $\nabla \mathrm{DP}_\Omega(\theta)$ as a layer involves backpropagating through the gradient $\nabla \mathrm{DP}_\Omega(\theta)$. This requires to apply the *Jacobian* of $\nabla \mathrm{DP}_\Omega$ oper-

ator (a linear map from $\mathbb{R}^{N\times N}$ to $\mathbb{R}^{N\times N}$), or in other words to apply the *Hessian* of DP_Ω, to an input sensibility vector $\mathbf{Z}$, computing

$$\nabla^2\mathrm{DP}_\Omega(\theta)\mathbf{Z} = \nabla\langle\nabla\mathrm{DP}_\Omega(\theta), \mathbf{Z}\rangle \in \mathbb{R}^{N\times N},$$

where derivatives are w.r.t. θ. The above vector may be computed in two ways, that differ in the order in which derivatives are computed. Using automatic differentiation frameworks such as *PyTorch* (Paszke et al., 2017), we may backpropagate over the computational graph a first time to compute the gradient $\nabla\mathrm{DP}_\Omega(\theta)$, while recording operations. We may then compute $\langle\nabla\mathrm{DP}_\Omega(\theta), \mathbf{Z}\rangle$, and backpropagate once again. However, due to the structure of the problem, it proves more efficient, adapting Pearlmutter's approach (1994), to directly compute $\langle\nabla\mathrm{DP}_\Omega(\theta), \mathbf{Z}\rangle \in \mathbb{R}$, namely, the *directional derivative* at θ along $\mathbf{Z}$. This is done by applying the chain rule in one topologically-ordered pass over G. Similarly to the gradient computation, we record products with the *local* Hessians $\mathbf{H}_i(\theta) \triangleq \nabla^2\max_\Omega(\theta_i + \nu(\theta))$ along the way. We then compute the gradient of the directional derivative using backpropagation. This yields a recursion for computing $\nabla^2\mathrm{DP}_\Omega(\theta)\mathbf{Z}$ in reverse topological-order over G. The complete derivation and the pseudo-code are given in Section B.1.7. They allow to implement DP_Ω as as a custom twice-differentiable module in existing software. For both approaches, the computational cost is $\mathcal{O}(|\mathcal{E}|)$, the same as for gradient computation. In our experiments in Section 8.5.2, our custom Hessian-vector product computation brings a $3\times/12\times$ speed-up during the backward pass on GPU/CPU *vs.* automatic differentiation.

RELATED WORKS. Smoothing LP formulations was also used for MAP inference (Meshi et al., 2015) or optimal transport (Blondel et al., 2018) but these works do not address how to differentiate through the smoothed formulation. An alternative approach to create structured prediction layers, fundamentally different both in the forward and backward passes, is SparseMAP (Niculae et al., 2018).

SUMMARY. We have constructed the operator $\mathrm{DP}_\Omega(\theta)$, a smooth, convex and tractable relaxation to the *value* of $\mathrm{LP}(\theta)$. We have also shown that $\nabla\mathrm{DP}_\Omega(\theta)$ belongs to $\mathrm{conv}(\mathcal{Y})$ and is therefore a sound relaxation to *solutions* of $\mathrm{LP}(\theta)$. To conclude this section, we formally define our proposed two layers.

Definition 8.1. *Differentiable dynamic programming layers*

$$\begin{aligned} \textit{Value layer:} \quad & DP_\Omega(\theta) \in \mathbb{R} \\ \textit{Gradient layer:} \quad & \nabla DP_\Omega(\theta) \in \mathrm{conv}(\mathcal{Y}) \end{aligned}$$

8.4 EXAMPLES OF COMPUTATIONAL GRAPHS

We now illustrate two instantiations of our framework for specific computational graphs.

8.4.1 *Sequence prediction*

We demonstrate in this section how to instantiate DP_Ω to the computational graph of the Viterbi algorithm (Rabiner, n.d.; Viterbi, 1967), one of the most famous instances of DP algorithm. We call the resulting operator Vit_Ω. We wish to tag a sequence $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_T)$ of vectors in $\mathbb{R}^D$ (*e.g.*, word representations) with the most probable output sequence (*e.g.*, entity tags) $\mathbf{y} = (y_1, \dots, y_T) \in [S]^T$. This problem can be cast as finding the highest-scoring path on a *treillis* G. While $\mathbf{y}$ can always be represented as a sparse $N \times N$ binary matrix, it is convenient to represent it instead as a $T \times S \times S$ binary tensor $\mathbf{Y}$, such that $y_{t,i,j} = 1$ if $\mathbf{y}$ transitions from node j to node i on time t, and 0 otherwise — we set $y_0 = 1$. The potentials can similarly be organized as a $T \times S \times S$ real tensor, such that $\theta_{t,i,j} = \phi_t(\mathbf{x}_t, i, j)$. Traditionally, the potential functions ϕ_t were human-engineered (Sutton, McCallum, et al., 2012, Section 2.5). In recent works and in our approach, they are learned end-to-end (Bottou et al., 1997; Collobert et al., 2011; Lample et al., 2016).

Using the above binary tensor representation, the inner product $\langle \mathbf{Y}, \theta \rangle$ is equal to $\sum_{t=1}^{T} \phi_t(\mathbf{x}_t, y_t, y_{t-1})$, $\mathbf{y}$'s cumulated score. This is illustrated in Figure 8.2 on the task of part-of-speech tagging. The bold arrows indicate one possible output sequence $\mathbf{y}$, *i.e.*, one possible path in G.

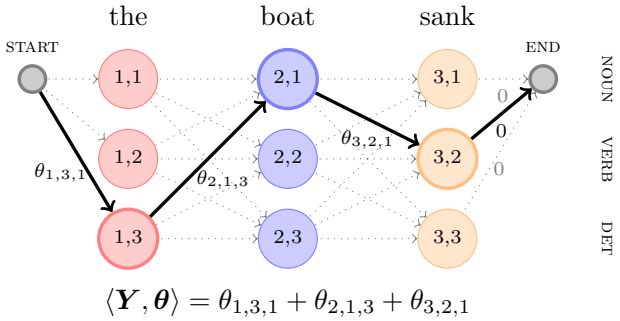


Figure 8.2 – Computational graph of the Viterbi algorithm.

When $\Omega = -\mathrm{H}$, we recover linear-chain conditional random fields (CRF) (Lafferty et al., 2001) and the probability of $\mathbf{y}$ ($\mathbf{Y}$ in tensor representation) given $\mathbf{X}$ is

$$p_{\theta,-\mathrm{H}}(\mathbf{y}|\mathbf{X}) \propto \exp(\langle \mathbf{Y}, \theta \rangle) = \exp\Big(\sum_{t=1}^{T} \phi_t(\mathbf{x}_t, y_t, y_{t-1})\Big). \tag{8.7}$$

From Prop. 8.3, the gradient $\nabla \mathrm{Vit}_{-\mathrm{H}}(\theta) = \mathbf{E} \in \mathbb{R}^{T \times S \times S}$ is such that $e_{t,i,j} = p_{\theta,-\mathrm{H}}(y_t = i, y_{t-1} = j|\mathbf{X})$. The marginal probability of state i at time t is simply $p_{\theta,-\mathrm{H}}(y_t = i|\mathbf{X}) = \sum_{j=1}^{S} e_{t,i,j}$. Using a different Ω simply changes the distribution over state transitions. When $\Omega = \|\cdot\|^2$, the marginal probabilities are typically *sparse*. Pseudo-code for $\mathrm{Vit}_\Omega(\theta)$, as well as gradient and Hessian-product computations, is provided in Section B.2.2. The case $\Omega = \|\cdot\|^2$ is new to our knowledge.

When $\Omega = -\mathrm{H}$, the marginal probabilities are traditionally computed using the forward-backward algorithm (Baum and Petrie, 1966). In contrast, we compute $\nabla\mathrm{Vit}_{-\mathrm{H}}(\theta)$ using backpropagation while efficiently maintaining the marginalization. An advantage of our approach is that all operations are numerically stable. The relation between forward-backward and backpropagation has been noted before (*e.g.*, Eisner (2016)). However, the analysis is led using $(+, \times)$ operations, instead of $(\max_\Omega, +)$ as we do. Our Viterbi instantiation can be generalized to graphical models with a tree structure, and to approximate inference in general graphical models, since unrolled loopy belief propagation (Pearl, 1988) yields a dynamic program. We note that continuous beam search Goyal et al., 2017 can also be cleanly rewritten and extended using Vit_Ω operators.

8.4.2 *Time-series alignment*

We now demonstrate how to instantiate DP_Ω to the computational graph of DTW (Sakoe and Chiba, 1978), whose goal is to seek the *minimal* cost alignment between two time-series. We call the resulting operator DTW_Ω. Formally, let $\mathrm{N_A}$ and $\mathrm{N_B}$ be the lengths of two time-series, $\mathbf{A}$ and $\mathbf{B}$. Let $\mathbf{a}_i$ and $\mathbf{b}_j$ be the i^{th} and j^{th} observations of $\mathbf{A}$ and $\mathbf{B}$, respectively. Since edge weights only depend on child nodes, it is convenient to rearrange $\mathbf{Y}$ and θ as $\mathrm{N_A} \times \mathrm{N_B}$ matrices. Namely, we represent an alignment $\mathbf{Y}$ as a $\mathrm{N_A} \times \mathrm{N_B}$ binary matrix, such that $y_{i,j} = 1$ if $\mathbf{a}_i$ is aligned with $\mathbf{b}_j$, and 0 otherwise. Likewise, we represent θ as a $\mathrm{N_A} \times \mathrm{N_B}$ matrix. A classical example is $\theta_{i,j} = \mathrm{d}(\mathbf{a}_i, \mathbf{b}_j)$, for some differentiable discrepancy measure d. We write $\mathcal{Y}$ the set of all monotonic alignment matrices, such that the path that connects the upper-left $(1,1)$ matrix entry to the lower-right $(\mathrm{N_A}, \mathrm{N_B})$ one uses only $\downarrow, \rightarrow, \searrow$ moves. The DAG associated with $\mathcal{Y}$ is illustrated in Figure 8.3 with $\mathrm{N_A} = 4$ and $\mathrm{N_B} = 3$ below.

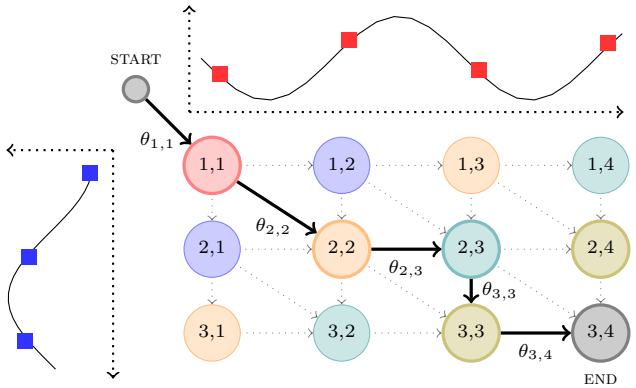


Figure 8.3 – Computational graph of the DTW algorithm.

Again, the bold arrows indicate one possible path $\mathbf{Y} \in \mathcal{Y}$ from start to end in the DAG, and correspond to one possible alignment. Using this representation, the cost of an alignment (cumulated cost along the path) is conveniently computed by $\langle \mathbf{Y}, \theta \rangle$. The value $\mathrm{DTW}_\Omega(\theta)$

can be used to define a loss between alignments or between time-series. Following Proposition 8.3, $\nabla \mathrm{DTW}_\Omega(\theta) = \mathbf{E} \in \mathbb{R}^{N_A \times N_B}$ can be understood as a soft alignment matrix. This matrix is sparse when $\Omega = \|\cdot\|^2$, as illustrated in Figure 8.1 (right).

Pseudo-code to compute $\mathrm{DTW}_\Omega(\theta)$ as well as its gradient and its Hessian products are provided in Section B.2.3. When $\Omega = -\mathrm{H}$, the operator $\mathrm{DTW}_\Omega(\theta)$ defines a conditional random field known as soft-DTW, and the probability $p_{\theta,\Omega}(\mathbf{Y}|\boldsymbol{A}, \mathbf{B})$ is a Gibbs distribution similar to (8.7) (Cuturi and Blondel, 2017). The case $\Omega = \|\cdot\|^2$ and the computation of $\nabla^2 \mathrm{DTW}_\Omega(\theta)\mathbf{Z}$ are new and allow new applications.

8.5 DIFFERENTIABLE STRUCTURED PREDICTION

We now apply the proposed layers, $\mathrm{DP}_\Omega(\theta)$ and $\nabla \mathrm{DP}_\Omega(\theta)$, to structured prediction (Bakır et al., 2007), whose goal is to predict a structured output $\mathbf{Y} \in \mathcal{Y}$ associated with a structured input $\mathbf{X} \in \mathcal{X}$. We define old and new structured losses, and demonstrate them on two structured prediction tasks: named entity recognition and time-series alignment.

8.5.1 *Structured loss functions*

Throughout this section, we assume that the potentials $\theta \in \boldsymbol{\Theta}$ have already been computed using a function from $\mathcal{X}$ to $\boldsymbol{\Theta}$ and let $C\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ be a cost function between the ground-truth output $\mathbf{Y}_{\text{true}}$ and the predicted output $\mathbf{Y}$.

CONVEX LOSSES. Because C is typically non-convex, the cost augmented structured hinge loss (Tsochantaridis et al., 2005) is often used instead for linear models

$$\ell_C(\mathbf{Y}_{\text{true}}; \theta) \triangleq \max_{\mathbf{Y} \in \mathcal{Y}} C(\mathbf{Y}_{\text{true}}, \mathbf{Y}) + \langle \mathbf{Y}, \theta \rangle - \langle \mathbf{Y}_{\text{true}}, \theta \rangle. \tag{8.8}$$

This is a convex upper-bound on $C(\mathbf{Y}_{\text{true}}, \mathbf{Y}^\star(\theta))$, where $\mathbf{Y}^\star(\theta)$ is defined in (8.4). To make the cost-augmented decoding tractable, it is usually assumed that $C(\mathbf{Y}_{\text{true}}, \mathbf{Y})$ is linear in $\mathbf{Y}$, *i.e.*, it can be written as $\langle \mathbf{C}_{\mathbf{Y}_{\text{true}}}, \mathbf{Y} \rangle$ for some matrix $\mathbf{C}_{\mathbf{Y}_{\text{true}}}$. We can then rewrite (8.8) using our notation as

$$\ell_C(\mathbf{Y}_{\text{true}}; \theta) = \mathrm{LP}(\theta + \mathbf{C}_{\mathbf{Y}_{\text{true}}}) - \langle \mathbf{Y}_{\text{true}}, \theta \rangle.$$

However, this loss function is non-differentiable. We therefore propose to relax LP by substituting it with DP_Ω:

$$\ell_{C,\Omega}(\mathbf{Y}_{\text{true}}; \theta) \triangleq \mathrm{DP}_\Omega(\theta + \mathbf{C}_{\mathbf{Y}_{\text{true}}}) - \langle \mathbf{Y}_{\text{true}}, \theta \rangle.$$

Losses in this class are convex, smooth, tractable for any Ω, and by Proposition 8.2, property 2, a sensible approximation of ℓ_C. In addition, they only require to backpropagate through $\mathrm{DP}_\Omega(\theta)$ at training time. It is easy to check that we recover the structured perceptron loss

with $\ell_{0,0}$ (Collins, 2002), the structured hinge loss with $\ell_{C,0}$ (Tsochantaridis et al., 2005) and the CRF loss with $\ell_{0,-H}$ (Lafferty et al., 2001). The last one has been used on top of LSTMs in several recent works (Lample et al., 2016; Ma and Hovy, 2016). Minimizing $\ell_{0,-H}(\theta)$ is equivalent to maximizing the likelihood $p_{\theta,-H}(Y_{\text{true}})$. However, minimizing $\ell_{0,\|\cdot\|^2}$ is *not* equivalent to maximizing $p_{\theta,\|\cdot\|^2}(Y_{\text{true}})$. In fact, the former is convex while the latter is not.

NON-CONVEX LOSSES. A direct approach that uses the output distribution $p_{\theta,\Omega}$ minimizes the risk $\sum_{y\in\mathcal{Y}} p_{\theta,-H}(Y)C(Y_{\text{true}}, Y)$. As discussed by Stoyanov and Eisner (2012), this can be achieved by backpropagating through the minimum risk decoder. However, the risk is usually non-differentiable, piecewise constant (D. A. Smith and Eisner, 2006) and several smoothing heuristics are necessary to make the method work (Stoyanov and Eisner, 2012).

Another principled approach is to consider a differentiable approximation $\Delta\colon \mathcal{Y} \times \operatorname{conv}(\mathcal{Y}) \to \mathbb{R}_+$ of the cost C. We can then relax $C(Y_{\text{true}}, Y^\star(\theta))$ by $\Delta(Y_{\text{true}}, \nabla \mathrm{DP}_\Omega(\theta))$. Unlike minimum risk training, this approach is differentiable everywhere when $\Omega = -H$. Both approaches require to backpropagate through $\nabla \mathrm{DP}_\Omega(\theta)$, which is twice as costly as backpropagating through $\mathrm{DP}_\Omega(\theta)$ (see Section 8.3.4).

8.5.2 *Named entity recognition*

Let $X = (x_1, \cdots, x_T)$ be an input sentence, where each word x_t is represented by a vector in $\mathbb{R}^D$, computed using a neural recurrent architecture trained end-to-end. We wish to tag each word with named entities, *i.e.*, identify blocks of words that correspond to names, locations, dates, etc. We use the specialized operator Vit_Ω described in Section 8.4.1. We construct the potential tensor $\theta(X) \in \mathbb{R}^{T\times S\times S}$ as

$$\forall\, t > 1, \quad \theta(X)_{t,i,j} \triangleq w_i^\top x_t + b_i + t_{i,j},$$

and $\theta(X)_{1,i,j} \triangleq w_i^\top x_t + b_i$, where $(w_i, b_i) \in \mathbb{R}^D \times \mathbb{R}$ is the linear classifier associated with tag i and $T \in \mathbb{R}^{S\times S}$ is a transition matrix. We learn W, b and T along with the network producing X, and compare two losses:

$$\begin{aligned} &\text{Surrogate convex loss:} && \ell_{0,\Omega}(Y_{\text{true}}; \theta), \\ &\text{Relaxed loss:} && \Delta(Y_{\text{true}}, \nabla \mathrm{DP}_\Omega(\theta)), \end{aligned}$$

where $\Delta(Y_{\text{true}}, Y)$ is the squared ℓ_2 distance when $\Omega = \|\cdot\|_2^2$ and the Kullback-Leibler divergence when $\Omega = -H$, applied row-wise to the marginalization of Y_{true} and Y.

EXPERIMENTS. We measure the performance of the different losses and regularizations on the four languages of the CoNLL 2003 dataset. Following Lample et al. (2016), who use the $\ell_{0,-H}$ loss, we use a character LSTM and FastText (Joulin et al., 2016) pretrained embeddings computed using *o*n Wikipedia. Those are fed to a word bidirectional LSTM to obtain X. Architecture details are provided in Section B.3.1. Results are reported in Table 8.1, along with reference

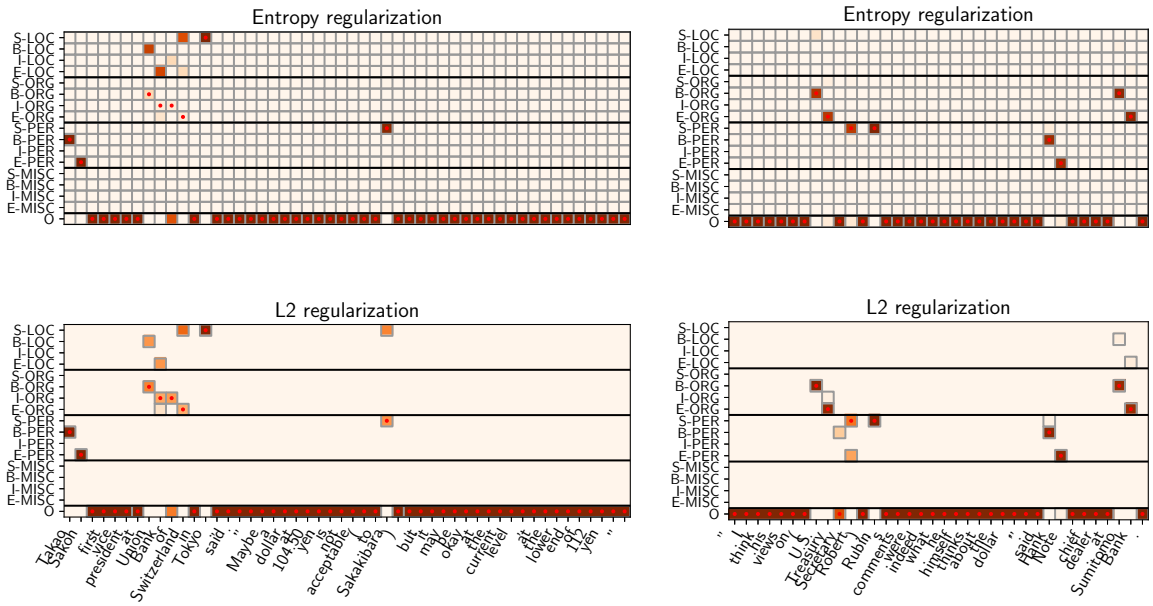


Figure 8.4: Test predictions from the entropy and ℓ_2^2 regularized NER models. Red dots indicate ground truth. When using ℓ_2^2 regularization, model predictions are sparse (grey borders indicates non-zero cells). They are thus easier to introspect for ambiguities, as we can list a finite number of possible outputs.

results with different pretrained embeddings. We first note that the non-regularized structured perceptron loss $\ell_{0,0}$, that involves working with subgradients of $\mathrm{DP}(\theta)$, perform significantly worse than regularized losses. With proper parameter selections, all regularized losses perform within 1% F_1-score of each other, although entropy-regularized losses perform slightly better on 3 out of 4 languages. However, the ℓ_2^2-regularized losses yield sparse predictions, whereas entropy regularization always yields dense probability vectors. Qualitatively, this allows to identify ambiguous predictions more easily. This is illustrated in Figure 8.4, in which we display a few tagged English sequences. The model using ℓ_2^2 regularization correctly identifies an ambiguous entity (*Union Bank of Switzerland*) and proposes two non zero tag sequences: *Union Bank of Switzerland* as an organization, or *Union Bank* as an organization and *Switzerland* as a location.

SET OF TOP PREDICTIONS. Probabilities of every tag sequence can be computed using the matrix $\mathbf{Q}$, as described in Section 8.3.3 — this remains tractable as long as the matrix $\mathbf{Q}$ is *sparse enough*, so that the number of non-zero probabilities sequence remains low. Using the ℓ_2^2 regulariation thus allows to enumerate all non-zero probability entities and provide the user with a set of top k predictions. Potentially, this would allow to trade precision for recall at test time. In contrast, the model using negentropy regularization never assign a zero probability to any tag sequence — it is not tractable to sort these probabilities and provide the user with a small set of interesting sequences.

Table 8.1 – F_1 score comparison on CoNLL03 NER datasets.

Ω	Loss	English	Spanish	German	Dutch
Negent.	Surrogate	90.80	**86.68**	77.35	**87.56**
	Relaxed	90.47	86.20	**77.56**	87.37
ℓ_2^2	Surrogate	**90.86**	85.51	76.01	86.58
	Relaxed	89.49	84.07	76.91	85.90
0	Struct. perceptron	86.52	81.48	68.81	80.49
Lample et al., 2016		*90.96*	*85.75*	*78.76*	*81.74*

Table 8.2 – Mean absolute deviation of alignment using an end-to-end trained multinomial classifier and a pre-trained one.

Linear model	Train	Test
End-to-end trained	$\mathbf{0.17 \pm 0.01}$	$\mathbf{1.07 \pm 0.61}$
Pretrained	1.80 ± 0.14	3.69 ± 2.85
Random θ	14.64 ± 2.63	14.64 ± 0.29

8.5.3 *Supervised audio-to-score transcription*

We use our framework to perform supervised audio-to-score alignment on the Bach 10 dataset (Duan and Pardo, 2011). The dataset consists of 10 music pieces with audio tracks, MIDI transcriptions, and annotated alignments between them. We transform the audio tracks into a sequence of audio frames using a feature extractor (see Section B.3.2) to obtain a sequence $\boldsymbol{A} \in \mathbb{R}^{N_A \times D}$, while the associated score sequence is represented by $\mathbf{B} \in \mathbb{R}^{N_B \times K}$ (each row $\mathbf{b}_j$ is a one-hot vector corresponding to one key b_j). Each pair $(\boldsymbol{A}, \mathbf{B})$ is associated with an alignment $\boldsymbol{Y}_{\text{true}} \in \mathbb{R}^{N_A \times N_B}$. As described in Section 8.4.2, we define a discrepancy matrix $\theta \in \mathbb{R}^{N_A \times N_B}$ between the elements of the two sequences. We set the cost between an audio frame and a key to be the log-likelihood of this key given a multinomial linear classifier:

$$\begin{aligned} \forall i \in [N_A], \mathbf{l}_i \triangleq -\log(\text{softmax}(\boldsymbol{W}^\top \boldsymbol{a}_i + \boldsymbol{c})) \in \mathbb{R}^K \\ \text{and } \forall j \in [N_B],\ \theta_{i,j} \triangleq l_{i,b_j}, \end{aligned} \tag{8.9}$$

where $(\boldsymbol{W}, \mathbf{c}) \in \mathbb{R}^{D \times K} \times \mathbb{R}^K$ are learned classifier parameters. We predict a soft alignment by $\boldsymbol{Y} = \nabla \text{DTW}_{-H}(\theta)$. Following (Garreau et al., 2014), we define the relaxed loss

$$\Delta(\boldsymbol{Y}_{\text{true}}, \boldsymbol{Y}) \triangleq \|\mathbf{L}(\boldsymbol{Y} - \boldsymbol{Y}_{\text{true}})^\top\|_F^2,$$

where $\mathbf{L}$ a the lower triangular matrix filled with 1. When $\boldsymbol{Y} \in \mathcal{Y}$ is a true alignment matrix, $\Delta(\boldsymbol{Y}_{\text{true}}, \boldsymbol{Y})$ is the area between the path of

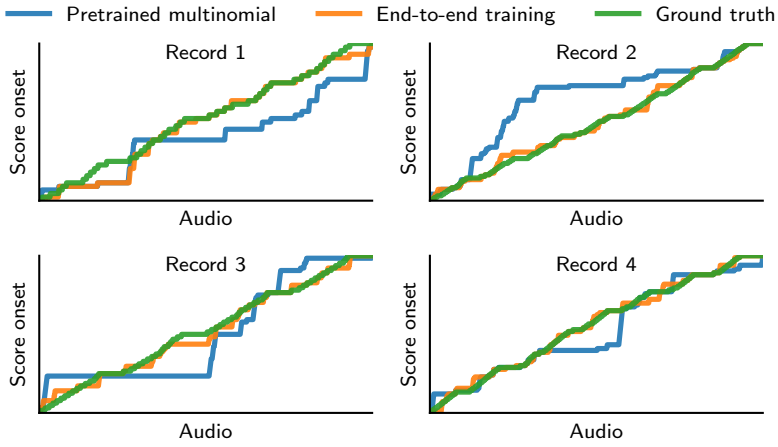


Figure 8.5 – Alignment maps between score onsets and audio frames on test data from the Bach10 dataset. Our end-to-end trained model qualitatively performs better than the baseline model.

$\mathbf{Y}_{\text{true}}$ and $\mathbf{Y}$, which corresponds to the *mean absolute deviation* in the audio literature. When $\mathbf{Y} \in \text{conv}(\mathcal{Y})$, it is a convex relaxation of the area. At test time, once θ is learned, we use the non-regularized DTW algorithm to output a hard alignment $\mathbf{Y}^\star(\theta) \in \mathcal{Y}$.

RESULTS. We perform a leave-one-out cross-validation of our model performance, learning the multinomial classifier on 9 pieces and assessing the quality of the alignment on the remaining piece. We report the mean absolute deviation on both train and test sets. A solid baseline consists in learning the multinomial classifier $(\boldsymbol{W}, \mathbf{c})$ beforehand, *i.e.*, without end-to-end training. We then use this model to compute θ as in (8.9) and obtain $\mathbf{Y}^\star(\theta)$. As shown in Table 8.2, our end-to-end technique outperforms this baseline by a large margin.

In Figure 8.5, we display the alignment maps we obtained using our algorithm and using the baseline multinomial model followed by a hard-DTW alignment computation. These alignment maps correspond to the predicted onsets of keys. Our model (in orange) performs visibly better in predicting onsets.

End-to-end training thus allows to *fine-tune* the distance matrix θ for the task at hand.

8.6 STRUCTURED AND SPARSE ATTENTION

We show in this section how to apply our framework to neural sequence-to-sequence models augmented with an attention mechanism (Bahdanau et al., 2015). An encoder first produces a list of vectors $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_T)$ representing the input sequence. A decoder is then used to greedily produce the corresponding output sequence. To simplify the notation, we focus on one time step of the decoding procedure. Given the decoder's current hidden state $\boldsymbol{z}$ and $\mathbf{X}$ as inputs, the role of the attention mechanism is to produce a distribution $\boldsymbol{w} \in \triangle^T$ over $\mathbf{X}$, for the current time step. This distribution is then

typically used to produce a context vector $\mathbf{c} \triangleq \mathbf{X}^\top \boldsymbol{w}$, that is in turn invoved in the computation of the output sequence's next element.

STRUCTURED ATTENTION LAYERS. Y. Kim et al. (2017) proposed a segmentation attention layer, which is capable of taking into account the transitions between elements of $\mathbf{X}$. They use a linear-chain CRF to model the probability $p_{\theta,-H}(\mathbf{y}|\mathbf{X})$ of a sequence $\mathbf{y} = (y_1, \ldots, y_T)$, where each y_t is either 1 ("pay attention") or 0. They then propose to use normalized marginal probabilities as attention weights: $w_t \propto p_{\theta,-H}(y_t = 1|\mathbf{X})$. They show how to backpropagate gradients through the forward-backward algorithm, which they use to compute the marginal probabilities.

GENERALIZING STRUCTURED ATTENTION. With Section 8.4.1 notation, any $\mathbf{y}$ can be represented as a tensor $\mathbf{Y} \in \{0,1\}^{T\times 2\times 2}$ and the potentials as a tensor $\theta \in \mathbb{R}^{T\times 2\times 2}$. Similarly to Y. Kim et al. (2017), we define

$$\theta_{t,1,j} \triangleq \mathbf{x}_t \mathbf{M} \boldsymbol{z} + t_{1,j} \quad \text{and} \quad \theta_{t,0,j} \triangleq t_{0,j},$$

where $\mathbf{x}\mathbf{M}\boldsymbol{z}$ is a learned bilinear form and $\mathbf{T} \in \mathbb{R}^{2\times 2}$ is a learned transition matrix. Following Section 8.4.1, the gradient $\nabla \mathrm{Vit}_\Omega(\theta)$ is equal to the expected matrix $\mathbf{E} \in \mathbb{R}^{T\times 2\times 2}$ and the marginals are obtained by marginalizing that matrix. Hence, we can set $w_t \propto p_{\theta,\Omega}(y_t = 1|\mathbf{X}) = e_{t,1,0} + e_{t,1,1}$. Backpropagating through $\nabla \mathrm{Vit}_\Omega(\theta)$ can be carried out using our approach outlined in Section 8.3.4. This approach is not only more general, but also simpler and more robust to underflow problems than backpropagating through the forward-backward algorithm as done by Y. Kim et al. (2017).

EXPERIMENTS. We demonstrate structured attention layers with an LSTM encoder and decoder to perform French to English translation using data from a 1 million sentence subset of the WMT14 fr-en challenge. We illustrate an example of attention map obtained with negentropy and ℓ_2^2 regularizations in Figure 8.6. Non-zero elements are underlined with borders: ℓ_2^2-regularized attention maps are sparse and more interpretable — this provides a structured alternative to sparsemax attention (Martins and Astudillo, 2016). Results were all within 0.8 point of BLEU score on the newstest2014 dataset. For French to English, standard softmax attention obtained **27.96**, while entropy and ℓ_2^2 regularized structured attention obtained **27.96** and **27.19** — introducing structure and sparsity therefore provides enhanced interpretability with comparable performance. We provide model details, full results and further visualizations in Section B.3.3.

8.7 CONCLUSION

In this chapter, we proposed a theoretical framework for turning a broad class of dynamic programs into convex, differentiable and tractable operators, using the novel point of view of smoothed max

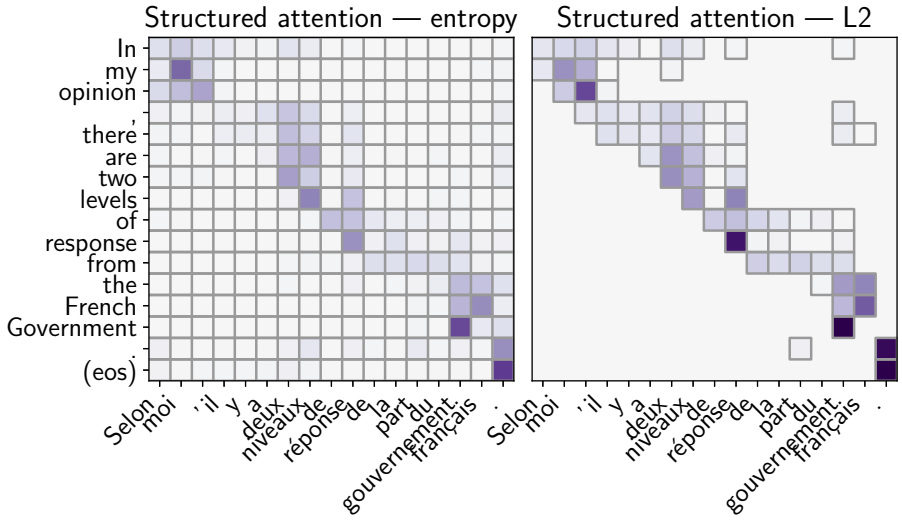


Figure 8.6 – Attention maps obtained with structured attention. Although both regularizations led to the same translation (y-axis) in this example, attention is sparse and more interpretable with ℓ_2^2.

operators. This approach sheds a new light on how to transform dynamic programs that predict hard assignments (*e.g.,* the maximum a-posteriori estimator in a probabilistic graphical model or an alignment matrix between two time-series) into continuous and probabilistic ones. We provided a new argument in favor of negentropy regularization by showing that it is the only one to preserve *associativity* of the smoothed max operator. We showed that different regularizations induce different distributions over outputs and that ℓ_2^2 regularization has other benefits, in terms of sparsity of the expected outputs. Generally speaking, performing inference in a graphical model and backpropagating through it reduces to computing the first and second-order derivatives of a relaxed maximum-likelihood estimation — leveraging this observation yields elegant and efficient algorithms that are readily usable in deep learning frameworks, with various promising applications.

Part v

CONCLUSION

9 CONCLUSION

In this thesis, we developed two new approaches for functional MRI analysis, that opens new perspectives for taking advantage of the amount of data that is now available.

First, in Chapter 3–5, we showed that sparse matrix factorization techniques were amenable to the terabytes of data produced by resting state fMRI. We learned that introducing random perturbation in learning algorithms can prove beneficial if these perturbations allows to perform faster updates while keeping most of the signal. We can know learn dictionaries of 1000 components from the HCP1200 dataset in a few days, which opens new perspectives for subsequent analysis: we are the first to provide so many continuous functional networks learned on nearly 5000 resting-state fMRI records.

We are now performing an extensive validation of these networks for various neuro-imaging tasks. In a work to appear, we show that using functional networks learned on thousands of subjects for reducing the dimensionality of input data is beneficial for diverse data analysis tasks performed by neuroscientists. We argue in favor gathering fMRI studies in the form of loadings over the functional networks we provide, at different scale (128, 512, 1024 components), in common public repositories.

Second, in Chapter 7, we showed the interest of dusing eeper models in predictive modeling for neuro-imaging. We established that multi-layer models could identify meaningful cogntitive directions in which decoding is made easier, and successfully aggregate the information from many studies to improve decoding accuracy. We performed an extensive study of the components that made our model accurate and interpretable, and established the interest of newly introduced regularization and training techniques (Dropout, batch normalization, Adam) in the field of fMRI. The method based on ensembling and NMF that we introduced extract interesting directions in the output space of an intermediary layer is new and may be of interest in other applications — the problem of producing interpretable deep models is indeed quite fundamental in the field. This line of research brings an interesting perspective in machine learning: first, learning over-parametrized models with stochastic optimization and regularization allows to take a step forward in performance. Yet, to interpret the predictions of these models, we have to step down and transform the learned models in ways that reduces the effect of random training, so as to recover meaningful parameters.

The matrix factorization methods we developed in Chapter 4–5 exhibits the power of stochastic subsampling, which appears an efficient method to accelerate training of models with high-dimensional inputs. The SMM framework that we extended into SAMM includes stochastic gradient descent (SGD as an instance. The performance

gain obtained by sketching surrogate computations and freezing a large fraction of the model parameter (the dictionary $\mathbf{D}$) at each iteration suggests that these approaches should be useful when running SGD on more general non-convex objectives than the matrix factorization one. Meanwhile, our methods proved useful in several MF applications (hyperspectral imaging, collaborative filtering), and are likely to provide large speed-ups for working with high spatial and temporal resolution modalites, such as electron microscopy imaging, tomography, etc.

Finally, the smoothing approach proposed in Chapter (8) is simple yet general, and is likely to be amenable to other algorithms that have an approximate dynamic programming structure (*e. g.*, Dijsktra algorithm), and to reinforcement learning problems. The fact that ℓ_2^2 smoothing provides a way to compute sparse marginals in graphical models is also an interesting properties. It permits to output small sets of top-scored predictions, which should improve performance (typically, in text analysis and language translation).

9.1 SOFTWARE

A number of software contributions have been performed within the context of this thesis. All produced code was written in Python, using a combination of *scikit-learn*[11], *nilearn*[12], *Cython*[13], *PyTorch*[14] for experiment design, efficient routine writing and model design, respectively. Software references are provided in the main section of this thesis.

[11] *scikit-learn.org*
[12] *nilearn.github.io*
[13] *cython.org*
[14] *pytorch.org*

The work presented in Chapter 4–5 is available as a Python package called *modl*, that heavily relies on Cython for core algorithmic implementation:

github.com/arthurmensch/modl

The work presented in Chapter 7 is available as a Python package called *cogspaces*. The multi-layer model is defined and trained using *PyTorch*, while the pipelines for handling data and evaluating performance relies on *scikit-learn*.

github.com/arthurmensch/cogspaces

The work presented in Chapter 8 is available as a Python package called *didyprog*. The new dynamic programming layers are implemented using *PyTorch* low-level CPU/GPU API. We provide these layers and wrap them in a higher level library for natural language processing, and in existing models for neural machine translation.

github.com/arthurmensch/didyprog

During these three years, I had the joy to contribute to *scikit-learn* (improvement in the *decomposition* module, SAGA algorithm, *cython* compilation system) and *nilearn* (dictionary learning module).

BIBLIOGRAPHY

Abraham, A., Milham, M. P., Di Martino, A., Craddock, R. C., Samaras, D., Thirion, B., & Varoquaux, G. (2017). Deriving reproducible biomarkers from multi-site resting-state data: an Autism-based example. *NeuroImage*, *147*, 736–745.

Abraham, A., Pedregosa, F., Eickenberg, M., Gervais, P., Mueller, A., Kossaifi, J., ... Varoquaux, G. (2014). Machine learning for neuroimaging with Scikit-Learn. *Frontiers in Neuroinformatics*, *8*, 14.

Aji, S. M., & McEliece, R. J. (2000). The generalized distributive law. *IEEE Transactions on Information Theory*, *46*(2), 325–343.

Alvarez, R. P., Jasdzewski, G., & Poldrack, R. A. (2002). Building memories in two languages: an fMRI study of episodic encoding in bilinguals. In *Society for Neuroscience Abstracts*.

Amalric, M., & Dehaene, S. (2016). Origins of the brain networks for advanced mathematics in expert mathematicians. *Proceedings of the National Academy of Sciences*, *113*(18), 4909–4917.

Amos, B., & Kolter, J. Z. (2017). Optnet: differentiable optimization as a layer in neural networks. In *Proceedings of the International Conference on Machine Learning* (pp. 136–145).

Ando, R. K., & Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, *6*, 1817–1853.

Aron, A. R., Behrens, T. E., Smith, S., Frank, M. J., & Poldrack, R. (2007). Triangulating a cognitive control network using diffusion-weighted magnetic resonance imaging (MRI) and functional MRI. *The Journal of Neuroscience*, *27*, 3743–3752.

Aron, A. R., Gluck, M., & Poldrack, R. A. (2006). Long-term test–retest reliability of functional MRI in a classification learning task. *NeuroImage*, *29*, 1000–1006.

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural Machine Translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representation*.

Bakır, G., Hofmann, T., Schölkopf, B., Smola, A. J., Taskar, B., & Vishwanathan, S. V. N. (2007). *Predicting structured data*. The MIT Press.

Banderier, C., & Schwer, S. (2005). Why Delannoy numbers? *Journal of Statistical Planning and Inference*, *135*(1), 40–54.

Barch, D. M., Burgess, G. C., Harms, M. P., Petersen, S. E., Schlaggar, B. L., Corbetta, M., ... Consortium, W. U.-M. H. (2013). Function in the human connectome: task-fMRI and individual differences in behavior. *NeuroImage*, *80*, 169–189.

Barrett, L. F. (2009). The future of psychology: connecting mind to brain. *Perspectives on Psychological Science*, *4*(4), 326–339.

Baum, L. E., & Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, *37*(6), 1554–1563.

Beck, A., & Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, *2*(1), 183–202.

Beck, A., & Teboulle, M. (2012). Smoothing and first order methods: a unified framework. *SIAM Journal on Optimization*, *22*(2), 557–580.

Beck, A., & Tetruashvili, L. (2013). On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization*, *23*(4), 2037–2060.

Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D. S., & Smith, K. (2011). Cython: the best of both worlds. *Computing in Science & Engineering*, *13*(2), 31–39.

Bell, A. J., & Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, *7*(6), 1129–1159.

Bell, R. M., & Koren, Y. (2007). Lessons from the Netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, *9*(2), 75–79.

Bellman, R. (1952). On the theory of dynamic programming. *Proceedings of the National Academy of Sciences*, *38*(8), 716–719.

Bertsekas, D. P. (1971). *Control of uncertain systems with a set-membership description of the uncertainty.* (Doctoral dissertation, Massachusetts Institute of Technology).

Bingham, E., & Mannila, H. (2001). Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the SIGKDD Conference* (pp. 245–250).

Bishop, C. M. (1995). *Pattern Recognition and Machine Learning*. Springer.

Biswal, B., Zerrin Yetkin, F., Haughton, V. M., & Hyde, J. S. (1995). Functional connectivity in the motor cortex of resting human brain using echo-planar MRI. *Magnetic Resonance in Medicine*, *34*(4), 537–541.

Blondel, M., Seguy, V., & Rolet, A. (2018). Smooth and sparse optimal transport. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*.

Borwein, J. M., & Lewis, A. S. (2010). *Convex analysis and nonlinear optimization: theory and examples*. Springer Science & Business Media.

Bottou, L. (1999). On-line learning and stochastic approximations. *On-line Learning in Neural Networks*, 9–42.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT* (pp. 177–186).

Bottou, L., Bengio, Y., & LeCun, Y. (1997). Global training of document processing systems using graph transformer networks. In *Proceedings of the Conference on Computer Vision and Pattern Recognition* (pp. 489–494).

Bottou, L., Curtis, F., & Nocedal, J. (2018). Optimization methods for large-scale machine learning. *SIAM Review*, *60*(2), 223–311.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Burer, S., & Monteiro, R. D. C. (2004). Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming*, *103*(3), 427–444.

Button, K. S., Ioannidis, J. P. A., Mokrysz, C., Nosek, B. A., Flint, J., Robinson, E. S. J., & Munafò, M. R. (2013). Power failure: why small sample size undermines the reliability of neuroscience. *Nature Reviews Neuroscience*, *14*(5), 365–376.

Calhoun, V., Adali, T., Pearlson, G., & Pekar, J. (2001). A method for making group inferences from functional MRI data using independent component analysis. *Human Brain Mapping*.

Candès, E. J., & Recht, B. (2009). Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, *9*(6), 717–772.

Candès, E. J., & Tao, T. (2006). Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Transactions on Information Theory*, *52*(12), 5406–5425.

Cappé, O., & Moulines, E. (2009). Online EM algorithm for latent data models. *Journal of the Royal Statistical Society: Series B*, *71*(3), 593–613.

Cauchy, L. A. (1847). Méthode générale pour la résolution des systèmes d'équations simultanées. In *Compte Rendu à l'Académie des Sciences de Paris*.

Cauvet, E. (2012). *Traitement des structures syntaxiques dans le langage et dans la musique* (Doctoral dissertation, Paris 6).

Chen, Y., Nasrabadi, N. M., & Tran, T. D. (2011). Hyperspectral image classification using dictionary-based sparse representation. *IEEE Transactions on Geoscience and Remote Sensing*, *49*(10), 3973–3985.

Cohen, J. R. (2009). *The development and generality of self-control* (Doctoral dissertation, University of the City of Los Angeles).

Collier, A. K., Wolf, D. H., Valdez, J. N., Turetsky, B. I., Elliott, M. A., Gur, R. E., & Gur, R. C. (2014). Comparison of auditory and visual oddball fMRI in schizophrenia. *Schizophrenia research*, *158*, 183–188.

Collins, M. (2002). Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of ACL* (pp. 1–8).

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceeding of the International Conference on Machine Learning* (pp. 160–167).

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, *12*, 2493–2537.

Cox, D. D., & Savoy, R. L. (2003). Functional magnetic resonance imaging (fMRI) "brain reading": detecting and classifying distributed patterns of fMRI activity in human visual cortex. *NeuroImage*, *19*(2 Pt 1), 261–270.

Cuturi, M., & Blondel, M. (2017). Soft-dtw: a differentiable loss function for time-series. In *Proceedings of the International Conference on Machine Learning* (pp. 894–903).

Danskin, J. M. (1966). The theory of max-min, with applications. *SIAM Journal on Applied Mathematics*, *14*(4), 641–664.

Desikan, R. S., Ségonne, F., Fischl, B., Quinn, B. T., Dickerson, B. C., Blacker, D., ... Killiany, R. J. (2006). An automated labeling system for subdividing the human cerebral cortex on MRI scans into gyral based regions of interest. *NeuroImage*, *31*(3), 968–980.

Devauchelle, A. D., Oppenheim, C., Rizzi, L., Dehaene, S., & Pallier, C. (2009). Sentence syntax and content in the human temporal lobe: an fMRI adaptation study in auditory and visual modalities. *Journal of Cognitive Neuroscience*, *21*(5), 1000–1012.

Devlin, J. T., & Poldrack, R. A. (2007). In praise of tedious anatomy. *NeuroImage*, *37*(4), 1033–1041, discussion 1050–1058.

Djolonga, J., & Krause, A. (2017). Differentiable learning of submodular functions. In *Advances in Neural Information Processing Systems* (pp. 1014–1024).

Dohmatob, E., Mensch, A., Varoquaux, G., & Thirion, B. (2016). Learning brain regions via large-scale online structured sparse dictionary learning. In *Advances in Neural Information Processing Systems*.

Doob, J. L. (1990). *Stochastic processes*. John Wiley & Sons.

Duan, Z., & Pardo, B. (2011). Soundprism: an online system for score-informed source separation of music audio. *IEEE Journal of Selected Topics in Signal Processing*, *5*(6), 1205–1215.

Duchi, J., Shalev-Shwartz, S., Singer, Y., & Chandra, T. (2008). Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *Proceedings of the International Conference on Machine Learning* (pp. 272–279).

Duchi, J., & Singer, Y. (2009). Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, *10*, 2899–2934.

Duncan, K., Pattamadilok, C., Knierim, I., & Devlin, J. (2009). Consistency and variability in functional localisers. *NeuroImage*, *46*, 1018–1026.

Eckart, C., & Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, *1*(3), 211–218.

Eisner, J. (2016). Inside-outside and forward-backward algorithms are just backprop (tutorial paper). In *Proceedings of the Workshop on Structured Prediction for NLP* (pp. 1–17).

Etzel, J. A., Gazzola, V., & Keysers, C. (2009). An introduction to anatomical ROI-based fMRI classification analysis. *Brain Research*, *1282*, 114–125.

Evans, A., Collins, D., Mills, S., Brown, E., Kelly, R., & Peters, T. (1993). 3d statistical neuroanatomical models from 305 MRI volumes. (pp. 1813–1817). IEEE.

Fazel, M., Hindi, H., & Boyd, S. (2001). A rank minimization heuristic with application to minimum order system approximation. (4734–4739 vol.6). IEEE.

Yu-Feng, Z., Yong, H., Chao-Zhe, Z., Qing-Jiu, C., Man-Qiu, S., Meng, L., ... Yu-Feng, W. (2007). Altered baseline brain activity in children with ADHD revealed by resting-state functional MRI. *Brain and Development*, *29*(2), 83–91.

Fisher, R. A. (1938). The statistical utilization of multiple measurements. *Annals of Human Genetics*, *8*(4), 376–386.

Foerde, K., Knowlton, B., & Poldrack, R. A. (2006). Modulation of competing memory systems by distraction. *Proceedings of the National Academy of Science*, *103*, 11778–11783.

Fox, M. D., & Raichle, M. E. (2007). Spontaneous fluctuations in brain activity observed with functional magnetic resonance imaging. *Nature Reviews Neuroscience*, *8*(9), 700.

Friedman, J., Hastie, T., Höfling, H., & Tibshirani, R. (2007). Pathwise coordinate optimization. *The Annals of Applied Statistics*, *1*(2), 302–332.

Friston, K. J., Holmes, A. P., Worsley, K. J., Poline, J.-P., Frith, C. D., & Frackowiak, R. S. (1994). Statistical parametric maps in functional imaging: a general linear approach. *Human brain mapping*, *2*(4), 189–210.

Garreau, D., Lajugie, R., Arlot, S., & Bach, F. (2014). Metric learning for temporal sequence alignment. In *Advances in Neural Information Processing Systems* (pp. 1817–1825).

Gauthier, B., Eger, E., Hesselmann, G., Giraud, A.-L., & Kleinschmidt, A. (2012). Temporal tuning properties along the human ventral visual stream. *The Journal of Neuroscience*, *32*, 14433–14441.

Gorgolewski, K. J., Storkey, A., Bastin, M. E., Whittle, I. R., Wardlaw, J. M., & Pernet, C. R. (2013). A test-retest fMRI dataset for motor, language and spatial attention functions. *GigaScience*, *2*(1), 6.

Goyal, K., Neubig, G., Dyer, C., & Berg-Kirkpatrick, T. (2017). A continuous relaxation of beam search for end-to-end training of neural sequence models. *arXiv:1708.00111 [cs]*.

Greicius, M. D. (2008). Resting-state functional connectivity in neuropsychiatric disorders. *Current Opinion in Neurology*, *21*(4), 424–430.

Greicius, M. D., Krasnow, B., Reiss, A. L., & Menon, V. (2003). Functional connectivity in the resting brain: a network analysis of the default mode hypothesis. *Proceedings of the National Academy of Sciences*, *100*(1), 253–258.

Grosenick, L., Klingenberg, B., Katovich, K., Knutson, B., & Taylor, J. E. (2013). Interpretable whole-brain prediction analysis with GraphNet. *NeuroImage*, *72*, 304–321.

Gselmann, E. (2011). Entropy functions and functional equations. *Mathematical Communications*, (16), 347–357.

Halko, N., Martinsson, P.-G., & Tropp, J. A. (2011). Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, *53*(2), 217–288.

Hara, N., Cauvet, E., Devauchelle, A. D., Dehaene, S., Pallier, C. et al. (2009). Neural correlates of constituent structure in language and music. *NeuroImage*, *47*, S143.

Hastie, T., Mazumder, R., Lee, J. D., & Zadeh, R. (2015). Matrix completion and low-rank SVD via fast alternating least squares. *Journal of Machine Learning Research*, *16*, 3367–3402.

Haxby, J. V., Gobbini, I. M., Furey, M. L., Ishai, A., Schouten, J. L., & Pietrini, P. (2001a). Distributed and overlapping representations of faces and objects in ventral temporal cortex. *Science*, *293*(5539), 2425–2430.

Haxby, J. V., Gobbini, M. I., Furey, M. L., Ishai, A., Schouten, J. L., & Pietrini, P. (2001b). Distributed and overlapping representations of faces and objects in ventral temporal cortex. *Science*, *293*(5539), 2425–2430.

Haynes, J.-D. (2015). A primer on pattern-based approaches to fMRI: principles, pitfalls, and perspectives. *Neuron*, *87*(2), 257–270.

Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., & Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their applications*, *13*(4), 18–28.

Henson, R. N., Wakeman, D. G., Litvak, V., & Friston, K. J. (2011). A parametric empirical bayesian framework for the EEG/MEG inverse problem: generative models for multi-subject and multi-modal integration. *Frontiers in Human Neuroscience*, *5*.

Himberg, J., Hyvärinen, A., & Esposito, F. (2004). Validating the independent components of neuroimaging time series via clustering and visualization. *NeuroImage*, *22*, 1214.

Horibe, Y. (1988). Entropy of terminal distributions and the Fibonnacci trees. *The Fibonacci Quarterly*, (26), 135–140.

Hoyer, P. O. (2004). Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, *5*, 1457–1469.

Hyvärinen, A., & Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural Networks*, *13*(4-5), 411–430.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning* (pp. 448–456).

Jimura, K., Cazalis, F., Stover, E. R. S., & Poldrack, R. A. (2014). The neural basis of task switching changes with skill acquisition. *Frontiers in Human Neuroscience*, *8*.

Johnson, W. B., & Lindenstrauss, J. (1984). Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, *26*(1), 189–206.

Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., & Mikolov, T. (2016). Fasttext.zip: compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Kandel, E. R., Schwartz, J. H., Jessell, T. M., Biochemistry, D. o., Jessell, M. B. T., Siegelbaum, S., & Hudspeth, A. J. (1981). *Principles of neural science*. Elsevier.

Kanwisher, N., McDermott, J., & Chun, M. M. (1997). The fusiform face area: a module in human extrastriate cortex specialized for face perception. *Journal of Neuroscience*, *17*(11), 4302–4311.

Kelly, A., & Milham, M. (N.d.). Simon task. https://openfmri.org/dataset/ds000101.

Kelly, A., Uddin, L. Q., Biswal, B. B., Castellanos, F., & Milham, M. (2008). Competition between functional brain networks mediates behavioral variability. *NeuroImage*, *39*, 527.

Kim, H., & Park, H. (2007). Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, *23*(12), 1495–1502.

Kim, Y., Denton, C., Hoang, L., & Rush, A. M. (2017). Structured attention networks. In *Proceedings of the International Conference on Learning Representation*.

Kingma, D. P., & Ba, J. (2015). Adam: a method for stochastic optimization. In *International Conference for Learning Representations*.

Kingma, D. P., Salimans, T., & Welling, M. (2015). Variational Dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems* (pp. 2575–2583).

Knops, A., Thirion, B., Hubbard, E. M., Michel, V., & Dehaene, S. (2009). Recruitment of an area involved in eye movements during mental arithmetic. *Science*, *324*, 1583–1585.

Koyejo, O., & Poldrack, R. A. (2013). Decoding cognitive processes from functional MRI. In *NIPS Workshop on Machine Learning for Interpretable Neuroimaging* (pp. 5–10).

Kriegeskorte, N., Goebel, R., & Bandettini, P. (2006). Information-based functional brain mapping. *Proceedings of the National Academy of Sciences*, *103*(10), 3863–3868.

Kuhn, H. W. (1956). Variants of the Hungarian method for assignment problems. *Naval Research Logistics*, *3*(4), 253–258.

Lafferty, J., McCallum, A., & Pereira, F. C. (2001). Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning* (pp. 282–289).

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., & Dyer, C. (2016). Neural architectures for named entity recognition. In *Proceedings of NAACL* (pp. 260–270).

Lauterbur, P. (1973). Image formation by induced local interactions: examples employing nuclear magnetic resonance. *Nature*, *242*, 190.

Leblond, R., Pedregosa, F., & Lacoste-Julien, S. (2017). ASAGA: asynchronous parallel SAGA. In *Proceedings of the International Conference on Artificial Intelligence and Statistics* (pp. 46–54).

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Levy, O., & Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems* (pp. 2177–2185).

Lindquist, M. A., Loh, J. M., Atlas, L. Y., & Wager, T. D. (2009). Modeling the hemodynamic response function in fMRI: efficiency, bias and mis-modeling. *NeuroImage*, *45*(1), S187–S198.

Linnainmaa, S. (1970). *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors* (Doctoral dissertation, Univ. Helsinki).

Loula, J., Varoquaux, G., & Thirion, B. (2017). Decoding fMRI activity in the time domain improves classification performance. *NeuroImage*.

Lu, Y., Dhillon, P., Foster, D. P., & Ungar, L. (2013). Faster ridge regression via the subsampled randomized hadamard transform. In *Advances in Neural Information Processing Systems* (pp. 369–377).

Luong, T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP* (pp. 1412–1421).

Ma, X., & Hovy, E. (2016). End-to-end sequence labeling via bidirectional LSTM-CNNs-CRF. In *Proceedings of ACL* (pp. 1064–1074).

Maggioni, M., Katkovnik, V., Egiazarian, K., & Foi, A. (2013). Nonlocal transform-domain filter for volumetric data denoising and reconstruction. *IEEE Transactions on Image Processings*, *22*(1), 119–133.

Mairal, J. (2013a). Optimization with first-order surrogate functions. In *Proceedings of the International Conference on Machine Learning* (pp. 783–791).

Mairal, J. (2013b). Stochastic majorization-minimization algorithms for large-scale optimization. In *Advances in Neural Information Processing Systems* (pp. 2283–2291).

Mairal, J., Bach, F., & Ponce, J. (2014). Sparse modeling for image and vision processing. *Foundations and Trends in Computer Graphics and Vision*, *8*(2-3), 85–283.

Mairal, J., Bach, F., Ponce, J., & Sapiro, G. (2010). Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, *11*, 19–60.

Mairal, J., Ponce, J., Sapiro, G., Zisserman, A., & Bach, F. R. (2009). Supervised Dictionary Learning. In D. Koller, D. Schuurmans, Y. Bengio, & L. Bottou (Eds.), *Advances in Neural Information Processing Systems* (pp. 1033–1040). Curran Associates, Inc.

Makni, S., Idier, J., Vincent, T., Thirion, B., Dehaene-Lambertz, G., & Ciuciu, P. (2008). A fully Bayesian approach to the parcel-based detection-estimation of brain activity in fMRI. *NeuroImage*, *41*(3), 941–969.

Mardani, M., Mateos, G., & Giannakis, G. B. (2015). Subspace Learning and Imputation for Streaming Big Data Matrices and Tensors. *IEEE Transactions on Signal Processing*, *63*(10), 2663–2677.

Martins, A. F., & Astudillo, R. F. (2016). From softmax to sparsemax: a sparse model of attention and multi-label classification. In *Proceedings of the International Conference on Machine Learning* (pp. 1614–1623).

McKeown, M. J., Makeig, S., Brown, G. G., Jung, T. P., Kindermann, S. S., Bell, A. J., & Sejnowski, T. J. (1998). Analysis of fMRI data by blind separation into independent spatial components. *Human Brain Mapping*, *6*(3), 160–188.

Mensch, A., & Blondel, M. (2018). Differentiable dynamic programming for structured prediction and attention. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Mensch, A., Mairal, J., Bzdok, D., Thirion, B., & Varoquaux, G. (2017). Learning neural representations of human cognition across many fMRI studies. In *Advances in Neural Information Processing Systems*.

Mensch, A., Mairal, J., Thirion, B., & Varoquaux, G. (2016a). Dictionary learning for massive matrix factorization. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Mensch, A., Mairal, J., Thirion, B., & Varoquaux, G. (2018a). Extracting universal representations of cognition across brain-imaging studies. *arXiv:1809.06035 [stat.ML]*.

Mensch, A., Mairal, J., Thirion, B., & Varoquaux, G. (2018b). Stochastic Subsampling for factorizing huge matrices. *IEEE Transactions on Signal Processing*, *66*(1), 113–128.

Mensch, A., Varoquaux, G., & Thirion, B. (2016b). Compressed online dictionary learning for fast fMRI decomposition. In *Proceedings of the IEEE International Symposium on Biomedical Imaging (ISBI)*.

Meshi, O., Mahdavi, M., & Schwing, A. G. (2015). Smooth and strong: MAP inference with linear convergence. In *Adavnces in Neural Information Processing Systems*.

Métivier, M. (1982). *Semimartingales: a course on stochastic processes*. Walter de Gruyter.

Michel, V., Gramfort, A., Varoquaux, G., Eger, E., & Thirion, B. (2011). Total Variation regularization for fMRI-based prediction of behavior. *IEEE Transactions on Medical Imaging*, *30*(7), 1328–1340.

Michelot, C. (1986). A finite algorithm for finding the projection of a point onto the canonical simplex of $\mathbb{R}^n$. *Journal of Optimization Theory and Applications*, *50*(1), 195–200.

Milham, M. P. P. D., Fair, D. P.-C., Mennes, M. P. D., & Mostofsky, S. H. M. D. (2012). The ADHD-200 consortium: a model to advance the translational potential of neuroimaging in clinical neuroscience. *Frontiers in Systems Neuroscience*, *6*.

Molchanov, D., Ashukha, A., & Vetrov, D. (2017). Variational Dropout Sparsifies Deep Neural Networks. In *Proceedings of the International Conference on Machine Learning* (pp. 2498–2507).

Moran, J. M., Jolly, E., & Mitchell, J. P. (2012). Social-cognitive deficits in normal aging. *The Journal of Neuroscience*, *32*, 5553–5561.

Moreau, J.-J. (1965). Proximité et dualité dans un espace hilbertien. *Bullet de la Société Mathémathique de France*, *93*(2), 273–299.

Mourão-Miranda, J., Bokde, A. L. W., Born, C., Hampel, H., & Stetter, M. (2005). Classifying brain states and determining the discriminating activation patterns: support vector machine on functional MRI data. *NeuroImage*, *28*(4), 980–995.

Mumford, J. A., Turner, B. O., Ashby, F. G., & Poldrack, R. A. (2012). Deconvolving BOLD activation in event-related designs for multivoxel pattern classification analyses. *NeuroImage*, *59*(3), 2636–2643.

Nesterov, Y. (2005). Smooth minimization of non-smooth functions. *Mathematical Programming*, *103*(1), 127–152.

Newell, A. (1973). You can't play 20 questions with nature and win: Projective comments on the papers of this symposium. *Visual Information Processing*, 1–26.

Neyshabur, B. (2017). *Implicit Regularization in Deep Learning* (Doctoral dissertation, Toyota Technological Institute at Chicago).

Ng, B., & Abugharbieh, R. (2011). Generalized sparse regularization with application to fMRI brain decoding. In *Proceedings of the International Conference on Information Processing in Medical Imaging* (pp. 612–623).

Niculae, V., & Blondel, M. (2017). A regularized framework for sparse and structured neural attention. In *Adavnces in Neural Information Processing Systems* (pp. 3340–3350).

Niculae, V., Martins, A. F., Blondel, M., & Cardie, C. (2018). SparseMAP: differentiable sparse structured inference. In *Proceedings of the International Conference on Machine Learning*.

Nooner, K. B., Colcombe, S. J., Tobe, R. H., Mennes, M., Benedict, M. M., Moreno, A. L., ... Milham, M. P. (2012). The NKI Rockland Sample: a Model for Accelerating the Pace of Discovery Science in Psychiatry. *Frontiers in Neuroscience*, *6*, 152.

Nowak, A., Folqué, D., & Bruna, J. (2018). Divide and conquer networks. *Proceedings of the International Conference on Learning Representation*.

Ogawa, S., Lee, T.-M., Kay, A. R., & Tank, D. W. (1990). Brain magnetic resonance imaging with contrast dependent on blood oxygenation. *Proceedings of the National Academy of Sciences*, *87*(24), 9868–9872.

Olshausen, B. A., & Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, *37*(23), 3311–3325.

Ortega, J. M., & Rheinboldt, W. C. (1970). *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press.

Pan, S. J., & Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359.

Papadopoulos Orfanos, D., Michel, V., Schwartz, Y., Pinel, P., Moreno, A., Le Bihan, D., & Frouin, V. (2017). The Brainomics/Localizer database. *NeuroImage*, *144*, 309–314.

Paszke, A., Gross, S., Chintala, S., & Chanan, G. (2017). Pytorch: tensors and dynamic neural networks in Python with strong GPU acceleration.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier.

Pearlmutter, B. A. (1994). Fast exact multiplication by the Hessian. *Neural computation*, *6*(1), 147–160.

Pedregosa, F. (2015). *Feature extraction and supervised learning on fMRI: from practice to theory* (Doctoral dissertation, Université Pierre et Marie Curie-Paris VI).

Pedregosa, F., Eickenberg, M., Ciuciu, P., Thirion, B., & Gramfort, A. (2015). Data-driven HRF estimation for encoding and decoding models. *NeuroImage*, *104*, 209–220.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, É. (2011). Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Peng, Y., Meng, D., Xu, Z., Gao, C., Yang, Y., & Zhang, B. (2014). Decomposable nonlocal tensor dictionary learning for multispectral image denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2949–2956).

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: global vectors for word representation. In *Proceeding of the Conference on EMNLP* (pp. 1532–43).

Pereira, F., Mitchell, T., & Botvinick, M. (2009). Machine learning classifiers and fMRI: a tutorial overview. *NeuroImage*, *45*(1 Suppl), S199–S209.

Petersen, S. E., Fox, P. T., Posner, M. I., Mintun, M., & Raichle, M. E. (1989). Positron emission tomographic studies of the processing of singe words. *Journal of cognitive neuroscience*, *1*(2), 153–170.

Pilanci, M., & Wainwright, M. (2015). Iterative hessian sketch: fast and accurate solution approximation for constrained least squares. *Journal of Machine Learning Research*, *17*, 1–33.

Pinel, P., & Dehaene, S. (2013). Genetic and environmental contributions to brain activation during calculation. *NeuroImage*, *81*, 306–316.

Pinel, P., Thirion, B., Meriaux, S., Jobert, A., Serres, J., Bihan, D. L., ... Dehaene, S. (2007a). Fast reproducible identification and large-scale databasing of individual functional cognitive networks. *BMC neuroscience*, *8*, 91.

Pinel, P., Thirion, B., Meriaux, S., Jobert, A., Serres, J., Le Bihan, D., ... Dehaene, S. (2007b). Fast reproducible identification and large-scale databasing of individual functional cognitive networks. *BMC Neuroscience*, *8*, 91.

Pinho, A. L., Amadon, A., Ruest, T., Fabre, M., Dohmatob, E., Denghien, I., ... Thirion, B. (2018). Individual Brain Charting, a high resolution fMRI dataset for cognitive mapping. *Scientific Data*, *5*, 180105.

Poldrack, R. A., Baker, C. I., Durnez, J., Gorgolewski, K. J., Matthews, P. M., Munafò, M. R., ... Yarkoni, T. (2017). Scanning the horizon: towards transparent and reproducible neuroimaging research. *Nature Reviews Neuroscience*, *18*(2), 115–126.

Poldrack, R. A., Barch, D. M., Mitchell, J., Wager, T. D., Wagner, A. D., Devlin, J. T., ... Milham, M. (2013). Toward open sharing of task-based fMRI data: the OpenfMRI project. *Frontiers in Neuroinformatics*, *7*, 12.

Poldrack, R. A., Clark, J., Pare-Blagoev, E., Shohamy, D., Creso Moyano, J., Myers, C., & Gluck, M. (2001). Interactive memory systems in the human brain. *Nature*, *414*(6863), 546–550.

Poldrack, R. A., Congdon, E., Triplett, W., Gorgolewski, K. J., Karlsgodt, K., Mumford, J. A., ... Cannon, T. et al. (2016). A phenome-wide examination of neural and cognitive function. *Scientific Data*, *3*, 160110.

Poldrack, R. A., Halchenko, Y. O., & Hanson, S. J. (2009). Decoding the large-scale structure of brain function by classifying mental states across individuals. *Psychological Science*, *20*(11), 1364–1372.

Poldrack, R. A., Nichols, T., & Mumford, J. A. (2011). *Handbook of functional MRI data analysis*. Cambridge University Press.

Poldrack, R. A., & Yarkoni, T. (2016). From brain maps to cognitive ontologies: informatics and the search for mental structure. *Annual Review of Psychology*, *67*(1), 587–612.

Rabiner, L. R. (N.d.). A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE* (Vol. 77, 2, pp. 257–286).

Rahim, M., Thirion, B., & Varoquaux, G. (2017). Population-shrinkage of covariance to estimate better brain functional connectivity. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 460–468).

Raichle, M. E., & Mintun, M. A. (2006). Brain work and brain imaging. *Annual Review of Neuroscience*, *29*, 449–476.

Raskutti, G., & Mahoney, M. (2015). Statistical and algorithmic perspectives on randomized sketching for ordinary least-squares. In *Proceedings of the International Conference on Machine Learning* (pp. 617–625).

Razaviyayn, M., Hong, M., & Luo, Z.-Q. (2013). A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, *23*(2), 1126–1153.

Recht, B., & Ré, C. (2013). Parallel stochastic gradient algorithms for large-scale matrix completion. *Mathematical Programming Computation*, *5*(2), 201–226.

Rennie, J. D. M., & Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the International Conference on Machine Learning* (pp. 713–719).

Rizk-Jackson, A., Aron, A. R., & Poldrack, R. (N.d.). Classification learning and stop-signal (one year test-retest). https://openfmri.org/dataset/dsoooo17.

Rokhlin, V., Szlam, A., & Tygert, M. (2009). A randomized algorithm for principal component analysis. *SIAM Journal on Matrix Analysis and Applications*, *31*(3), 1100–1124.

Roy, A. K., Shehzad, Z., Margulies, D. S., Kelly, A. C., Uddin, L. Q., Gotimer, K., … Milham, M. P. (2009). Functional connectivity of the human amygdala using resting state fMRI. *NeuroImage*, *45*(2), 614–626.

Sabuncu, M. R., Singer, B. D., Conroy, B., Bryan, R. E., Ramadge, P. J., & Haxby, J. V. (2009). Function-based intersubject alignment of human cortical anatomy. *Cerebral Cortex*, *20*(1), 130–140.

Sakoe, H., & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *26*, 43–49.

Salimi-Khorshidi, G., Smith, S. M., Keltner, J. R., Wager, T. D., & Nichols, T. E. (2009). Meta-analysis of neuroimaging data: a

comparison of image-based and coordinate-based pooling of studies. *NeuroImage*, *45*(3), 810–823.

Sarlos, T. (2006). Improved approximation algorithms for large matrices via random projections. In *Proceedings of the IEEE Symposium on Foundations of Computer Science* (pp. 143–152).

Saxe, R., Brett, M., & Kanwisher, N. (2006). Divide and conquer: a defense of functional localizers. *NeuroImage*, *30*(4), 1088–1096, discussion 1097–1099.

Schonberg, T., Fox, C., Mumford, J. A., Congdon, C., Trepel, C., & Poldrack, R. A. (2012). Decreasing ventromedial prefrontal cortex activity during sequential risk-taking: an fMRI investigation of the balloon analog risk task. *Frontiers in Neuroscience*, *6*, 80.

Schwartz, Y., Thirion, B., & Varoquaux, G. (2013). Mapping paradigm ontologies to and from the brain. In *Advances in Neural Information Processing Systems* (pp. 1673–1681).

Shafto, M. A., Tyler, L. K., Dixon, M., Taylor, J. R., Rowe, J. B., Cusack, R., ... Matthews, F. E. (2014). The Cambridge Centre for Ageing and Neuroscience (Cam-CAN) study protocol: a cross-sectional, lifespan, multidisciplinary examination of healthy cognitive ageing. *BMC Neurology*, *14*, 204.

Smith, D. A., & Eisner, J. (2006). Minimum risk annealing for training log-linear models. In *Proceedings of COLING/ACL* (pp. 787–794).

Smith, S. M., Fox, P. T., Miller, K. L., Glahn, D. C., Fox, P. M., Mackay, C. E., ... Laird, A. R. (2009). Correspondence of the brain's functional architecture during activation and rest. *Proceedings of the National Academy of Sciences*, *106*(31), 13040–13045.

Smith, S. M., Hyvärinen, A., Varoquaux, G., Miller, K. L., & Beckmann, C. F. (2014). Group-pca for very large fMRI datasets. *NeuroImage*, *101*, 738.

Smith, S. M., Nichols, T. E., Vidaurre, D., Winkler, A. M., Behrens, T. E., Glasser, M. F., ... Miller, K. L. (2015). A positive-negative mode of population covariation links brain connectivity, demographics and behavior. *Nature Neuroscience*, *18*(11), 1565.

Soltani-Farani, A., Rabiee, H. R., & Hosseini, S. A. (2015). Spatial-aware dictionary learning for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, *53*(1), 527–541.

Srebro, N., Rennie, J., & Jaakkola, T. S. (2004). Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems* (pp. 1329–1336).

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929–1958.

Stoyanov, V., & Eisner, J. (2012). Minimum-risk training of approximate CRF-based NLP systems. In *Proceedings of NAACL* (pp. 120–130).

Sudlow, C., Gallacher, J., Allen, N., Beral, V., Burton, P., Danesh, J., ... Landray, M. (2015). UK BioBank: an open access resource for

identifying the causes of a wide range of complex diseases of middle and old age. *PLoS Medicine*, *12*(3), 1–10.

Sulanke, R. A. (2003). Objects counted by the central Delannoy numbers. *Journal of Integer Sequences*, *6*(1), 3.

Sutton, C., McCallum, A. et al. (2012). An introduction to conditional random fields. *Foundations and Trends on Machine Learning*, *4*(4), 267–373.

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

Takács, G., Pilászy, I., Németh, B., & Tikk, D. (2009). Scalable collaborative filtering approaches for large recommender systems. *Journal of Machine Learning Research*, *10*, 623–656.

Tavor, I., Jones, O. P., Mars, R. B., Smith, S. M., Behrens, T. E., & Jbabdi, S. (2016). Task-free MRI predicts individual differences in brain activity during task performance. *Science*, *352*(6282), 216–220.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, *58*(1), 267–288.

Tom, S. M., Fox, C. R., Trepel, C., & Poldrack, R. A. (2007). The neural basis of loss aversion in decision-making under risk. *Science*, *315*(5811), 515–518.

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, *6*, 1453–1484.

Turner, J. A., & Laird, A. R. (2012). The cognitive paradigm ontology: design and application. *Neuroinformatics*, *10*(1), 57–66.

Uncapher, M. R., Hutchinson, J. B., & Wagner, A. D. (2011). Dissociable effects of top-down and bottom-up attention during episodic encoding. *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, *31*(35), 12613–12628.

Uttal, W. R. (2001). *The new phrenology: the limits of localizing cognitive processes in the brain.* The MIT press.

Vagharchakian, L., Dehaene-Lambertz, G., Pallier, C., & Dehaene, S. (2012). A temporal bottleneck in the language comprehension network. *The Journal of Neuroscience*, *32*, 9089–9102.

Van der Vaart, A. W. (2000). *Asymptotic statistics*. Cambridge University Press.

Van Essen, D. C., Ugurbil, K., Auerbach, E., Barch, D., Behrens, T. E. J., Bucholz, R., ... Curtiss, S. W. (2012). The Human Connectome Project: a data acquisition perspective. *NeuroImage*, *62*(4), 2222–2231.

Vane, G. (1987). First results from the airborne visible/infrared imaging spectrometer (AVIRIS). In *Annual Technical Symposium of the International Society of Optics and Photonic* (pp. 166–175).

Varoquaux, G., Gramfort, A., Pedregosa, F., Michel, V., & Thirion, B. (2011). Multi-subject dictionary learning to segment an atlas of brain spontaneous activity. *Proceedings of the International Conference on Information Processing in Medical Imaging*, *22*, 562.

Verdu, S., & Poor, H. V. (1987). Abstract dynamic programming models under commutativity conditions. *SIAM Journal on Control and Optimization*, *25*(4), 990–1006.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, *13*(2), 260–269.

Wager, S., Wang, S., & Liang, P. S. (2013). Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems* (pp. 351–359).

Wager, T. D., Atlas, L. Y., Lindquist, M. A., Roy, M., Woo, C.-W., & Kross, E. (2013). An fMRI-based neurologic signature of physical pain. *New England Journal of Medicine*, *368*(15), 1388–1397.

Wager, T. D., Davidson, M. L., Hughes, B. L., Lindquist, M. A., & Ochsner, K. N. (2008). Prefrontal-subcortical pathways mediating successful emotion regulation. *Neuron*, *59*, 1037–1050.

Wainwright, M. J., & Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends on Machine Learning*, *1*(1–2), 1–305.

Weng, S.-J., Wiggins, J. L., Peltier, S. J., Carrasco, M., Risi, S., Lord, C., & Monk, C. S. (2010). Alterations of resting state functional connectivity in the default network in adolescents with autism spectrum disorders. *Brain Research*, *1313*, 202–214.

Wright, S. J. (2015). Coordinate descent algorithms. *Mathematical Programming*, *151*(1), 3–34.

Wu, T., Wang, L., Chen, Y., Zhao, C., Li, K., & Chan, P. (2009). Changes of functional connectivity of the motor network in the resting state in Parkinson's disease. *Neuroscience Letters*, *460*(1), 6–10.

Xue, G., Aron, A. R., & Poldrack, R. A. (2008). Common neural substrates for inhibition of spoken and manual responses. *Cerebral Cortex*, *18*, 1923–1932.

Xue, G., & Poldrack, R. A. (2007). The neural substrates of visual perceptual learning of words: implications for the visual word form area hypothesis. *Journal of Cognitive Neuroscience*, *19*, 1643–1655.

Xue, Y., Liao, X., Carin, L., & Krishnapuram, B. (2007). Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*, *8*(Jan), 35–63.

Yamashita, O., Sato, M., Yoshioka, T., Tong, F., & Kamitani, Y. (2008). Sparse estimation automatically selects voxels relevant for the decoding of fMRI activity patterns. *NeuroImage*, *42*(4), 1414–1429.

Yarkoni, T., Poldrack, R. A., Nichols, T. E., Van Essen, D. C., & Wager, T. D. (2011). Large-scale automated synthesis of human functional neuroimaging data. *Nature Methods*, *8*(8), 665–670.

Yeo, T. B. T., Krienen, F. M., Sepulcre, J., Sabuncu, M. R., Lashkari, D., Hollinshead, M., ... Buckner, R. L. (2011). The organization of the human cerebral cortex estimated by intrinsic functional connectivity. *Journal of Neurophysiology*, *106*(3), 1125–1165.

Yu, H.-F., Hsieh, C.-J., & Dhillon, I. (2012). Scalable coordinate descent approaches to parallel matrix factorization for recommender

systems. In *Proceedings of the International Conference on Data Mining* (pp. 765–774).

Yuan, M., & Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *68*(1), 49–67.

Zhang, Y., Roughan, M., Willinger, W., & Qiu, L. (2009). Spatio temporal compressive sensing and internet traffic matrices. *IEEE/ACM Transactions on Networking*, *20*(3), 662–676.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Methodological)*, *67*(2), 301–320.

Zou, H., Hastie, T., & Tibshirani, R. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, *15*(2), 265–286.

APPENDICES

A

PROOFS OF CHAPTER 7 — SOMF AND SAMM ANALYSIS

This appendix to Chapter 5 contain the detailed proofs of Proposition 5.3 and Proposition 5.1. This section can be skipped at first reading.

A.1 PROOFS OF CONVERGENCE

We introduce three lemmas that will be crucial to prove SAMM convergence, before establishing it by proving Proposition 5.3. Finally, we show that SOMF is indeed an instance of SAMM (*i.e.* meets the assumptions **(C)**–**(I)**), proving Proposition 5.1.

A.1.1 *Basic properties of the surrogates, estimate stability*

Let us first recall a basic inequality for L-Lipschitz continuous functions. This inequality is useful in the demonstration of Lemma A.2 and Proposition 5.3. Let $f : \Theta \subset \mathbb{R}^K \to \mathbb{R}$ be a function with L-Lipschitz gradient. That is, for all $\theta, \theta' \in \Theta, \|\nabla f(\theta) - \nabla f(\theta')\|_2 \leqslant L\|\theta - \theta'\|_2$. Then, for all $\theta, \theta' \in \Theta$,

$$f(\theta') \leqslant f(\theta) + \nabla f(\theta)^\top (\theta' - \theta) + \frac{L}{2}\|\theta - \theta'\|_2^2. \tag{A.1}$$

In this section, we derive an important result on the stability and optimality of the sequence $(\theta_t)_t$, formalized in Lemma A.3 — introduced in the main text. We first introduce a numerical lemma on the boundedness of well-behaved deterministic and random sequence.

Lemma A.1 (Bounded quasi-geometric sequences)**.** *Let $(x_t)_t$ be a sequence in $\mathbb{R}^+$, $u : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, $t_0 \in \mathbb{N}$ and $\alpha \in [0, 1)$ such that, for all $t \geqslant t_0$, $x_t \leqslant \alpha x_{t-1} + u(x_t, x_{t-1})$, where $u(x, y) \in o(x + y)$ for $x, y \to \infty$. Then $(x_t)_t$ is bounded.*

Let now $(X_t)_t$ be a random sequence in $\mathbb{R}^+$, such that $\mathbb{E}[X_t] < \infty$. We define $(\mathcal{F}_t)_t$ the filtration adapted to $(X_t)_t$. If, for all $t > t_0$, there exists a σ-algebra $\mathcal{F}_{t'}$ such that $\mathcal{F}_{t-1} \subseteq \mathcal{F}_{t'} \subseteq \mathcal{F}_t$ and

$$\mathbb{E}[X_t | \mathcal{F}_{t'}] \leqslant \alpha X_{t-1} + u(X_t, X_{t-1}),$$

then $(X_t)_t$ is bounded almost surely.

Proof. We first focus on the deterministic case. Assume that $(x_t)_t$ is not bounded. Then there exists a subsequence of $(x_t)_t$ that diverges towards $+\infty$. We assume without loss of generality that $(x_t)_t \to \infty$.

Then, $x_t + x_{t-1} \to \infty$ and for all $\epsilon > 0$, using the asymptotic bounds on u, there exists $t_1 \geqslant t_0$ such that

$$\forall t \geqslant t_1,\ x_t \leqslant \alpha x_{t-1} + \epsilon(x_t + x_{t-1})$$
$$\text{and therefore} \quad x_t \leqslant \frac{\alpha+\epsilon}{1-\epsilon} x_{t-1}.$$

Setting ϵ small enough, we obtain that x_t is bounded by a geometrically decreasing sequence after t_1, and converges to 0, which contradicts our hypothesis. This is enough to conclude.

In the random case, we consider a realization of $(X_t)_t$ that is not bounded, and assumes without loss of generality that it diverges to $+\infty$. Following the reasoning above, there exists $\beta < 1$, $t_1 > 0$, such that for all $t > t_1$, $\mathbb{E}[X_t|\mathcal{F}_{t'}] \leqslant \beta X_{t-1}$, where $\mathcal{F}_{t-1} \subseteq \mathcal{F}_{t'} \subseteq \mathcal{F}_t$. Taking the expectation conditioned on $\mathcal{F}_{t-1}$, $\mathbb{E}[X_t|\mathcal{F}_{t-1}] \leqslant \beta X_{t-1}$, as X_{t-1} is deterministic conditioned on $\mathcal{F}_{t-1}$. Therefore X_t is a supermartingale beyond a certain time. As $\mathbb{E}[X_t] < \infty$, Doob's forward convergence lemma on discrete martingales (Doob, 1990) ensures that $(X_t)_t$ converges almost surely. Therefore the event $\{(X_t)_t \text{ is not bounded}\}$ cannot happen on a set with non-zero probability, less it would lead to a contradiction. The lemma follows. □

We then derive some properties of the approximate surrogate functions used in SAMM. The proof is adapted from Mairal (2013b).

Lemma A.2 (Basic properties of approximate surrogate functions)**.** *Consider any sequence of iterates $(\theta_t)_t$ and assume there exists $\epsilon > 0$ such that $g_t \in \mathcal{T}_{L,\rho}(f_t, \theta_{t-1}, \epsilon)$ for all $t \geqslant 1$. Define $h_t \triangleq g_t - f_t$ for all $t \geqslant 1$, $\bar{h}_0 \triangleq h_0$ and $\bar{h}_t \triangleq (1-w_t)\bar{h}_{t-1} + w_t h_t$. Under assumptions (D) – (G),*

(i) *$(\nabla h_t(\theta_{t-1}))_{t>0}$ is uniformly bounded and there exists R' such that $\{\nabla h_t\}_t$ is uniformly bounded by R'.*

(ii) *$(h_t)_t$ and $(\bar{h}_t)_t$ are uniformly R'-Lipschitz, $(g_t)_t$ and $(\bar{g}_t)_t$ are uniformly $(R+R')$-Lipschitz.*

Proof. We first prove (i). We set $\alpha > 0$ and define $\theta' = \theta_t - \alpha \frac{\nabla h_t(\theta_t)}{\|\nabla h_t(\theta_t)\|_2}$. As h_t has a L-Lipschitz gradient on $\mathbb{R}^K$, using Taylor's inequality (A.1)

$$h_t(\theta') \leqslant h_t(\theta_t) - \alpha\|\nabla h_t(\theta_t)\|_2 + \frac{L\alpha^2}{2} \tag{A.2}$$
$$\|\nabla h_t(\theta_t)\|_2 \leqslant \frac{1}{\alpha}(h_t(\theta_t) - h_t(\theta')) + \frac{L\alpha}{2} \leqslant \frac{2}{\alpha}\epsilon + \frac{L\alpha}{2},$$

where we use $h_t(\theta_t) < \epsilon$ and $-h_t(\theta'_t) \leqslant \epsilon$ from the assumption $g_t \in \mathcal{T}_{L,\rho}(f_t, \theta_{t-1}, \epsilon)$. Moreover, by definition, ∇h_t exists and is L-lipschitz for all t. Therefore, $\forall\, t \geqslant 1$,

$$\|\nabla h_t(\theta)\|_2 \leqslant \|\nabla h_t(\theta_t)\|_2 + L\|\theta_{t-1} - \theta\|_2$$

Since Θ is compact and $(\|\nabla h_t(\theta_t)\|_2)_{t\geqslant 1}$ is bounded in (A.2), ∇h_t is bounded by R' independent of t. (ii) follows by basic considerations on Lipschitz functions. □

Finally, we prove a result on the stability of the estimates, that derives from combining the properties of $(g_t)_t$ and the geometric decrease assumption **(I)**.

Lemma A.3 (Estimate stability under SAMM approximation)**.** *In the same setting as Lemma A.2, with the additional assumption **(I)** (expected linear decrease of $\bar{g}_t$ suboptimality), the sequence $\|\theta_t - \theta_{t-1}\|_2$ converges to 0 as fast as $(w_t)_t$, and θ_t is asymptotically an exact minimizer. Namely, almost surely,*

$$\|\theta_t - \theta_{t-1}\|_2 \in \mathcal{O}(w_t) \text{ and } \bar{g}_t(\theta_t) - \bar{g}_t(\theta_t^\star) \in \mathcal{O}(w_t^2).$$

Proof. We first establish the result when a deterministic version of **(I)** holds, as it makes derivations simpler to follow.

A.1.1.1 *Deterministic decrease rate*

We temporarily assume that decays are deterministic.

(I_{det}) For all $t > 0$, $\bar{g}_t(\theta_t) < \bar{g}_t(\theta_{t-1})$. Moreover, there exists $\mu > 0$ such that, for all $t > 0$

$$\begin{aligned}\bar{g}_t(\theta_t) - \bar{g}_t(\theta_t^\star) &\leqslant (1-\mu)(\bar{g}_t(\theta_{t-1}) - \bar{g}_t(\theta_t^\star))\\ \text{where} \quad \theta_t^\star &= \operatorname*{argmin}_{\theta\in\Theta} \bar{g}_t(\theta),\end{aligned}$$

We introduce the following auxiliary positive values, that we will seek to bound in the proof:

$$\begin{aligned} A_t &\triangleq \|\theta_t - \theta_{t-1}\|_2, \quad B_t \triangleq \|\theta_t - \theta_t^\star\|_2,\\ C_t &\triangleq \|\theta_t^\star - \theta_{t-1}^\star\|_2, \quad D_t \triangleq \bar{g}_t(\theta_t) - \bar{g}_t(\theta_t^\star).\end{aligned}$$

Our goal is to bound A_t. We first relate it to C_t and B_t using convexity of ℓ_2 norm:

$$A_t^2 \leqslant 3B_t^2 + 3B_{t-1}^2 + 3C_t^2. \tag{A.3}$$

As $\theta_t^\star$ is the minimizer of $\bar{g}_t$, by strong convexity of $(\bar{g}_t)_t$,

$$\frac{\rho}{2} B_t^2 = \frac{\rho}{2}\|\theta_t - \theta_t^\star\|_2^2 \leqslant D_t, \tag{A.4}$$

while we also have

$$\begin{aligned}&\frac{\rho}{2}\|\theta_t^\star - \theta_{t-1}^\star\|_2^2 \leqslant \bar{g}_t(\theta_{t-1}^\star) - \bar{g}_t(\theta_t^\star)\\ &\leqslant (1-w_t)\big(\bar{g}_{t-1}(\theta_{t-1}^\star) - \bar{g}_{t-1}(\theta_t^\star)\big) + w_t\big(g_t(\theta_{t-1}^\star) - g_t(\theta_t^\star)\big)\\ &\leqslant w_t(R+R')\|\theta_t^\star - \theta_{t-1}^\star\|_2, \text{ and thus } C_t \leqslant w_t\frac{2Q}{\rho}.\end{aligned} \tag{A.5}$$

The second inequalities holds because $\theta_{t-1}^\star$ is a minimizer of $\bar{g}_{t-1}$ and g_t is Q-Lipschitz, where $Q \triangleq R + R'$, using Lemma A.2. Replacing (A.4) and (A.5) in (A.3) yields

$$A_t^2 \leqslant \frac{6}{\rho}(D_t + D_{t-1}) + \frac{12Q^2}{\rho} w_t^2, \tag{A.6}$$

and we are left to show that $D_t \in \mathcal{O}(w_t^2)$ to conclude. For this, we decompose the inequality from **(I_{det})** into

$$\begin{aligned}
D_t &\leqslant (1-\mu)(\bar{g}_t(\theta_{t-1}) - \bar{g}_t(\theta_t^\star)) \\
&= (1-\mu)\Big(w_t\big(g_t(\theta_{t-1}) - g_t(\theta_t)\big) + w_t\big(g_t(\theta_t) - g_t(\theta_t^\star)\big)\Big) \\
&\quad + (1-\mu)\Big((1-w_t)\big(\bar{g}_{t-1}(\theta_{t-1}) - \bar{g}_{t-1}(\theta_{t-1}^\star)\big) \\
&\qquad\qquad + (1-w_t)\big(\bar{g}_{t-1}(\theta_{t-1}^\star) - \bar{g}_{t-1}(\theta_t^\star)\big)\Big) \\
&\leqslant (1-\mu)(w_t Q(A_t + B_t) + D_{t-1}), \qquad \text{(A.7)}
\end{aligned}$$

where the second inequality holds for the same reasons as in (A.5). Injecting (A.4) and (A.6) in (A.7), we obtain

$$\tilde{D}_t \leqslant (1-\mu)\tilde{D}_{t-1}\frac{w_{t-1}^2}{w_t^2} + u(\tilde{D}_t, \tilde{D}_{t-1}), \qquad \text{(A.8)}$$

where we define $\tilde{D}_t \triangleq \frac{D_t}{w_t^2}$.

Injecting (A.4) and (A.6) in (A.7), we obtain

$$\tilde{D}_t \leqslant (1-\mu)\tilde{D}_{t-1}\frac{w_{t-1}^2}{w_t^2} + u(\tilde{D}_t, \tilde{D}_{t-1}), \quad \text{where}$$

$$u(\tilde{D}_t, \tilde{D}_{t-1}) \triangleq (1-\mu)\tilde{Q}\left(\sqrt{3(\tilde{D}_t + \tilde{D}_{t-1}\frac{w_{t-1}^2}{w_t^2}) + \tilde{Q}} + \sqrt{\tilde{D}_t}\right).$$

From assumption **(G)**, $\frac{w_{t-1}^2}{w_t^2} \to 1$, and we have, from elementary comparisons, that $u(\tilde{D}_t, \tilde{D}_{t-1}) \in o(\tilde{D}_t + \tilde{D}_{t-1})$ if $D_t \to \infty$. Using the determistictic result of Lemma A.1, this ensures that $\tilde{D}_t$ is bounded. Combined with (A.4), this allows to conclude.

A.1.1.2 *Stochastic decrease rates*

In the general case **(I)**, the inequalities (A.4), (A.5) and (A.6) holds, and (A.8) is replaced by

$$\mathbb{E}[\tilde{D}_t | \mathcal{F}_{t-\frac{1}{2}}] \leqslant (1-\mu)\tilde{D}_{t-1}\frac{w_{t-1}^2}{w_t^2} + u(\tilde{D}_t, \tilde{D}_{t-1}),$$

Taking the expectation of this inequality and using Jensen inequality, we show that (A.7) holds when replacing $\tilde{D}_t$ by $\mathbb{E}[\tilde{D}_t]$. This shows that $\mathbb{E}[D_t] \in \mathcal{O}(w_t^2)$ and thus $\mathbb{E}[D_t] < \infty$. The result follows from Lemma A.1, that applies as $\mathcal{F}_{t-1} \subseteq \mathcal{F}_{t-\frac{1}{2}} \subseteq \mathcal{F}_t$. □

A.1.2 *Convergence of SAMM — Proof of Proposition 5.3*

We now proceed to prove the Proposition 5.3, that extends the stochastic majorization-minimization framework to allow approximations in both majorization and minimizations steps.

Proof of Proposition 5.3. We adapt the proof of Proposition 3.3 from Mairal (2013b) (reproduced as Proposition 5.2 in our work). Relaxing

tightness and majorizing hypotheseses introduces some extra error terms in the derivations. Assumption **(H)** allows to control these extra terms without breaking convergence. The stability Lemma A.3 is important in steps 3 and 5.

A.1.2.1 *Almost sure convergence of* $(\bar{g}_t(\theta_t))$

We control the positive expected variation of $(g_t(\theta_t))_t$ to show that it is a converging quasi-martingale. By construction of $\bar{g}_t$ and properties of the surrogates $g_t \in \mathcal{T}_{\rho,L}(f_t, \theta_{t-1}, \epsilon_t)$, where ϵ_t is a non-negative sequence that meets **(H)**,

$$\begin{aligned}
&\bar{g}_t(\theta_t) - \bar{g}_{t-1}(\theta_{t-1}) \\
&= (\bar{g}_t(\theta_t) - \bar{g}_t(\theta_{t-1})) + w_t(g_t(\theta_{t-1}) - \bar{g}_{t-1}(\theta_{t-1})) \\
&\leqslant w_t(g_t(\theta_{t-1}) - \bar{g}_{t-1}(\theta_{t-1})) \\
&\leqslant w_t(g_t(\theta_{t-1}) - f_t(\theta_{t-1})) + w_t(f_t(\theta_{t-1}) - \bar{f}_{t-1}(\theta_{t-1})) \\
&\quad + w_t(\bar{f}_{t-1}(\theta_{t-1}) - \bar{g}_{t-1}(\theta_{t-1})) \\
&\leqslant w_t(f_t(\theta_{t-1}) - \bar{f}_{t-1}(\theta_{t-1})) + w_t(\bar{\epsilon}_{t-1} + \epsilon_t), \qquad \text{(A.9)}
\end{aligned}$$

where the average error sequence $(\bar{\epsilon}_t)_t$ is defined recursively: $\bar{\epsilon}_0 \triangleq \epsilon_0$ and $\bar{\epsilon}_t \triangleq (1 - w_t)\epsilon_{t-1} + w_t\epsilon_t$. The first inequality uses $\bar{g}_t(\theta_t) \leqslant \bar{g}_t(\theta_{t-1})$. To obtain the forth inequality we observe

$$g_t(\theta_{t-1}) - f_t(\theta_{t-1}) < \epsilon_t$$

by definition of ϵ_t and $\bar{f}_t(\theta_{t-1}) - \bar{g}_t(\theta_{t-1}) \leqslant \bar{\epsilon}_t$, which can easily be shown by induction on t. Then, taking the conditional expectation with respect to $\mathcal{F}_{t-1}$,

$$\begin{aligned}
&\mathbb{E}[\bar{g}_t(\theta_t) - \bar{g}_{t-1}(\theta_{t-1})|\mathcal{F}_{t-1}] \\
&\leqslant w_t \sup_{\theta\in\Theta} |f(\theta) - \bar{f}_{t-1}(\theta)| + w_t(\bar{\epsilon}_{t-1} + \mathbb{E}[\epsilon_t|\mathcal{F}_{t-1}]). \qquad \text{(A.10)}
\end{aligned}$$

We have used the fact that ϵ_{t-1} is deterministic with respect to $\mathcal{F}_{t-1}$. To ensure convergence, we must bound both terms in (A.10): the first term is the same as in the original proof with exact surrogate, while the second is the perturbative term introduced by the approximation sequence $(\epsilon_t)_t$. We use Lemma B.7 from Mairal (2013b), derived from the theory of empirical processes: $\mathbb{E}[\sup_{\theta\in\Theta} |f(\theta) - \bar{f}_{t-1}(\theta)|] = \mathcal{O}(w_t t^{1/2})$, and thus

$$\sum_{t=1}^{\infty} w_t \mathbb{E}[\sup_{\theta\in\Theta} |f(\theta) - \bar{f}_{t-1}(\theta)|] < C \sum_{t=1}^{\infty} t^{1/2} w_t^2 < \infty \qquad \text{(A.11)}$$

where C is a constant, as $t^{1/2}w_t^2 = t^{1/2-2u}$ and $u > 3/4$ from **(G)**. Let us now focus on the second term of (A.10). Defining, for all $1 \leqslant i \leqslant t$, $w_i^t = w_i \prod_{j=i+1}^{t}(1 - w_j)$,

$$\mathbb{E}[\bar{\epsilon}_t] = \sum_{i=1}^{t} w_i^t \mathbb{E}[\epsilon_t] \leqslant w_t \sum_{i=1}^{t} \mathbb{E}[\epsilon_t].$$

We set $\eta > 0$ so that $2(u-1)-\eta > -1$. Assumption **(H)** ensures $\mathbb{E}[\epsilon_t] \in \mathcal{O}(t^{2(u-1)-\eta})$, which allows to bound the partial sum

$$\sum_{i=1}^{t} \mathbb{E}[\epsilon_i] \in \mathcal{O}(t^{2u-1-\eta}).$$

Therefore, we have

$$\begin{aligned} &w_t \mathbb{E}[\bar{\epsilon}_{t-1} + \mathbb{E}[\epsilon_t|\mathcal{F}_{t-1}]] = w_t \mathbb{E}[\epsilon_{t-1}] + w_t \mathbb{E}[\epsilon_t] \\ &\leqslant w_t^2 \Big(\sum_{i=1}^{t} \mathbb{E}[\epsilon_t] \Big) + w_t \mathbb{E}[\epsilon_t] \\ &\leqslant A t^{2u-2u-1-\eta} + B t^{2u-u-2-\eta} \leqslant C t^{-1-\eta}, \end{aligned} \tag{A.12}$$

where we use $u < 1$ on the third line and the definition of $(w_t)_t$ on the second line. Thus $\sum_{t=1}^{\infty} w_t \mathbb{E}[\bar{\epsilon}_{t-1} + \mathbb{E}[\epsilon_t|\mathcal{F}_{t-1}]] < \infty$. We use quasi-martingale theory to conclude, as in Mairal (2013b). We define the variable δ_t to be 1 if $\mathbb{E}[\bar{g}_t(\theta_t) - \bar{g}_{t-1}(\theta_{t-1})|\mathcal{F}_{t-1}] \geqslant 0$, and 0 otherwise. As all terms of (A.10) are positive:

$$\begin{aligned} &\sum_{t=1}^{\infty} \mathbb{E}[\delta_t(\bar{g}_t(\theta_t) - \bar{g}_{t-1}(\theta_{t-1}))] \\ &= \sum_{t=1}^{\infty} \mathbb{E}[\delta_t \mathbb{E}[\bar{g}_t(\theta_t) - \bar{g}_{t-1}(\theta_{t-1})|\mathcal{F}_{t-1}]] \\ &\leqslant \sum_{t=1}^{\infty} w_t \mathbb{E}[\sup_{\theta \in \Theta} |f(\theta) - \bar{f}_{t-1}(\theta)| + \bar{\epsilon}_{t-1} + \mathbb{E}[\epsilon_t|\mathcal{F}_{t-1}]] < \infty. \end{aligned}$$

As $\bar{g}_t$ are bounded from below ($\bar{f}_t$ is bounded from **(D)** and we easily show that $\bar{\epsilon}_t$ is bounded), we can a quasi-martingale convergence theorem originally found in Métivier (1982). It ensures that $(g_t(\theta_t))_{t \geqslant 1}$ converges almost surely to an integrable random variable $g^\star$, and that $\sum_{t=1}^{\infty} \mathbb{E}[|\mathbb{E}[\bar{g}_t(\theta_t) - \bar{g}_{t-1}(\theta_{t-1})|\mathcal{F}_{t-1}]] < \infty$ almost surely.

A.1.2.2 *Almost sure convergence of* $\bar{f}(\theta_t)$

We rewrite the second inequality of (A.9), adding $\bar{\epsilon}_t$ on both sides:

$$\begin{aligned} 0 &\leqslant w_t\big(\bar{g}_{t-1}(\theta_{t-1}) - \bar{f}_{t-1}(\theta_{t-1}) + \bar{\epsilon}_{t-1}\big) \\ &\leqslant w_t\big(g_t(\theta_{t-1}) - f_t(\theta_{t-1})\big) + w_t\big(f_t(\theta_{t-1}) - \bar{f}_{t-1}(\theta_{t-1})\big) \\ &\quad + \big(\bar{g}_{t-1}(\theta_{t-1}) - \bar{g}_t(\theta_t)\big) + w_t \bar{\epsilon}_{t-1} \\ &\leqslant w_t\big(f_t(\theta_{t-1}) - \bar{f}_{t-1}(\theta_{t-1})\big) + \big(\bar{g}_{t-1}(\theta_{t-1}) - \bar{g}_t(\theta_t)\big) \\ &\quad + w_t(\epsilon_t + \bar{\epsilon}_{t-1}), \end{aligned} \tag{A.13}$$

where the left side bound has been obtained in the last paragraph by induction and the right side bound arises from the definition of ϵ_t. Taking the expectation of (A.13) conditioned on $\mathcal{F}_{t-1}$, almost surely,

$$\begin{aligned} 0 \leqslant\ & w_t(f(\theta_{t-1}) - \bar{f}_{t-1}(\theta_{t-1})) \\ & - \mathbb{E}[\bar{g}_t(\theta_t) - \bar{g}_{t-1}(\theta_{t-1})|\mathcal{F}_{t-1}] + w_t(\bar{\epsilon}_{t-1} + \mathbb{E}[\epsilon_t|\mathcal{F}_{t-1}]), \end{aligned}$$

We separately study the three terms of the previous upper bound. The first two terms can undergo the same analysis as in Mairal (2013b). First, almost sure convergence of the sum

$$\sum_{t=1}^{\infty} \mathbb{E}\left[\left|\mathbb{E}[\bar{g}_t(\theta_t) - \bar{g}_{t-1}(\theta_{t-1})|\mathcal{F}_{t-1}]\right|\right]$$

implies that $\mathbb{E}\left[\bar{g}_t(\theta_t) - \bar{g}_{t-1}(\theta_{t-1})|\mathcal{F}_{t-1}\right]$ is the summand of an almost surely converging sum. Second, $w_t\left(f(\theta_{t-1}) - \bar{f}_{t-1}(\theta_{t-1})\right)$ is the summand of an absolutely converging sum with probability one, less it would contradict (A.11). To bound the third term, we have once more to control the perturbation introduced by $(\epsilon_t)_t$. We have $\sum_{t=1}^{\infty} w_t\bar{\epsilon}_{t-1} + w_t\mathbb{E}[\epsilon_t|\mathcal{F}_{t-1}] < \infty$ almost surely, otherwise Fubini's theorem would invalidate (A.12).

As the three terms are the summand of absolutely converging sums, the positive term $w_t(\bar{g}_{t-1}(\theta_{t-1}) - \bar{f}_{t-1}(\theta_{t-1}) + \bar{\epsilon}_{t-1})$ is the summand of an almost surely convergent sum. This is not enough to prove that $\bar{h}_t(\theta_t) \triangleq \bar{g}_t(\theta_t) - \bar{f}_t(\theta_t) \to_\infty 0$, hence we follow (Mairal, 2013b) and make use of its Lemma A.6. We define $X_t \triangleq \bar{h}_{t-1}(\theta_{t-1}) + \bar{\epsilon}_{t-1}$. As **(H)** holds, we use Lemma A.3, which ensures that $(\bar{h}_t)_{t\geqslant 1}$ are uniformly R'-Lipschitz and $\|\theta_t - \theta_{t-1}\|_2 = \mathcal{O}(w_t)$. Hence,

$$\begin{aligned}
|X_{t+1} - X_t| &\leqslant |\bar{h}_t(\theta_t) - \bar{h}_{t-1}(\theta_{t-1})| + |\bar{\epsilon}_t - \bar{\epsilon}_{t-1}| \\
&\leqslant R'\|\theta_t - \theta_{t-1}\|_2 + |\bar{\epsilon}_t - \bar{\epsilon}_{t-1}|, \quad \text{as } \bar{h}_t \text{ is } R'\text{-Lipschitz} \\
&\leqslant \mathcal{O}(w_t) + |\bar{\epsilon}_t - \bar{\epsilon}_{t-1}|, \quad \text{as } \|\theta_t - \theta_{t-1}\|_2 = \mathcal{O}(w_t)
\end{aligned}$$

From assumption **(H)**, $(\epsilon_t)_t$ and $(\bar{\epsilon}_t)_t$ are bounded. Therefore $|\bar{\epsilon}_t - \bar{\epsilon}_{t-1}| \leqslant w_t(|\epsilon_t| + |\bar{\epsilon}_{t-1}|) \in \mathcal{O}(w_t)$ and hence

$$|X_{t+1} - X_t| \leqslant \mathcal{O}(w_t).$$

Lemma A.6 from Mairal (2013b) then ensures that X_t converges to zero with probability one. Assumption **(H)** ensures that $\epsilon_t \to_\infty 0$ almost surely, from which we can easily deduce $\bar{\epsilon}_t \to_\infty 0$ almost surely. Therefore $\bar{h}_t(\theta_t) \to 0$ with probability one and $(\bar{f}_t(\theta_t))_{t\geqslant 1}$ converges almost surely to $g^\star$.

A.1.2.3 *Almost sure convergence of* $\bar{f}(\theta_t)$

Lemma B.7 of (Mairal, 2013b), based on empirical process theory (Van der Vaart, 2000), ensures that $\bar{f}_t$ uniformly converges to $\bar{f}$. Therefore, $(\bar{f}(\theta_t))_{t\geqslant 1}$ converges almost surely to $g^\star$.

A.1.2.4 *Asymptotic stationary point condition*

Preliminary to the final result, we establish the asymptotic stationary point condition (A.15) as in Mairal (2013b). This requires to adapt the original proof to take into account the errors in surrogate computation and minimization. We set $\alpha > 0$. By definition, $\nabla\bar{h}_t$ is L-Lipschitz over $\mathbb{R}^K$. Following the same computation as in (A.2), we obtain, for all $\alpha > 0$,

$$\|\nabla\bar{h}_t(\theta_t)\|_2 \leqslant \frac{2}{\alpha}\bar{\epsilon}_t + \frac{L\alpha}{2}, \tag{A.14}$$

where we use $|\bar{h}_t(\theta)| \leqslant \bar{\varepsilon}_t$ for all $\theta \in \mathbb{R}^K$. As $\bar{\varepsilon}_t \to 0$ and the inequality (A.14) is true for all α, $\|\nabla \bar{h}_t(\theta_t)\|_2 \to_\infty 0$ almost surely. From the strong convexity of $\bar{g}_t$ and Lemma A.3, $\|\theta_t - \theta_t^\star\|_2$ converges to zero, which ensures

$$\|\nabla \bar{h}_t(\theta_t^\star)\|_2 \leqslant \|\bar{\nabla} h_t(\theta_t)\|_2 + L\|\theta_t - \theta_t^\star\|_2 \to_\infty 0. \tag{A.15}$$

A.1.2.5 *Parametrized surrogates*

We use assumption **(F)** to finally prove the property, adapting the proof of Proposition 3.4 in Mairal (2013b). We first recall the derivations for obtaining (A.16) We define $(\kappa_t)_t$ such that $\bar{g}_t = g_{\kappa_t}$ for all $t > 0$. We assume that θ_∞ is a limit point of $(\theta_t)_t$. As Θ is compact, there exists an increasing sequence $(t_k)_k$ such that $(\theta_{t_k})_k$ converges toward θ_∞. As $\mathcal{K}$ is compact, a converging subsequence of $(\kappa_{t_k})_k$ can be extracted, that converges towards $\kappa_\infty \in \mathcal{K}$. From the sake of simplicity, we drop subindices and assume without loss of generality that $\theta_t \to \theta_\infty$ and $\kappa_t \to \kappa_\infty$. From the compact parametrization assumption, we easily show that $(\bar{g}_{\kappa_t})_t$ uniformly converges towards $\bar{g}_\infty \triangleq \bar{g}_{\kappa_\infty}$. Then, defining $\bar{h}_\infty = \bar{g}_\infty - \bar{f}$, for all $\theta \in \Theta$,

$$\nabla \bar{f}(\theta_\infty, \theta - \theta_\infty) = \nabla \bar{g}_\infty(\theta_\infty, \theta - \theta_\infty) - \nabla \bar{h}_\infty(\theta_\infty, \theta - \theta_\infty) \tag{A.16}$$

We first show that $\nabla \bar{f}(\theta_\infty, \theta - \theta_\infty) \geqslant 0$ for all $\theta \in \Theta$. We consider the sequence $(\theta_t^\star)_t$. From Lemma A.3, $\|\theta_t - \theta_t^\star\|_2 \to 0$, which implies $\theta_t^\star \to \theta_\infty$. $\bar{g}_t$ converges uniformly towards $\bar{g}_\infty$, which implies $(\bar{g}_t(\theta_t^\star))_t \to \bar{g}_\infty(\theta_\infty)$. Furthermore, as $\theta_t^\star$ minimizes $\bar{g}_t$, for all $t > 0$ and $\theta \in \Theta$, $\bar{g}_t(\theta_t^\star) \leqslant \bar{g}_t(\theta)$. This implies $\bar{g}_\infty(\theta_\infty) \leqslant \inf_{\theta \in \Theta} \bar{g}_\infty(\theta)$ by taking the limit for $t \to \infty$. Therefore θ_∞ is the minimizer of $\bar{g}_\infty$ and thus $\nabla \bar{g}_\infty(\theta_\infty, \theta - \theta_\infty) \geqslant 0$.

Adapting the work of Mairal (2013b), we perform the first-order expansion of $\bar{h}_t$ around $\theta_t^\star$ (instead of θ_t in the original proof) and show that $\nabla \bar{h}_\infty(\theta_\infty, \theta - \theta_\infty) = 0$, as $\bar{h}_t$ differentiable, $\|\nabla \bar{h}_t(\theta_t^\star)\|_2 \to 0$ and $\theta_t^\star \to \theta_\infty$. This is sufficient to conclude. □

A.1.3 *Convergence of SOMF — Proof of Proposition 5.1*

Proof of Proposition 5.1. From assumption **(D)**, $(x_t)_t$ is ℓ_2-bounded by a constant X. With assumption **(A)**, it implies that $(\alpha_t)_t$ is ℓ_2-bounded by a constant A. This is enough to show that $(g_t)_t$ and $(\theta_t)_t$ meet basic assumptions **(C)**–**(F)**. Assumption **(G)** immediately implies **(B)**. It remains to show that $(g_t)_t$ and $(\theta_t)_t$ meet the assumptions **(H)** and **(I)**. This will allow to cast SOMF as an instance of SAMM and conclude.

A.1.3.1 *The computation of $\mathbf{D}_t$ verifies (I)*

We define $\mathbf{D}_t^\star = \operatorname{argmin}_{\mathbf{D} \in \mathcal{C}} \bar{g}_t(\mathbf{D})$. We show that performing subsampled block coordinate descent on $\bar{g}_t$ is sufficient to meet assumption **(I)**, where $\theta_t = \mathbf{D}_t$. We separately analyse the exceptional case where no subsampling is done and the general case.

First, with small but non-zero probability, $\mathbf{M}_t = \mathbf{I}_p$ and Alg. 3 performs a single pass of simple block coordinate descent on $\bar{g}_t$. In

this case, as $\bar{g}_t$ is strongly convex from **(A)**, (Beck and Tetruashvili, 2013; Wright, 2015) ensures that the sub-optimality decreases at least of factor $1-\mu$ with a single pass of block coordinate descent, where $\mu > 0$ is a constant independent of t. We provide an explicit μ in Section A.1.4.1.

In the general case, the function value decreases deterministically at each minimization step: $\bar{g}_t(\mathbf{D}_t) \leqslant \bar{g}_t(\mathbf{D}_{t-1})$. As a consequence, $\mathbb{E}[\bar{g}_t(\mathbf{D}_t)|\mathcal{F}_{t-\frac{1}{2}}, \mathbf{M}_t \neq \mathbf{I}_p] \leqslant \bar{g}_t(\mathbf{D}_{t-1})$. Furthermore, $\bar{g}_t$ and hence $\bar{g}_t(\mathbf{D}_t^\star)$ are deterministic with respect to $\mathcal{F}_{t-\frac{1}{2}}$, which implies

$$\mathbb{E}[\bar{g}_t(\mathbf{D}_t^\star)|\mathcal{F}_{t-\frac{1}{2}}, \mathbf{M}_t \neq \mathbf{I}_p] = \bar{g}_t(\mathbf{D}_t^\star).$$

Defining $d \triangleq \mathbb{P}[\mathbf{M}_t = \mathbf{I}_p]$, we split the sub-optimality expectation and combine the analysis of both cases:

$$\begin{aligned}
&\mathbb{E}[\bar{g}_t(\mathbf{D}_t) - \bar{g}_t(\mathbf{D}_t^\star)|\mathcal{F}_{t-\frac{1}{2}}] \\
&= d\mathbb{E}[\bar{g}_t(\mathbf{D}_t) - \bar{g}_t(\mathbf{D}_t^\star)|\mathcal{F}_{t-\frac{1}{2}}, \mathbf{M}_t = \mathbf{I}_p] \\
&\quad + (1-d)\mathbb{E}[\bar{g}_t(\mathbf{D}_t) - \bar{g}_t(\mathbf{D}_t^\star)|\mathcal{F}_{t-\frac{1}{2}}, \mathbf{M}_t \neq \mathbf{I}_p] \\
&\leqslant \big(d(1-\mu) + (1-d)\big)(\bar{g}_t(\mathbf{D}_{t-1}) - \bar{g}_t(\mathbf{D}_t^\star)) \\
&= \big(1 - d\mu\big)(\bar{g}_t(\mathbf{D}_{t-1}) - \bar{g}_t(\mathbf{D}_t^\star)).
\end{aligned}$$

A.1.3.2 *The surrogates $(g_t)_t$ verify* **(H)**

We define $g_t^\star \in \mathcal{S}_{\rho,L}(f_t, \mathbf{D}_{t-1})$ the surrogate used in OMF at iteration t, which depends on the *exact* computation of $\boldsymbol{\alpha}_t^\star$, while the surrogate g_t used in SOMF relies on approximated $\boldsymbol{\alpha}_t$. Formally, using the loss function $\ell(\boldsymbol{\alpha}, \mathbf{G}, \boldsymbol{\beta}) \triangleq \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{G}\boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \boldsymbol{\beta} + \lambda\Omega(\boldsymbol{\alpha})$, we recall the definitions

$$\begin{aligned}
&\boldsymbol{\alpha}_t^\star \triangleq \operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^k} \ell(\boldsymbol{\alpha}, \mathbf{G}_t^\star, \boldsymbol{\beta}_t^\star),\ \boldsymbol{\alpha}_t \triangleq \operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^k} \ell(\boldsymbol{\alpha}, \mathbf{G}_t, \boldsymbol{\beta}_t), \\
&g_t^\star(\mathbf{D}) \triangleq \ell(\boldsymbol{\alpha}_t^\star, \mathbf{D}^\top\mathbf{D}, \mathbf{D}^\top\mathbf{x}_t),\ g_t(\mathbf{D}) \triangleq \ell(\boldsymbol{\alpha}_t, \mathbf{D}^\top\mathbf{D}, \mathbf{D}^\top\mathbf{x}_t).
\end{aligned}$$

The matrices $\mathbf{G}_t^\star$, $\boldsymbol{\beta}_t^\star$ are defined in (4.12) and $\mathbf{G}_t$, $\boldsymbol{\beta}_t$ in either the update rules (b) or (c). We define $\epsilon_t \triangleq \|g_t^\star - g_t\|_\infty$ to be the ℓ_∞ difference between the approximate surrogate of SOMF and the exact surrogate of OMF, as illustrated in Figure 5.1. By definition, $g_t \in \mathcal{T}_{\rho,L}(f_t, \theta_{t-1}, \epsilon_t)$. We first show that ϵ_t can be bounded by the Froebenius distance between the approximate parameters $\mathbf{G}_t$, $\boldsymbol{\beta}_t$ and the exact parameters $\mathbf{G}_t^\star, \boldsymbol{\beta}_t^\star$. Using Cauchy-Schwartz inequality, we first show that there exists a constant $C' > 0$ such that for all $\mathbf{D} \in \mathcal{C}$,

$$|g_t(\mathbf{D}) - g_t^\star(\mathbf{D})| \leqslant C'\|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_t^*\|_2. \tag{A.17}$$

Then, we show that the distance $\|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_t^*\|_2$ can itself be bounded: there exists $C'' > 0$ constant such that

$$\|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_t^\star\|_2 \leqslant C''(\|\mathbf{G}_t^\star - \mathbf{G}_t\|_F + \|\boldsymbol{\beta}_t^\star - \boldsymbol{\beta}_t\|_2). \tag{A.18}$$

We combine both equations and take the supremum over $\mathbf{D} \in \mathcal{C}$, yielding

$$\epsilon_t \leqslant C(\|\mathbf{G}_t^\star - \mathbf{G}_t\|_F + \|\boldsymbol{\beta}_t^\star - \boldsymbol{\beta}_t\|_2), \tag{A.19}$$

where C is constant. Detailed derivation of (A.17) and (A.18) relies on assumption **(A)** and are reported in Section A.1.4.2.

In a second step, we show that $\|\mathbf{G}_t^\star - \mathbf{G}_t\|_F$ and $\|\boldsymbol{\beta}_t^\star - \boldsymbol{\beta}_t\|_2$ vanish almost surely, sufficiently fast. We focus on bounding $\|\boldsymbol{\beta}_t - \boldsymbol{\beta}_t^\star\|_2$ and proceed similarly for $\|\mathbf{G}_t - \mathbf{G}_t^\star\|_F$ when the update rules (b) are used. For $t > 0$, we write $i \triangleq i_t$. Then

$$\boldsymbol{\beta}_t \triangleq \boldsymbol{\beta}_t^{(i)} = \sum_{s \leqslant t, \boldsymbol{x}_s = \boldsymbol{x}^{(i)}} \gamma_{s,t}^{(i)} \mathbf{D}_{s-1}^\top \mathbf{M}_s \boldsymbol{x}^{(i)},$$

where $\gamma_{s,t}^{(i)} = \gamma_{c_t^{(i)}} \prod_{s<t, \boldsymbol{x}_s = \boldsymbol{x}^{(i)}} (1 - \gamma_{c_s^{(i)}})$ and $c_t^{(i)} = \left|\left\{s \leqslant t, \boldsymbol{x}_s = \boldsymbol{x}^{(i)}\right\}\right|$. We can then decompose $\boldsymbol{\beta}_t - \boldsymbol{\beta}_t^\star$ as

$$\begin{aligned}\boldsymbol{\beta}_t - \boldsymbol{\beta}_t^\star = &\sum_{s \leqslant t, \boldsymbol{x}_s = \boldsymbol{x}_t = \boldsymbol{x}^{(i)}} \gamma_{s,t}^{(i)} (\mathbf{D}_{s-1} - \mathbf{D}_{t-1})^\top \mathbf{M}_s \boldsymbol{x}^{(i)} \\ &+ \mathbf{D}_{t-1}^\top \Big(\sum_{s \leqslant t, \boldsymbol{x}_s = \boldsymbol{x}^{i)}} \gamma_{s,t}^{(i)} \mathbf{M}_s - \mathbf{I} \Big) \boldsymbol{x}^{(i)}. \end{aligned} \tag{A.20}$$

The latter equation is composed of two terms: the first one captures the approximation made by using old dictionaries in the computation of $(\boldsymbol{\beta}_t)_t$, while the second captures how the masking effect is averaged out as the number of epochs increases. Assumption **(B)** allows to bound both terms at the same time. Setting $\eta \triangleq \frac{1}{2} \min\left(\nu - \frac{3}{4}, (3u - 2) - \nu\right) > 0$, a tedious but elementary derivation presented in Section A.1.4.3 indeed shows $\mathbb{E}[\|\boldsymbol{\beta}_t - \boldsymbol{\beta}_t^\star\|_2] \in \mathcal{O}(t^{2(u-1)-\eta})$ and $\epsilon_t \to 0$ almost surely. The SOMF algorithm therefore meets assumption **(H)** and is a convergent SAMM algorithm. Proposition 5.1 follows. □

We postponed the proof of three highly technical results in the proof above. We turn to establish them.

A.1.4 *Detailed derivations in the proof of Proposition 5.1*

Let us first exhibit a scaler $\mu > 0$ independent of t,for which **(I)** is met.

A.1.4.1 *Geometric rate for single pass subsampled block coordinate descent*

For $\mathbf{D}^{(j)} \in \mathbb{R}^{p \times k}$ any matrix with non-zero j-th column $\mathbf{d}^{(j)}$ and zero elsewhere

$$\nabla \bar{g}_t(\mathbf{D} + \mathbf{D}^{(j)}) - \nabla \bar{g}_t(\mathbf{D}) = \bar{\mathbf{C}}_t[j, j] \mathbf{d}^{(j)}$$

and hence $\bar{g}_t$ gradient has *component Lipschitz constant* $L_j = \bar{\mathbf{C}}_t[j, j]$ for component j, as already noted by Mairal et al. (2010). Using the terminology from Wright (2015), $\nabla \bar{g}_t$ has *coordinate Lipschitz constant*

$$L_{max} \triangleq \max_{0 \leqslant j < k} \bar{\mathbf{C}}_t[j, j] \leqslant \max_{t > 0, 0 \leqslant j < k} \boldsymbol{\alpha}_t[j]^2 \leqslant A^2,$$

as $(\boldsymbol{\alpha}_t)_t$ is bounded from **(A)**. As a consequence, $\bar{g}_t$ gradient is also L-Lipschitz continuous, where Wright, 2015 note that $L \leqslant \sqrt{k} L_{max}$. Moreover, $\bar{g}_t$ is strongly convex with strong convexity modulus $\rho > 0$

by hypothesis **(A)**. Then, Beck and Tetruashvili, 2013 ensures that after one cycle over the k blocks

$$\begin{aligned}
&\mathbb{E}[\bar{g}_t(\mathbf{D}_t) - \bar{g}_t(\mathbf{D}_t^\star)|\mathcal{F}_{t-1}, \mathbf{M}_t = \mathbf{I}_p] \\
&\leqslant \big(1 - \frac{\rho}{2L_{\max}(1+kL^2/L_{\max}^2)}\big)(\bar{g}_t(\mathbf{D}_{t-1}) - \bar{g}_t(\mathbf{D}_t^\star)) \\
&\leqslant \big(1-\mu\big)(\bar{g}_t(\mathbf{D}_{t-1}) - \bar{g}_t(\mathbf{D}_t^\star)) \quad \text{where} \quad \mu \triangleq \frac{\rho}{2A^2(1+k^2)}
\end{aligned}$$

A.1.4.2 *Controling* ϵ_t *from* $(\mathbf{G}_t, \boldsymbol{\beta}_t), (\mathbf{G}_t^\star, \boldsymbol{\beta}_t^\star)$ — *Equations* (A.17) *and* (A.18)

We detail the derivations that are required to show that **(H)** is met in the proof of SOMF convergence. We first show that $(\boldsymbol{\alpha}_t)_t$ is bounded. We choose $D > 0$ such that $\|\mathbf{d}^{(j)}\|_2 \leqslant D$ for all $j \in [k]$ and $\mathbf{D} \in \mathcal{C}$, and X such that $\|\mathbf{x}\|_2 \leqslant X$ for all $\mathbf{x} \in \mathcal{X}$. From assumption **(A)**, using the second-order growth condition, for all $t > 0$,

$$\begin{aligned}
\frac{\rho}{2}\|\boldsymbol{\alpha}_t - 0\|_2^2 &\leqslant \lambda\Omega(0) - (\frac{1}{2}\boldsymbol{\alpha}_t^\top \mathbf{G}_t \boldsymbol{\alpha}_t - \boldsymbol{\alpha}_t^\top \boldsymbol{\beta}_t + \lambda\Omega(\boldsymbol{\alpha}_t) \\
\frac{\rho}{2}\|\boldsymbol{\alpha}_t\|_2^2 + \frac{1}{2}\boldsymbol{\alpha}_t^\top \mathbf{G}_t \boldsymbol{\alpha}_t &\leqslant 0 + \|\boldsymbol{\alpha}_t\|_2\|\boldsymbol{\beta}_t\|_2, \quad \text{hence} \\
\rho\|\boldsymbol{\alpha}_t\|_2^2 &\leqslant \sqrt{k}rDX\|\boldsymbol{\alpha}_t\|_2, \quad \text{and therefore} \\
\|\boldsymbol{\alpha}_t\|_2 &\leqslant \frac{\sqrt{k}rDX}{\rho} \triangleq A.
\end{aligned}$$

We have successively used the fact that $\Omega(0) = 0$, $\Omega(\boldsymbol{\alpha}_t) \geqslant 0$, and $\|\boldsymbol{\beta}_t\|_2 \leqslant \sqrt{k}rDX$, which can be shown by a simple induction on the number of epochs. For all $t > 0$, from the definition of $\boldsymbol{\alpha}_t$ and $\boldsymbol{\alpha}_t^\star$, for all $\mathbf{D} \in \mathcal{C}$:

$$\begin{aligned}
|g_t(\mathbf{D}) - g_t^\star(\mathbf{D})| &= \Big|\frac{1}{2}\mathrm{Tr}\,\mathbf{D}^\top\mathbf{D}(\boldsymbol{\alpha}_t\boldsymbol{\alpha}_t^\top - \boldsymbol{\alpha}_t^\star\boldsymbol{\alpha}_t^{\star\top}) - (\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_t^\star)^\top\mathbf{D}^\top\mathbf{x}_t\Big| \\
&\leqslant \frac{1}{2}\|\mathbf{D}^\top\mathbf{D}\|_F\|\boldsymbol{\alpha}_t\boldsymbol{\alpha}_t^\top - \boldsymbol{\alpha}_t^\star\boldsymbol{\alpha}_t^{\star\top}\|_F \\
&\quad + \|\mathbf{D}\|_F\|\mathbf{x}_t\|_2\|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_t^\star\|_2 \\
&\leqslant (kD^2A + \sqrt{k}DX)\|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_t^\star\|_2,
\end{aligned}$$

where we use Cauchy-Schwartz inequality and elementary bounds on the Froebenius norm for the first inequality, and use $\boldsymbol{\alpha}_t, \boldsymbol{\alpha}_t^\star \leqslant A$, $\mathbf{x}_t \leqslant X$ for all $t > 0$ and $\mathbf{d}^{(j)} \leqslant D$ for all $j \in [k]$ to obtain the second inequality, which is (A.17) in the main text.

We now turn to control $\|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_t^\star\|_2$. We adapt the proof of Lemma B.6 from Mairal, 2013a, that states the lipschitz continuity of the minimizers of some parametrized functions. By definition,

$$\boldsymbol{\alpha}_t^\star = \operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^k} \ell(\boldsymbol{\alpha}, \mathbf{G}_t^\star, \boldsymbol{\beta}_t^\star) \qquad \boldsymbol{\alpha}_t = \operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^k} \ell(\boldsymbol{\alpha}, \mathbf{G}_t, \boldsymbol{\beta}_t),$$

Assumption **(A)** ensures that $\mathbf{G}_t \succ \rho \mathbf{I}_k$, therefore we can write the second-order growth condition

$$\begin{aligned}
\frac{\rho}{2}\|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_t^\star\|_2^2 &\leqslant \ell(\boldsymbol{\alpha}_t, \mathbf{G}_t^\star, \boldsymbol{\beta}_t^\star) - \ell(\boldsymbol{\alpha}_t, \mathbf{G}_t, \boldsymbol{\beta}_t) \\
\frac{\rho}{2}\|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_t^\star\|_2^2 &\leqslant \ell(\boldsymbol{\alpha}_t^\star, \mathbf{G}_t, \boldsymbol{\beta}_t) - \ell(\boldsymbol{\alpha}_t^\star, \mathbf{G}_t^\star, \boldsymbol{\beta}_t^\star), \quad \text{and therefore} \\
\rho\|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_t^\star\|_2^2 &\leqslant p(\boldsymbol{\alpha}_t) - p(\boldsymbol{\alpha}_t^\star), \quad \text{where} \\
p(\boldsymbol{\alpha}) &\triangleq \ell(\boldsymbol{\alpha}, \mathbf{G}_t, \boldsymbol{\beta}_t) - \ell(\boldsymbol{\alpha}_t, \mathbf{G}_t^\star, \boldsymbol{\beta}_t^\star).
\end{aligned}$$

p takes a simple form and can differentiated with respect to $\boldsymbol{\alpha}$. For all $\boldsymbol{\alpha} \in \mathbb{R}^k$ such that $\|\boldsymbol{\alpha}\|_2 \leqslant A$,

$$\begin{aligned}
p(\boldsymbol{\alpha}) &= \frac{1}{2}\boldsymbol{\alpha}^\top(\mathbf{G}_t - \mathbf{G}_t^\star)\boldsymbol{\alpha} - \boldsymbol{\alpha}^\top(\boldsymbol{\beta}_t - \boldsymbol{\beta}_t^\star) \\
\nabla p(\boldsymbol{\alpha}) &= (\mathbf{G}_t - \mathbf{G}_t^\star)\boldsymbol{\alpha} - (\boldsymbol{\beta}_t - \boldsymbol{\beta}_t^\star) \\
\|\nabla p(\boldsymbol{\alpha})\|_2 &\leqslant A\|\mathbf{G}_t - \mathbf{G}_t^\star\|_F + \|\boldsymbol{\beta}_t - \boldsymbol{\beta}_t^\star\|_2 \triangleq L
\end{aligned}$$

Therefore p is L-Lipschitz on the ball of size A where $\boldsymbol{\alpha}_t$ and $\boldsymbol{\alpha}_t^\star$ live, and

$$\begin{aligned}
\rho\|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_t^\star\|_2^2 &\leqslant L\|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_t^\star\|_2 \\
\|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_t^\star\|_2 &\leqslant \frac{A}{\rho}\|\mathbf{G}_t - \mathbf{G}_t^\star\|_F + \frac{1}{\rho}\|\boldsymbol{\beta}_t - \boldsymbol{\beta}_t^\star\|_2,
\end{aligned}$$

which is (A.18) in the main text. The bound (A.19) on ϵ_t immediately follows.

A.1.4.3 *Bounding $\|\boldsymbol{\beta}_t - \boldsymbol{\beta}_t^\star\|_2$ in equation (A.20)*

Taking the ℓ_2 norm in (A.20), we have $\|\boldsymbol{\beta}_t - \boldsymbol{\beta}_t^\star\|_2 \leqslant BL_t + CR_t$, where B and C are positive constants independent of t and we introduce the terms

$$\begin{aligned}
L_t &\triangleq \sum_{s \leqslant t, \mathbf{x}_s = \mathbf{x}_t = \mathbf{x}^{(i)}} \gamma_{s,t}^{(i)} \|\mathbf{D}_{s-1} - \mathbf{D}_{t-1}\|_F, \\
R_t &\triangleq \left\| \left(\textstyle\sum_{s \leqslant t, \mathbf{x}_s = \mathbf{x}^{(i)}} \gamma_{s,t}^{(i)} \mathbf{M}_s \right) - \mathbf{I} \right\|_F.
\end{aligned}$$

CONDITIONING ON THE SEQUENCE OF DRAWN INDICES. We recall that $(i_t)_t$ is the sequence of indices that are used to draw $(\mathbf{x}_t)_t$ from $\{\mathbf{x}^{(i)}\}_i$, namely such that $\mathbf{x}_t = \mathbf{x}^{(i_t)}$. $(i_t)_t$ is a sequence of i.i.d random variables, whose law is uniform in $[1, n]$. For each $i \in [n]$, we define the increasing sequence $(t_b^{(i)})_{b>0}$ that record the iterations at which sample (i) is drawn, *i.e.* such that $i_{t_b} = i$ for all $b > 0$. For $t > 0$, we recall that $c_t^{(i)} > 0$ is the integer that counts the number of time sample (i) has appeared in the algorithm, *i.e.* $c_t^{(i)} = \max\{b > 0, t_b^{(i)} \leqslant t\}$. These notations will help us understanding the behavior of $(L_t)_t$ and $(R_t)_t$.

BOUNDING R_t. The right term R_t takes its value into sequences that are running average of masking matrices. Formally, we have $R_t = \|\bar{\mathbf{M}}_t^{(i_t)} - \mathbf{I}\|_F$, where we define for all $i \in [n]$,

$$\bar{\mathbf{M}}_t^{(i)} \triangleq \sum_{b=1}^{c_t^{(i)}} \gamma^{(i)}_{t_b^{(i)}, t_c^{(i)}} \mathbf{M}_{t_b}, \quad \text{which follows the recursion}$$

$$\begin{cases} \bar{\mathbf{M}}_t^{(i)} &= (1-\gamma_{c_t^{(i)}})\bar{\mathbf{M}}_{t-1}^{(i)} + \gamma_{c_t^{(i)}}\mathbf{M}_t \quad \text{if } i = i_t \\ \bar{\mathbf{M}}_t^{(i)} &= \mathbf{M}_{t-1}^{(i)} \quad \text{if } i \neq i_t \\ \bar{\mathbf{M}}_0^{(i)} &= 0 \quad \text{for all } i \in [n] \end{cases} \tag{A.21}$$

When sampling a sequence of indices $(i_s)_{s>0}$, the n random matrix sequences $[(\bar{\mathbf{M}}_t^{(i)})_{t\leqslant 0}]_{i\in[n]}$ follows the same probability law as the sampling is uniform. We therefore focus on controling $(\bar{\mathbf{M}}_t^{(0)})_t$. For simplicity, we write $c_t \triangleq c_t^{(0)}$. When $\mathbb{E}[\cdot]$ is the expectation over the sequence of indices $(i_s)_s$,

$$\begin{aligned} \mathbb{E}[\|\bar{\mathbf{M}}_t^{(0)} - \mathbf{I}\|_F]^2 &\leqslant \mathbb{E}\big[\sum_{j=1}^{p} (\bar{\mathbf{M}}_t^{(0)}[j,j] - 1)\big] = p\mathbb{E}[(\bar{\mathbf{M}}_t^{(0)}[0,0] - 1)] \\ &\leqslant C\, p(c_t)^{1/2}\gamma_{c_t} = C\, p(c_t)^{1/2-\nu}, \end{aligned} \tag{A.22}$$

where C is a constant independent of tWe have simply bounded the Froebenius norm by the ℓ_1 norm in the first inequality and used the fact that all coefficients $\mathbf{M}_t[j,j]$ follows the same Bernouilli law for all $t > 0$, $j \in [p]$. We then used Lemma B.7 from Mairal, 2013b for the last inequality. This lemma applies as $\mathbf{M}_t[0,0]$ follows the recursion (A.21). It remains to take the expectation of (A.22), over all possible sampling trajectories $(i_s)_{s>0}$:

$$\begin{aligned} \mathbb{E}[R_t] &= \mathbb{E}\big[\mathbb{E}[R_t|(i_s)_s]\big] = \mathbb{E}\big[\mathbb{E}[\|\mathbf{M}_t^{(i_t)} - \mathbf{I}\|_F|(i_s)_s]\big] \\ &= \mathbb{E}\big[\mathbb{E}[\|\mathbf{M}_t^{(0)} - \mathbf{I}\|_F|(i_s)_s]\big] = \mathbb{E}[\|\mathbf{M}_t^{(0)} - \mathbf{I}\|_F] \\ &= Cp\mathbb{E}[(c_t)^{1/2-\nu}] \leqslant Cp\mathbb{E}[(c_t)^{2(u-1)-\eta}]. \end{aligned} \tag{A.23}$$

The last inequality arises from the definition of the exponent $\eta \triangleq \frac{1}{2}\min\big(\nu - \frac{3}{4}, (3u-2) - \nu\big)$, as follows. First, $\eta > 0$ as $u > \frac{11}{12}$. Then, we successively have

$$\frac{5}{2} - 2u < \frac{2}{3} < \frac{3}{4}, \quad \text{as } u > \frac{11}{12}, \qquad \nu \geqslant \frac{3}{4} + 2\eta > \frac{5}{2} - 2u + 2\eta,$$
$$\frac{1}{2} - \nu < \frac{1}{2} - \frac{5}{2} + 2u - 2\eta = 2(u-1) - 2\eta < 2(u-1) - \eta,$$

which allows to conclude. Lemma B.7 from Mairal, 2013b also ensures that $\mathbf{M}_t[0,0] \to 1$ almost surely when $t \to \infty$. Therefore $(\bar{\mathbf{M}}_t^{(0)} - \mathbf{I})_t$ converges towards 0 almost surely, given any sample sequence $(i_s)_s$. It thus converges almost surely when *all* random variables of the algorithm are considered. This is also true for $(\bar{\mathbf{M}}_t^{(i)} - \mathbf{I})_t$ for all $i \in [n]$ and hence for R_t.

BOUNDING L_t. As above, we define n sequences $[(L_t^{(i)})_t]_{i\in[n]}$, such that $L_t = L_t^{(i_t)}$ for all $t > 0$. Namely,

$$\begin{aligned} L_t^{(i)} &\triangleq \sum_{\substack{s\leqslant t,\\ \mathbf{x}_s=\mathbf{x}_t=\mathbf{x}^{(i)}}} \gamma_{s,t}^{(i)} \|\mathbf{D}_{s-1} - \mathbf{D}_{t-1}\|_F \\ &= \sum_{b=1}^{c_t^{(i)}} \gamma^{(i)}_{t_b^{(i)},t^{(i)}_{c_t^{(i)}}} \left\|\mathbf{D}_{t_b-1} - \mathbf{D}_{t_{c_t^{(i)}}-1}\right\|_F. \end{aligned}$$

Once again, the sequences $\left[(L_t^{(i)})_t\right]_i$ all follows the same distribution when sampling over sequence of indices $(i_s)_s$. We thus focus on bounding $(L_t^{(0)})_t$. Once again, we drop the (0) superscripts in the right expression for simplicity. We set $\mu \triangleq 3u - 2 - \eta$. From assumption **(B)** and the definition of η, we have $\nu < \mu < 1$. We split the sum in two parts, around index $d_t \triangleq c_t - \lfloor (c_t)^\mu \rfloor$, where $\lfloor\cdot\rfloor$ takes the integer part of a real number. For simplicity, we write $d \triangleq d_t$ and $c \triangleq c_t$ in the following.

$$\begin{aligned} L_t^{(0)} &= \sum_{b=1}^{c} \gamma_{t_b,t_c} \left\|\mathbf{D}_{t_b-1} - \mathbf{D}_{t_c-1}\right\|_F \\ &\leqslant 2\sqrt{k}D \sum_{b=1}^{d} \gamma_{t_b,t_c} + \sum_{b=d+1}^{c} \gamma_{t_b,t} \sum_{s=t_b-1}^{t_c-1} w_s \\ &\triangleq 2\sqrt{k}D L_{t,1}^{(0)} + L_{t,2}^{(0)} \end{aligned}$$

On the left side, we have bounded $\|\mathbf{D}_t\|_F$ by $\sqrt{k}D$, where D is defined in the previous section. The right part uses the bound on $\|\mathbf{D}_s - \mathbf{D}_t\|_F$ provided by Lemma A.3, that applies here as **(I)** is met and (A.19) ensures that $(\|g_t - g_t^\star\|_\infty)_t$ is bounded.

We now study both $L_{t,1}^{(0)}$ and $L_{t,2}^{(0)}$. First, for all $t > 0$,

$$\begin{aligned} L_{t,1}^{(0)} &\triangleq \sum_{b=1}^{d} \gamma_{t_b,t_c} = \sum_{b=1}^{d} \gamma_b \prod_{p=b+1}^{c} (1-\gamma_p) \leqslant \sum_{b=1}^{d} \gamma_b (1-\gamma_c)^{c-b} \\ &\leqslant \frac{(1-\gamma_c)^{\lfloor c^\mu \rfloor}}{\gamma_c} \leqslant c^\nu \exp\left(\log(1-\frac{1}{c^\nu})c^\mu\right) \\ &\leqslant C' c^\nu \exp(c^{\mu-\nu}) \leqslant C c^{2(u-1)-\eta} = C(c_t)^{2(u-1)-\eta}, \end{aligned}$$

where C and C' are constants independent of t. We have used $\mu > \nu$ for the third inequality, which ensures that $\log(1-\frac{1}{c^\nu})c^\mu \in \mathcal{O}(c^{\mu-\nu})$. Basic asymptotic comparison provides the last inequality, as $c_t \to \infty$ almost surely and the right term decays exponentially in $(c_t)_t$, while the left decays polynomially. As a consequence, $L_{t,1}^{(0)} \to 0$ almost surely.

Secondly, the right term can be bounded as $(w_t)_t$ decays sufficiently rapidly. Indeed, as $\sum_{b=1}^{c} \gamma_{t_b,t} = 1$, we have

$$\begin{aligned} L_{t,2}^{(0)} &\triangleq \sum_{b=d}^{c} \gamma_{t_b,t} \sum_{s=t_b-1}^{t_c-1} w_s \leqslant \max_{d\leqslant b\leqslant c} \Big(\sum_{s=t_b-1}^{t_c-1} w_s \Big) = \sum_{s=t_d-1}^{t_c-1} w_s \\ &\leqslant w_{t_d}(t_c - t_d) = \frac{t_c - t_d}{(t_d)^u} = \frac{c_t - d_t}{(d_t)^u} \frac{t_c - t_d}{c_t - d_t} (\frac{d_t}{t_d})^u \end{aligned}$$

from elementary comparisons. First, we use the definition of μ to draw

$$\frac{c_t - d_t}{(d_t)^u} \leqslant \frac{(c_t)^\mu}{(c_t)^u(1-c_t^{\mu-1})^u} \leqslant C(c_t)^{\mu-u} = C(c_t)^{2(u-1)-\eta},$$

were we use the fast that $\eta - 1 < 0$. We note that for all $b > 0$, $t_{b+1} - t_b$ follows a geometric law of parameter $\frac{1}{n}$, and expectation n. Therefore, as $c - d \to \infty$ when $t \to 0$, from the strong law of large numbers and linearity of the expectation

$$\frac{t_c - t_d}{c-d} = \frac{1}{c-d}\sum_{b=d}^{c-1} t_{b+1} - t_b \to n,$$
$$\frac{t_d}{d} = \frac{1}{d}\sum_{b=0}^{d-1} t_{b+1} - t_b \to n \quad \text{almost surely.}$$

As a consequence, $\frac{t_c - t_d}{c_t - d_t}(\frac{d_t}{t_d})^u \to n^{1-u}$ almost surely. This immediately shows $L_{t,2}^{(0)} \to 0$ and thus $L_t^{(0)} \to 0$ almost surely. As with R_t, this implies that $L_t \to 0$ almost surely and therefore

$$\|\beta_t - \beta_t^\star\|_2 \to 0 \quad \text{almost surely.}$$

Finally, from the dominated convergence theorem, $\mathbb{E}[\frac{t_c - t_d}{c_t - d_t}(\frac{d_t}{t_d})^u] \to n^{1-u}$ for $t \to \infty$. We can use Cauchy-Schartz inequality and write

$$\mathbb{E}[L_{t,2}^{(0)}] = \mathbb{E}[\frac{t_c - t_d}{(t_d)^u}] \leqslant \mathbb{E}[\frac{c_t - d_t}{(d_t)^u}]\mathbb{E}[\frac{t_c - t_d}{c_t - d_t}(\frac{d_t}{t_d})^u]$$
$$\leqslant C'\mathbb{E}[\frac{c_t - d_t}{(d_t)^u}] \leqslant C\,C'\mathbb{E}[(c_t)^{2(u-1)-\eta}],$$

where C' is a constant independent of t. Then

$$\mathbb{E}[L_t] = \mathbb{E}\left[\mathbb{E}[L_t^{(i_t)}|(i_s)_s]\right] = \mathbb{E}\left[\mathbb{E}[L_t^{(0)}|(i_s)_s]\right]$$
$$= \mathbb{E}[L_t^{(0)}] \leqslant 2\sqrt{k}D\mathbb{E}[L_{t,1}^{(0)}] + \mathbb{E}[L_{t,2}^{(0)}] \in \mathcal{O}((c_t)^{2(u-1)-\eta}).$$

Combined with (A.23), this shows that

$$\mathbb{E}[\|\beta_t - \beta_t^\star\|_2] \in \mathcal{O}((c_t)^{2(u-1)-\eta}).$$

As c_t follows a binomial distribution of parameter $(t, \frac{1}{n})$, $\frac{c_t}{t} \to \frac{1}{n}$ almost surely when $t \to 0$. Therefore $\mathbb{E}[(\frac{c_t}{t})^{2(u-1)-\eta})] \to n^{\eta-2(u-1)}$, and from Cauchy-Schwartz inequality,

$$\mathbb{E}[\|\beta_t - \beta_t^\star\|_2] \leqslant C\mathbb{E}[(\frac{c_t}{t})^{2(u-1)-\eta})]t^{2(u-1)-\eta} \in \mathcal{O}(t^{2(u-1)-\eta}).$$

We have reused the fact that converging sequences are bounded. This is enough to conclude.

B

PROOFS AND RESULTS FROM CHAPTER 8 — DIFFERENTIABLE DYNAMIC PROGRAMMING

B.1 PROOFS AND DETAILED DERIVATIONS

This section contains the proofs of the propositions and lemmas presented in the main text. It also contains derivations of gradient, directional derivative and Hessian-product computations.

B.1.1 *Proof of Lemma 8.1 (properties of* $\max_\Omega$*)*

PROPERTY 1 (BOUNDEDNESS). Let $\mathbf{q}^\star$ and $\mathbf{q}^\star_\Omega$ be the solutions of $\max_{\mathbf{q}\in\triangle^D} \mathbf{q}^\top \mathbf{x}$ and $\max_{\mathbf{q}\in\triangle^D} \mathbf{q}^\top \mathbf{x} - \Omega(\mathbf{q})$, respectively. Then, we have

$$\max_\Omega(\mathbf{x}) = \langle \mathbf{q}^\star_\Omega, \mathbf{x}\rangle - \Omega(\mathbf{q}^\star_\Omega) \geqslant \langle \mathbf{q}^\star, \mathbf{x}\rangle - \Omega(\mathbf{q}^\star) = \max(\mathbf{x}) - \Omega(\mathbf{q}^\star)$$

and

$$\max(\mathbf{x}) - \Omega(\mathbf{q}^\star_\Omega) \geqslant \langle \mathbf{q}^\star_\Omega, \mathbf{x}\rangle - \Omega(\mathbf{q}^\star_\Omega) = \max_\Omega(\mathbf{x}).$$

Combining the two and using $L_{\Omega,D} \leqslant \Omega(\mathbf{q}) \leqslant U_{\Omega,D}\ \forall \mathbf{q} \in \triangle^D$, we obtain

$$\max(\mathbf{x}) - U_{\Omega,D} \leqslant \max(\mathbf{x}) - \Omega(\mathbf{q}^\star) \leqslant \max_\Omega(\mathbf{x}) \leqslant \max(\mathbf{x}) - \Omega(\mathbf{q}^\star_\Omega)$$

$$\leqslant \max(\mathbf{x}) - L_{\Omega,D}.$$

When $\Omega(\mathbf{q}) = \sum_i q_i \log q_i$, we have the tight inequality $-\log D \leqslant \Omega(\mathbf{q}) \leqslant 0\ \forall \mathbf{q} \in \triangle^D$ and hence

$$\max(\mathbf{x}) \leqslant \max_\Omega(\mathbf{x}) \leqslant \max(\mathbf{x}) + \log D.$$

When $\Omega(\mathbf{q}) = \frac{1}{2}\|\mathbf{q}\|^2$, we have the tight inequality $\frac{1}{2D} \leqslant \Omega(\mathbf{q}) \leqslant \frac{1}{2}\ \forall \mathbf{q} \in \triangle^D$ and hence

$$\max(\mathbf{x}) - \frac{1}{2} \leqslant \max_\Omega(\mathbf{x}) \leqslant \max(\mathbf{x}) - \tfrac{1}{2D}.$$

Note that the difference $U_{\Omega,D} - L_{\Omega,D}$ is equal to $\log D$ when Ω is the negative entropy and to $\frac{D-1}{2D} \leqslant \frac{1}{2}$ when Ω is the squared ℓ_2 norm. Since $\log D > \frac{1}{2}$ for all integers $D \geqslant 2$, we get a better approximation of the max operator using squared ℓ_2 norm than using negative entropy, whenever $D \geqslant 2$.

PROPERTY 2 (DISTRIBUTIVITY OF $+$ OVER $\max_\Omega$). This follows immediately from

$$\begin{aligned}\max_\Omega(\mathbf{x} + c\mathbf{1}) &= \max_{\mathbf{q}\in\triangle^D} \langle \mathbf{q}, \mathbf{x} + c\mathbf{1}\rangle - \Omega(\mathbf{q}) \\ &= \max_{\mathbf{q}\in\triangle^D} \langle \mathbf{q}, \mathbf{x}\rangle - \Omega(\mathbf{q}) + c = \max_\Omega(\mathbf{x}) + c.\end{aligned}$$

Using our shorthand notation, this simply becomes

$$\max_{\mathbf{Y}\in\mathcal{Y}}{}_{\Omega}\ (f(\mathbf{Y})+c) = \left(\max_{\mathbf{Y}\in\mathcal{Y}}{}_{\Omega}\ f(\mathbf{Y})\right)+c.$$

PROPERTY 3 (COMMUTATIVITY). Assume $\Omega(\mathbf{P}\mathbf{q}) = \Omega(\mathbf{q})$ for all permutation matrices $\mathbf{P}$. Let $\mathbf{P}^{-1}$ be the inverse permutation matrix associated with $\mathbf{P}$. Then we have

$$\begin{aligned}\max_{\Omega}(\mathbf{P}\mathbf{x}) &= \max_{\mathbf{q}\in\triangle^D}\langle\mathbf{q},\mathbf{P}\mathbf{x}\rangle-\Omega(\mathbf{q}) = \max_{\mathbf{q}\in\triangle^D}\langle\mathbf{P}^{-1}\mathbf{q},\mathbf{x}\rangle-\Omega(\mathbf{q})\\ &= \max_{\mathbf{q}\in\triangle^D}\langle\mathbf{q},\mathbf{x}\rangle-\Omega(\mathbf{P}\mathbf{q}) = \max_{\mathbf{q}\in\triangle^D}\langle\mathbf{q},\mathbf{x}\rangle-\Omega(\mathbf{q}).\end{aligned}$$

PROPERTY 4 (NON-DECREASINGNESS IN EACH COORDINATE). If $\mathbf{x}\leqslant\mathbf{y}$, then for all $\mathbf{q}\in\triangle^D$, $\langle\mathbf{x},\mathbf{q}\rangle-\Omega(\mathbf{q})\leqslant\langle\mathbf{y},\mathbf{q}\rangle-\Omega(\mathbf{q})$, as all $\mathbf{q}$ coordinates are non-negative. Thus $\max_\Omega(\mathbf{x})\leqslant\max_\Omega(\mathbf{y})$.

PROPERTY 5 (INSENSITIVITY TO $-\infty$). As we have

$$\max_{\Omega}(\mathbf{x}) = \max_{\mathbf{q}\in\triangle^D}\langle\mathbf{q},\mathbf{x}\rangle-\Omega(\mathbf{q}),$$

if $x_j=-\infty$, then $q_j=\nabla\max_\Omega(\mathbf{x})_j=0$ is the only feasible solution for the j^{th} coordinate.

B.1.2 *Proof of Proposition 8.1 (optimality of DP recursion)*

Let $v_i(\theta)$ be the highest-score path up to node $i\in[N]$. Let $\mathcal{Y}_i$ be the set of paths $\mathbf{y}=(y_1,\ldots,y_L)$ starting from node 1 and reaching node i, that is $y_1=1$ and $y_L=i$. Note that L may depend on $\mathbf{y}$ but we do not make this dependency explicit. Because nodes are sorted in topological order, we can compute $v_i(\theta)$ by

$$\begin{aligned}v_i(\theta) = \max_{\mathbf{y}\in\mathcal{Y}_i}\sum_{t=2}^{L}\theta_{y_t,y_{t-1}} &= \max_{\mathbf{y}\in\mathcal{Y}_i}\sum_{t=2}^{L-1}\theta_{y_t,y_{t-1}}+\theta_{y_L,y_{L-1}}\\ &= \max_{\mathbf{y}\in\mathcal{Y}_i}\sum_{t=2}^{L-1}\theta_{y_t,y_{t-1}}+\theta_{i,y_{L-1}}.\end{aligned}$$

Recall that $\mathcal{P}_i$ is the set of parent nodes of node i. From the *associativity* of the max operator,

$$\begin{aligned}v_i(\theta) &= \max_{j\in\mathcal{P}_i}\max_{\substack{\mathbf{y}\in\mathcal{Y}_i\\ y_{L-1}=j}}\left(\sum_{t=2}^{L-1}\theta_{y_t,y_{t-1}}+\theta_{i,y_{L-1}}\right)\\ &= \max_{j\in\mathcal{P}_i}\max_{\substack{\mathbf{y}\in\mathcal{Y}_i\\ y_{L-1}=j}}\left(\sum_{t=2}^{L-1}\theta_{y_t,y_{t-1}}+\theta_{i,j}\right).\end{aligned}$$

From the *distributivity* of $+$ over max, we obtain

$$v_i(\theta) = \max_{j\in\mathcal{P}_i}\left(\max_{\substack{\mathbf{y}\in\mathcal{Y}_i\\ y_{L-1}=j}}\sum_{t=2}^{L-1}\theta_{y_t,y_{t-1}}\right)+\theta_{i,j} = \max_{j\in\mathcal{P}_i}v_j(\theta)+\theta_{i,j},$$

where we used the fact that the inner max operations are independent of $y_L=i$. This concludes the proof of the optimality of (8.3).

B.1.3 *Proof of Proposition 8.2 (properties of $DP_\Omega(\theta)$)*

We prove in this section the three main claims of Proposition 8.2. For the first two claims, we rewrite (8.3) and (8.6) using the following notations:

$$
\begin{aligned}
&\nu_i^0(\theta) \triangleq \max(\mathbf{u}_i^0(\theta)) \quad \text{and} \quad \nu_i^\Omega(\theta) \triangleq \max(\mathbf{u}_i^\Omega(\theta)), \quad \text{where} \\
&\mathbf{u}_i^0(\theta) \triangleq (\theta_{i,1} + \nu_1^0(\theta), \dots, \theta_{i,i-1} + \nu_{i-1}^0(\theta), -\infty, \dots, -\infty) \in \mathbb{R}^N, \\
&\mathbf{u}_i^\Omega(\theta) \triangleq (\theta_{i,1} + \nu_1^\Omega(\theta), \dots, \theta_{i,i-1} + \nu_{i-1}^\Omega(\theta), \underbrace{-\infty}_{i}, \dots, -\infty) \in \mathbb{R}^N.
\end{aligned}
$$

These definitions are indeed valid as per Lemma 8.1, property 5.

PROOF OF $DP_\Omega(\theta)$ CONVEXITY. Since $\nu_1^\Omega(\theta) = 0$, it is trivially convex. Assume that $\nu_2^\Omega(\theta), \dots, \nu_{i-1}^\Omega(\theta)$ are convex. Then, $\nu_i^\Omega(\theta)$ is the composition of $\max_\Omega$ and $\mathbf{u}_i^\Omega$, a convex function and a function which outputs a vector whose each coordinate is convex in θ. By induction, since $\max_\Omega$ is non-decreasing per coordinate (cf. Lemma 8.1 property 4), $\nu_i^\Omega(\theta)$ is convex (*e.g.*, Boyd and Vandenberghe, 2004, Section 3.2.4). Therefore $\nu_i^\Omega(\theta)$ is convex for all $i \in [N]$ and $DP_\Omega(\theta) = \nu_N^\Omega(\theta)$ is convex.

PROOF OF $DP_\Omega(\theta)$ BOUND. We clearly have $\nu_1^\Omega(\theta) \geqslant \nu_1^0(\theta)$. Assume that $\nu_j^\Omega(\theta) \geqslant \nu_j^0(\theta) - (j-1)U_{\Omega,N}$ for all $j \in \{2, \dots, i-1\}$. That is, $\mathbf{u}_i^\Omega(\theta) \geqslant \mathbf{u}_i^0(\theta) - (i-2)U_{\Omega,N}\mathbf{1}$, where $\mathbf{1} \in \mathbb{R}^N$ is the unit vector. Then, by induction, we have

$$
\begin{aligned}
\max_\Omega(\mathbf{u}_i^\Omega(\theta)) &\geqslant \max_\Omega(\mathbf{u}_i^0(\theta)) - (i-2)U_{\Omega,N} \\
&\geqslant \max(\mathbf{u}_i^0(\theta)) - (i-1)U_{\Omega,N},
\end{aligned}
$$

where we used Lemma 8.1, properties 1, 2 and 4. Therefore $\nu_i^\Omega(\theta) \geqslant \nu_i^0(\theta) - (i-1)U_{\Omega,N}$ for all $i \in [N]$ and hence, $DP_\Omega(\theta) \geqslant LP(\theta) - (N-1)U_{\Omega,N}$. Using a similar reasoning we obtain $\nu_i^0(\theta) - (i-1)L_{\Omega,N} \geqslant \nu_i^\Omega(\theta)$ and therefore $LP(\theta) - (N-1)L_{\Omega,N} \geqslant DP_\Omega(\theta)$. To summarize, we obtain

$$
LP(\theta) - (N-1)L_{\Omega,N} \geqslant DP_\Omega(\theta) \geqslant LP(\theta) - (N-1)U_{\Omega,N},
$$

which concludes the proof. Note that using property 1 of Lemma 8.1, this immediately implies a bound involving $LP_\Omega(\theta)$ instead of $LP(\theta)$.

PROOF THAT $\Omega = -\gamma H \Rightarrow DP_\Omega(\theta) = LP_\Omega(\theta)$. We first show that $\max_\Omega$ is associative.

Lemma B.1. *Associativity of* $\max_\Omega$ *when* $\Omega = -\gamma H$
We have $\max_\Omega(\max_\Omega(\mathbf{x}), c) = \max_\Omega(\mathbf{x}, c) \quad \forall \mathbf{x} \in \mathbb{R}^D, c \in \mathbb{R}$.

Proof. We simply use the closed form of $\max_\Omega$ when $\Omega = -\gamma H$ (cf. Section B.2.1):

$$\begin{aligned}
\max_\Omega(\max_\Omega(\mathbf{x}), c) &= \gamma \log(\exp(\max_\Omega(\mathbf{x})/\gamma) + \exp(c/\gamma)) \\
&= \gamma \log\left(\exp\left(\log \sum_{i=1}^{D} \exp(x_i/\gamma)\right) + \exp(c/\gamma)\right) \\
&= \gamma \log\left(\sum_{i=1}^{D} \exp(x_i/\gamma) + \exp(c/\gamma)\right) \\
&= \max_\Omega(\mathbf{x}, c),
\end{aligned}$$

and the lemma follows. □

Using our shorthand notation, Lemma B.1 can be used to write

$$\max_{\Omega\ (y_1,\dots,y_i,\dots,y_L)} f(\mathbf{y}) = \max_{\Omega\ \nu}\ \max_{\Omega\ (y_1,\dots,\nu,\dots,y_L)} f(\mathbf{y}).$$

This is precisely the associative property that we used in the proof of Proposition 8.1. The second property that we used, the distributivity of $+$ over max, holds for any $\max_\Omega$, as per Lemma 8.1 property 2. Thus, the same proof as Proposition 8.1 is also valid when we substitute max with $\max_\Omega$, when $\Omega = -\gamma H$, which yields $\mathrm{LP}_\Omega(\theta) = \mathrm{DP}_\Omega(\theta)$.

PROOF THAT $\Omega = -\gamma H \Leftarrow \mathrm{DP}_\Omega(\theta) = \mathrm{LP}_\Omega(\theta)$. Mirroring the previous proof, we first characterize the regularizations Ω for which $\max_\Omega$ is associative.

Lemma B.2. *Let $\Omega\colon \triangle^D \to \mathbb{R}$ be a regularization function, i.e., $\operatorname{dom}\Omega = \triangle^D$. Assume that there exist ω convex lower-semi-continuous defined on $[0,1]$ such that $\Omega(\mathbf{q}) = \sum_{i=1}^{d} \omega(q_i)$. If*

$$\max_\Omega(\max_\Omega(\mathbf{x}), c) = \max_\Omega(\mathbf{x}, c) \quad \forall \mathbf{x} \in \mathbb{R}^D, c \in \mathbb{R},$$

then $\Omega(\mathbf{q}) = -\gamma \sum_{i=1}^{d} q_i \log(q_i)$ for some $\gamma \geqslant 0$.

Proof. We start by writing the associativity property for three elements. For all $x_1, x_2, x_3 \in \mathbb{R}$,

$$\begin{aligned}
\max_\Omega\big((x_1, x_2, x_3)\big) &= \max_\Omega\big(\max_\Omega(x_1, x_2), x_3)\big) \\
&= \max_{\substack{q+q_3=1 \\ q, q_3 \geqslant 0}} q \max_{\substack{\tilde{q}_1+\tilde{q}_2=1 \\ \tilde{q}_i \geqslant 0}} \big(\tilde{q}_1 x_1 + \tilde{q}_2 x_2 - \omega(\tilde{q}_1) - \omega(\tilde{q}_2)\big) \\
&\qquad + q_3 x_3 - \omega(q_3) - \omega(q) \\
&= \max_{\substack{q_1+q_2+q_3=1 \\ q_i \geqslant 0}} q_1 x_1 + q_2 x_2 + q_3 x_3 - \Phi(q_1, q_2, q_3),
\end{aligned}$$

where we define $\Phi(q_1, q_2, q_3)$ as

$$(q_1 + q_2)\Big(\omega\big(\frac{q_1}{q_1+q_2}\big) + \omega\big(\frac{q_2}{q_1+q_2}\big)\Big) + \omega(q_1+q_2) + \omega(q_3).$$

We have performed a variable change $q_{1,2} = q\,\tilde{q}_{1,2}$ at the second line, and noticed $q = q_1 + q_2$. Therefore

$$\max_\Omega\big((x_1, x_2, x_3)\big) = \Phi^\star(x_1, x_2, x_3),$$

where $\Phi^\star$ is the convex conjugate of Φ restricted to $]0,1]^3$. By definition, we also have $\max_\Omega\big((x_1, x_2, x_3)\big) = \Omega^\star(x_1, x_2, x_3)$, so that $\Omega^\star = \Phi^\star$ on $\mathbb{R}^3$. As Ω is convex and lower semi-continuous, we can apply Moreau-Yoshida theorem and obtain $\Omega^{\star\star} = \Omega = \Phi^{\star\star} \leqslant \Phi$.

Suppose that there exists $\mathbf{q} = (q_1, q_2, q_3) \in \triangle^3$ such that we have $\Phi(q_1, q_2, q_3) < \Omega(q_1, q_2, q_3)$. Given the forms of Φ and Ω, $\Phi(q_1, q_2, 0) < \Omega(q_1, q_2, 0)$. We let $\mathbf{x} = (x_1, x_2, -\infty) \in \mathbb{R}^3$ such that

$$\begin{aligned}
\max_\Omega(x_1, x_2, -\infty) &= \max_\Omega(x_1, x_2) \\
&= x_1 q_1 + x_2 q_2 - \omega(q_1) - \omega(q_2) = \langle \mathbf{x}, \mathbf{q}\rangle - \Omega(\mathbf{q}) \\
&< \langle \mathbf{x}, \mathbf{q}\rangle - \Phi(\mathbf{q}) \leqslant \max_{\mathbf{q}\in\triangle^3} \langle \mathbf{x}, \mathbf{q}\rangle - \Phi(\mathbf{q}) \\
&= \max_\Omega\big(\max_\Omega(x_1, x_2), -\infty\big),
\end{aligned}$$

leading to a contradiction. Therefore $\Omega \geqslant \Phi$ over $\triangle^3$, and finally $\Omega = \Phi$. We have used the fact that the operator $\nabla\max_\Omega : \mathbb{R}^2 \to \triangle^2$ is surjective, as $\triangle^2$ is a one-dimensional segment, $\nabla\max_\Omega$ is continuous and reaches the extreme values $\nabla\max_\Omega(0, -\infty) = (1, 0)$ and $\nabla\max_\Omega(-\infty, 0) = (0, 1)$ — which allows to use the intermediate value theorem.

To conclude, for all $q_1, q_2 \in]0,1]$ such that $q_1 + q_2 \leqslant 1$, we have

$$\begin{aligned}
\omega(q_1) + \omega(q_2) = (q_1 + q_2)\Big(\omega\big(\frac{q_1}{q_1+q_2}\big) + \omega\big(\frac{q_2}{q_1+q_2}\big)\Big) \\
+ \omega(q_1 + q_2),
\end{aligned}$$

and thus, for all $0 < y \leqslant 1$, $0 < x < 1$,

$$\omega(xy) + \omega((1-x)y) - \omega(y) = y(\omega(x) + \omega(1-x)) \tag{B.1}$$

where we have set $y = q_1 + q_2$ and $x = \frac{q_1}{q_1+q_2}$. The functional equation (B.1) was first studied in the field of information theory. As first shown by Horibe (1988, Theorem 0), and further extended (Gselmann, 2011), all measurable solutions have the form

$$\omega(x) = -\gamma x \log(x),$$

where $\gamma \geqslant 0$ is a constant. The lemma follows. □

Assuming that Ω is not equal to $-\gamma H$ for any $\gamma \geqslant 0$, the previous lemma tells us that the associativity property is not met for a triplet $(x_1, x_2, x_3) \in \mathbb{R}^3$. In Figure B.1, we construct a graph G such that

$$\mathrm{DP}_\Omega(\boldsymbol{\theta}) = \max_\Omega(\max_\Omega(x_1, x_2), x_3) \neq \mathrm{LP}_\Omega(\boldsymbol{\theta}) = \max_\Omega(x_1, x_2, x_3)$$

The proposition follows.

B.1.4 *Computation of $\nabla LP_\Omega(\boldsymbol{\theta})$ and interpretation as an expectation*

We show that $\nabla\mathrm{LP}_\Omega(\boldsymbol{\theta}) \in \mathrm{conv}(\mathcal{Y})$, and characterize a path distribution of which $\nabla\mathrm{LP}_\Omega(\boldsymbol{\theta})$ is the expectation.

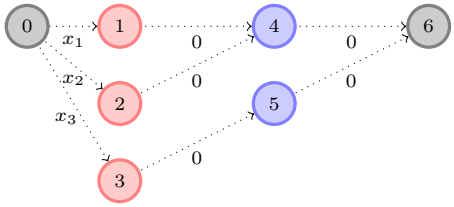


Figure B.1 – In general, $v_6(\theta) = \mathrm{DP}_\Omega(\theta) \neq \mathrm{LP}_\Omega(\theta)$.

CONVEX HULL OF $\mathcal{Y}$. We rewrite $\mathrm{LP}_\Omega(\theta) = \max_\Omega(\mathbf{u}(\theta))$, where $\mathbf{u}(\theta) \triangleq (\langle \mathbf{Y}, \theta \rangle)_{\mathbf{Y} \in \mathcal{Y}}$. Using the chain rule, we have

$$\nabla \mathrm{LP}_\Omega(\theta) = \mathbf{J}_\mathbf{u}(\theta)^\top \nabla \max_\Omega(\mathbf{u}(\theta)), \tag{B.2}$$

where $\mathbf{J}_\mathbf{u}$ is the Jacobian of $\mathbf{u}$ w.r.t. θ, a matrix of size $|\mathcal{Y}| \times (N \times N)$. The horizontal slices of $\mathbf{J}_\mathbf{u}$ are exactly all the paths $\mathbf{Y}$ of $\mathcal{Y}$. Using $\nabla \max_\Omega(\mathbf{u}(\theta)) \in \triangle^{|\mathcal{Y}|}$, we conclude that $\nabla \mathrm{LP}_\Omega(\theta) \in \mathrm{conv}(\mathcal{Y})$.

INDUCED DISTRIBUTION. From (B.2), we see that the regularized gradient $\nabla \mathrm{LP}_\Omega(\theta)$ rewrites as $\sum_{\mathbf{Y} \in \mathcal{Y}} p_{\theta,\Omega}(\mathbf{Y})\, \mathbf{Y}$, where we define the distribution

$$p_{\theta,\Omega}(\mathbf{Y}) \triangleq \Big(\nabla \max_\Omega(\mathbf{u}(\theta))\Big)_{\mathbf{Y} \in \mathcal{Y}}.$$

Unfortunately, since $\mathbf{u}(\theta) \in \mathbb{R}^{|\mathcal{Y}|}$, computing $p_{\theta,\Omega}(\mathbf{Y})$, let alone the expectation $\mathbb{E}_{\theta,\Omega}[\mathbf{Y}]$ under that distribution, is intractable for general Ω.

B.1.5 *Proof of Proposition 8.3 (computation of $\nabla DP_\Omega(\theta)$)*

GRADIENT COMPUTATION. We first derive the recursion over $E \triangleq \nabla \mathrm{DP}_\Omega(\theta)$ using sensitivity analysis, a.k.a backpropagation calculus. For any $(i,j) \in \mathcal{E}$, since $\theta_{i,j}$ influences only v_i, a straightforward application of the chain rule gives

$$e_{i,j} = \frac{\partial v_N}{\partial \theta_{i,j}} = \frac{\partial v_N}{\partial v_i} \frac{\partial v_i}{\partial \theta_{i,j}}. \tag{B.3}$$

Recall that $\boldsymbol{v} = (v_1, \ldots, v_N)$ and $\mathbf{q}_i \triangleq \nabla \max_\Omega(\theta_i + \boldsymbol{v})$. With this vector defined, we can now easily derive the two terms on the r.h.s of (B.3). Differentiating (8.6) w.r.t. $\theta_{i,j}$ straightforwardly gives the second term $\frac{\partial v_i}{\partial \theta_{i,j}} = q_{i,j}$.

The first term must be computed recursively. Recall that $\mathcal{C}_j$ denotes the children of node j. Since a node j influences only its children $i \in \mathcal{C}_j$, using the chain rule, we get

$$\frac{\partial v_N}{\partial v_j} = \sum_{i \in \mathcal{C}_j} \frac{\partial v_N}{\partial v_i} \frac{\partial v_i}{\partial v_j} \triangleq \bar{e}_j. \tag{B.4}$$

Differentiating (8.6) w.r.t. v_j again gives $\frac{\partial v_i}{\partial v_j} = q_{i,j}$. By definition, we also have $\frac{\partial v_N}{\partial v_i} = \bar{e}_i$ and $e_{i,j} = \bar{e}_i q_{i,j}$. Hence,

$$\bar{e}_j = \sum_{i \in \mathcal{C}_j} \bar{e}_i q_{i,j} = \sum_{i \in \mathcal{C}_j} e_{i,j}.$$

Combining the above, for any $j \in [N-1]$, we obtain the following two-step recursion

$$\forall i \in \mathcal{C}_j,\ e_{i,j} = \bar{e}_i q_{i,j} \quad \text{and} \quad \bar{e}_j = \sum_{i \in \mathcal{C}_j} e_{i,j}.$$

The values $(e_{i,j})_{(i,j)\in\mathcal{E}}$ can thus be computed in reverse topological order over the nodes of G, initializing $\bar{e}_N = \frac{\partial \nu_N}{\partial \nu_N} = 1$. The pseudo-code is summarized in Algorithm 5.

ASSOCIATED RANDOM WALK. It remains to show that E is also the expectation of $Y \in \mathcal{Y}$ support of the following random walk, defined informally in the main text. Formally, we define the random sequence $(w_t)_t$ as

$$w_0 = N, \quad \forall t > 0, \forall i \in [N], \forall j \in \mathcal{P}_i, \quad \mathbb{P}[w_t = j | w_{t-1} = i] = q_{i,j}.$$

We set $y_{i,j} \triangleq \mathbf{1}\{\exists\, t > 0 \text{ s.t. } w_{t-1} = i, w_t = j\}$ where $\mathbf{1}$ is the characteristic function of an event, thereby defining a random variable $Y \in \mathcal{Y}$, with distribution $\mathcal{D}$. We leave implicit the dependency of $\mathbb{P}$ in θ and Ω. As the depth of w_t (number of edges to connect to the root node) is strictly decreasing with t, $(w_t)_t$ reaches node 1 in finite time with probability one and is constant after this event. We introduce the random variables $(\bar{y}_j)_j$, defined for all $j \in [N]$ as

$$\bar{y}_j \triangleq \mathbf{1}\{\exists\, t \geqslant 0, w_t = j\} = \sum_{i \in \mathcal{C}_j} y_{i,j} \text{ if } j \neq N, 0 \text{ otherwise.}$$

By definition, using the fact that $\mathbb{P}[w_t = j | w_{t-1} = i]$ is independent of t (Markov property), for all $i \in \mathcal{C}_j$ and for all $j \in [N-1]$, we have

$$\begin{aligned}\mathbb{P}[y_{i,j} = 1] &= \mathbb{E}[y_{i,j}] \\ &= \mathbb{P}[\exists\, t > 0, w_{t-1} = i]\mathbb{P}[w_t = j | w_{t-1} = i] = \mathbb{E}[\bar{y}_i] q_{i,j}.\end{aligned}$$

Linearity of the expectation then provides

$$\mathbb{E}[\bar{y}_j] = \sum_{i \in \mathcal{C}_j} \mathbb{E}[y_{i,j}],$$

with initialization $\mathbb{E}[\bar{y}_N] = 1$. We recover the same two-step recursion as the one defining E and $\bar{e}$, with the same initialization. Hence the probabilistic interpretation of the gradient, where the expectation is taken with respect to the distribution $\mathcal{D}$ of Y:

$$E = \mathbb{E}_{\theta,\Omega}[Y] \quad \text{and} \quad \bar{e} = \mathbb{E}_{\theta,\Omega}[\bar{y}].$$

Algorithm 5 Compute $\mathrm{DP}_\Omega(\theta)$ and $\nabla\mathrm{DP}_\Omega(\theta)$

Input: Edge weights $\theta \in \mathbb{R}^{N\times N}$

$\nu_1 \leftarrow 0, \quad \bar{e}_N \leftarrow 1, \quad Q, E \leftarrow 0 \in \mathbb{R}^{N\times N}$

for $i \in [2,\dots,N]$ **do** ▷ Topological order

$\nu_i \leftarrow \max_{\Omega\, j\in\mathcal{P}_i} \theta_{i,j} + \nu_j$

$(q_{i,j})_{j\in\mathcal{P}_i} \leftarrow \nabla \max_{\Omega\, j\in\mathcal{P}_i} \theta_{i,j} + \nu_j$

for $j \in [N-1,\dots,1]$ **do** ▷ Reverse topological order

$\forall i \in \mathcal{C}_j,\ e_{i,j} \leftarrow q_{i,j}\bar{e}_i, \quad \bar{e}_j \leftarrow \sum_{i\in\mathcal{C}_j} e_{i,j}$

Return: $\mathrm{DP}_\Omega(\theta) = \nu_N$, $\nabla\mathrm{DP}_\Omega(\theta) = E \in \mathbb{R}^{N\times N}$

Intermediate computation for Algorithm 6

$\bar{e} \triangleq [\bar{e}]_{i=1}^N \in \mathbb{R}^N$, $Q \in \mathbb{R}^{N\times N}$

Algorithm 6 Compute $\langle\nabla\mathrm{DP}_\Omega(\theta), Z\rangle$ and $\nabla^2\mathrm{DP}_\Omega(\theta)Z$

Input: Edge weights and perturbation $\theta, Z \in \mathbb{R}^{N\times N}$

Call Algorithm 5 with input θ to get $\bar{e}$ and Q

$\dot{\nu}_1 \leftarrow 0; \quad \dot{e}_N \leftarrow 0, \quad \dot{Q}, \dot{E} \leftarrow 0 \in \mathbb{R}^{N\times N}$

for $i \in [2,\dots,N]$ **do** ▷ Topological order

$\dot{\nu}_i \leftarrow \sum_{j\in\mathcal{P}_i} q_{i,j}(z_{i,j} + \dot{\nu}_j)$ (A1)

$(\dot{q}_{i,j})_{j\in\mathcal{P}_i} \leftarrow J_\Omega\big((q_{i,j})_{j\in\mathcal{P}_i}\big)(z_{i,j} + \dot{\nu}_j)_{j\in\mathcal{P}_i}$ (A2)

for $j \in [N-1,\dots,1]$ **do** ▷ Reverse topological order

$\forall i \in \mathcal{C}_j,\ \dot{e}_{i,j} \leftarrow \dot{q}_{i,j}\bar{e}_i + q_{i,j}\dot{e}_i$ (A3)

$\dot{e}_j \leftarrow \sum_{i\in\mathcal{C}_j} \dot{e}_{i,j}$

Return: $\langle\nabla\mathrm{DP}_\Omega(\theta), Z\rangle = \dot{\nu}_N$

$\nabla^2\mathrm{DP}_\Omega(\theta)Z = \dot{E} \in \mathbb{R}^{N\times N}$

B.1.6 *Computation of the directional derivative* $\langle\nabla DP_\Omega(\theta), Z\rangle$

The derivations of the following two sections allows to write Algorithm 6. Let $\dot{\nu}_i \triangleq \langle\nabla\nu_i(\theta), Z\rangle$, where $\nu_i(\theta)$ is defined in (8.6). Since ν_i only directly depends on $\nu_j + \theta_{i,j}$ for $j \in \mathcal{P}_i$, a straightforward differentiation of $\langle\nabla\nu_i(\theta), Z\rangle$ gives

$$\dot{\nu}_i = \sum_{j\in\mathcal{P}_i} \frac{\partial\nu_i}{\partial\nu_j}\ (\dot{\nu}_j + z_{i,j})\,.$$

Recall that $\frac{\partial\nu_i}{\partial\nu_j} = q_{i,j}$ and has already been obtained when computing $\nabla\mathrm{DP}_\Omega(\theta)$. Hence equation (A1), reproduced here:

$$\forall i \in [2,\dots,N]: \qquad \dot{\nu}_i = \sum_{j\in\mathcal{P}_i} q_{i,j}(\dot{\nu}_j + z_{i,j}). \tag{B.5}$$

This recursion can be computed in topological order, starting from $\dot{\nu}_1 = 0$ to finish at $\dot{\nu}_N = \langle\nabla\mathrm{DP}_\Omega(\theta), Z\rangle$.

B.1.7 *Computation of the Hessian-vector product $\nabla^2 DP_\Omega(\theta)\mathbf{Z}$*

For convenience, let us define $\nabla^2 \mathrm{DP}_\Omega(\theta)\mathbf{Z} \triangleq \dot{\mathbf{E}}$. For $(i,j) \notin \mathcal{E}$, we evidently have $\dot{e}_{i,j} = 0$. For $(i,j) \in \mathcal{E}$, since $\theta_{i,j}$ influences only v_i and $\dot{v}_i$, we obtain

$$\dot{e}_{i,j} = \frac{\partial \dot{v}_N}{\partial \theta_{i,j}} = \frac{\partial \dot{v}_N}{\partial v_i}\frac{\partial v_i}{\partial \theta_{i,j}} + \frac{\partial \dot{v}_N}{\partial \dot{v}_i}\frac{\partial \dot{v}_i}{\partial \theta_{i,j}}.$$

We will now show how to derive each of the right-hand side terms in turn. We already know that $\frac{\partial v_i}{\partial \theta_{i,j}} = q_{i,j}$. We also have $\frac{\partial \dot{v}_N}{\partial \dot{v}_i} = u_i$. Indeed, observe that $\dot{v}_j$ only directly influences $\dot{v}_i$ for $i \in \mathcal{C}_i$. Therefore, we have

$$\frac{\partial \dot{v}_N}{\partial \dot{v}_j} = \sum_{i \in \mathcal{C}_j} \frac{\partial \dot{v}_N}{\partial \dot{v}_i} q_{i,j} \qquad \forall j \in [N-1] \tag{B.6}$$

and $\frac{\partial \dot{v}_N}{\partial \dot{v}_1} = 1$. Comparing (B.4) and (B.6), we see that $(\frac{\partial \dot{v}_N}{\partial \dot{v}_i})_i$ follows the same recursion as $(\frac{\partial v_N}{\partial v_i})_i$. Since $\frac{\partial \dot{v}_N}{\partial \dot{v}_n} = \frac{\partial v_N}{\partial v_n}$, both sequences are equal:

$$\frac{\partial \dot{v}_N}{\partial \dot{v}_i} = \frac{\partial v_N}{\partial v_i} = e_i.$$

Next, we derive $\frac{\partial \dot{v}_i}{\partial \theta_{i,j}}$. Since, for $j \in \mathcal{P}_i$, $\dot{v}_j + z_{i,j}$ does not depend on $\theta_{i,j}$, differentiating (B.5) w.r.t. $\theta_{i,j}$, we obtain

$$\begin{aligned}\frac{\partial \dot{v}_i}{\partial \theta_{i,j}} &= \sum_{k \in \mathcal{P}_i} \frac{\partial q_{i,j}}{\partial \theta_{i,j}}(\dot{v}_k + z_{i,k}) \\ &= \sum_{k \in \mathcal{P}_i} \frac{\partial^2 v_i}{\partial \theta_{i,j}\partial \theta_{i,k}}(\dot{v}_k + z_{i,k}) \triangleq \dot{q}_{i,j}.\end{aligned}$$

This can be conveniently rewritten in a vectorial form as

$$\dot{\mathbf{q}}_i = \nabla^2 \max_\Omega(\theta_i + \mathbf{v})\,(\mathbf{z}_i + \dot{\mathbf{v}}) = \mathbf{J}_\Omega(\mathbf{q}_i)\,(\mathbf{z}_i + \dot{\mathbf{v}}),$$

where we have defined $\dot{\mathbf{v}} \triangleq (\dot{v}_1, \dots, \dot{v}_N)$ and where we have used the function $\mathbf{J}_\Omega$ defined in Section B.2.1, that conveniently computes the Hessian of $\max_\Omega$ from its gradient. The Hessian has this form for both negentropy and ℓ_2^2 regularizations. In a practical implementation, we only need to compute the coordinates (i,j) of $\dot{\mathbf{Q}}$, for $j \in \mathcal{P}_i$. Namely, as specified in (A2),

$$(\dot{\mathbf{q}}_{i,j})_{j \in \mathcal{P}_i} \leftarrow \mathbf{J}_\Omega\big((\mathbf{q}_{i,j})_{j \in \mathcal{P}_i}\big)(z_{i,j} + \dot{v}_j)_{j \in \mathcal{P}_i}.$$

Finally, we derive $\frac{\partial \dot{v}_N}{\partial v_i}$. Since v_j influences only v_i and $\dot{v}_i$ for $i \in \mathcal{C}_j$, the chain rule gives

$$\frac{\partial \dot{v}_N}{\partial v_i} = \sum_{j \in \mathcal{C}_i} \frac{\partial \dot{v}_N}{\partial v_j}\frac{\partial v_j}{\partial v_i} + \frac{\partial \dot{v}_N}{\partial \dot{v}_j}\frac{\partial \dot{v}_j}{\partial v_i} = \sum_{j \in \mathcal{C}_j} \dot{e}_{i,j} \triangleq \dot{e}_i.$$

Combining the above, for any $j \in [N-1]$, we obtain the following two-step recursion (A3), reproduced here:

$$\forall i \in \mathcal{C}_j, \quad \dot{e}_{i,j} = \dot{q}_{i,j} e_i + q_{i,j} \dot{e}_i \quad \text{and} \quad \dot{e}_j = \sum_{i \in \mathcal{C}_j} \dot{e}_{i,j}.$$

Similarly to the computation of $\nabla \mathrm{DP}_\Omega(\theta)$, our algorithm computes this recursion in reverse topological order over the graph G, yielding $\nabla^2 \mathrm{DP}_\Omega(\theta) Z = \dot{E}$.

B.2 EXAMPLES OF ALGORITHM INSTANTIATIONS

We provide the explicit forms of $\max_\Omega$ and its derivative for the negentropy and ℓ_2^2 regularizations. Then, we provide details and pseudo-code for the two instances of differentiable dynamic programming presented in Section 8.4.

B.2.1 *Examples of* $\max_\Omega$

Negative entropy. When $\Omega(q) = \gamma \sum_{i=1}^D q_i \log q_i$, where $\gamma > 0$ (smaller is less regularized), we obtain

$$\begin{aligned}
\max_\Omega(x) &= \gamma \log \left(\sum_{i=1}^D \exp(x_i/\gamma) \right) \\
\nabla \max_\Omega(x) &= \exp(x/\gamma) \Big/ \sum_{i=1}^D \exp(x_i/\gamma) \\
\nabla^2 \max_\Omega(x) &= J_\Omega(\nabla \max_\Omega(x)),
\end{aligned}$$

where $J_\Omega(q) \triangleq (\mathrm{Diag}(q) - qq^\top)/\gamma$. Note that $\nabla \max_\Omega(x)$ recovers the usual "softmax" with temperature $\gamma = 1$. For a proof of the expression of $\max_\Omega$, see, *e.g.*, (Boyd and Vandenberghe, 2004, Example 3.25).

SQUARED ℓ_2 NORM. When $\Omega(x) = \frac{\gamma}{2}\|x\|_2^2$ with $\gamma > 0$, we obtain the following expressions

$$\begin{aligned}
\max_\Omega(x) &= \langle q^\star, x \rangle - \frac{\gamma}{2}\|q^\star\|_2^2 \\
\nabla \max_\Omega(x) &= \operatorname*{argmin}_{q \in \triangle^D} \|q - x/\gamma\|_2^2 = q^\star \\
\nabla^2 \max_\Omega(x) &= J_\Omega(\nabla \max_\Omega(x)),
\end{aligned}$$

where $J_\Omega(q) \triangleq (\mathrm{Diag}(s) - ss^\top/\|s\|_1)/\gamma$ and $s \in \{0,1\}^D$ is a vector that indicates the support of q. Note that $\nabla \max_\Omega(x)$ is precisely the Euclidean projection onto the simplex of x/γ and can be computed exactly in worst-case $\mathcal{O}(D \log D)$ time using the algorithm of (Michelot, 1986) or in expected $\mathcal{O}(D)$ time using the randomized pivot algorithm of (Duchi et al., 2008). It can be efficiently performed on Nvidia GPUs since recently. An important benefit of the squared ℓ_2 norm, compared to the negative entropy, is that $\nabla \max_\Omega(x)$ tends to be sparse. This is useful, among other things, to define sparse attention mechanisms (Martins and Astudillo, 2016; Niculae and Blondel, 2017).

B.2.2 *Sequence prediction with the smoothed Viterbi algorithm*

COMPUTATIONAL GRAPH. As illustrated in Section 8.4, the DAG contains a start node, S nodes for each time step and end node. Therefore $|\mathcal{V}| = N = TS + 2$. Only nodes from consecutive time steps are connected to each other. Taking into account the start and end nodes, the total number of edges is therefore $|\mathcal{E}| = (T-1)S^2 + 2S$.

REPRESENTATION. We follow the notation of Section 8.4, *i.e.* we represent $\mathbf{Y}$ and θ as $T \times S \times S$ tensors (we can safely ignore the edges connected to the end node since their value is 0). We represent $\mathbf{Y}$ as a binary tensor such that $y_{t,i,j} = 1$ if $\mathbf{Y}$ is in states i and j in time steps t and $t-1$, and $y_{t,i,j} = 0$ otherwise. Likewise, we represent the potentials θ as a real tensor such that $\theta_{t,i,j}$ contains the potential of transitioning from state j to state i on time t.

ALGORITHMS. Applying recursion (8.6) to this specific DAG, we obtain a smoothed version of the Viterbi algorithm. Let $\nu_{t,i}$ be the score of being in state i up to time t. We can rewrite the smoothed Bellman recursion as

$$\nu_{t,i}(\theta) \triangleq \max_{\Omega}_{j \in [S]} \nu_{t-1,j}(\theta) + \theta_{t,i,j} = \max_{\Omega}(\boldsymbol{\nu}_{t-1}(\theta) + \theta_{t,i}).$$

The value $\mathrm{Vit}_{\Omega}(\theta) \triangleq \max_{\Omega}(\boldsymbol{\nu}_T(\theta))$ can be computed in topological order, starting from $\boldsymbol{\nu}_0(\theta)$. The total computational cost is $\mathcal{O}(TS^2)$. Using the computations of Section 8.3.3 and Section 8.3.4 to this specific DAG, we can compute $\nabla \mathrm{Vit}_{\Omega}(\theta)$, $\langle \nabla \mathrm{Vit}_{\Omega}(\theta), Z \rangle$ and $\nabla^2 \mathrm{Vit}_{\Omega}(\theta) Z$ with the same complexity. The procedures are summarized in Algorithm 7 and Algorithm 8, respectively. From Proposition 8.2 property 1, $\mathrm{Vit}_{\Omega}(\theta)$ is a convex function for any Ω.

Algorithm 7 Compute $\mathrm{Vit}_{\Omega}(\theta)$ and $\nabla \mathrm{Vit}_{\Omega}(\theta)$

Input: Potential scores $\theta \in \mathbb{R}^{T \times S \times S}$

▷ Forward pass

$\boldsymbol{\nu}_0 = \mathbf{0}_S$

for $t \in [1, \dots, T], i \in [S]$ **do**

$\nu_{t,i} = \max_{\Omega}(\theta_{t,i} + \boldsymbol{\nu}_{t-1})$

$\mathbf{q}_{t,i} = \nabla \max_{\Omega}(\theta_{t,i} + \boldsymbol{\nu}_{t-1})$

$\nu_{T+1,1} = \max_{\Omega}(\boldsymbol{\nu}_T)$; $\quad \mathbf{q}_{T+1,1} = \nabla \max_{\Omega}(\boldsymbol{\nu}_T)$

▷ Backward pass

$\mathbf{u}_{T+1} = (1, 0, \dots, 0) \in \mathbb{R}^S$

for $t \in [T, \dots, 0], j \in [S]$ **do**

$\mathbf{e}_{t,\cdot,j} = \mathbf{q}_{t+1,\cdot,j} \circ \mathbf{u}_{t+1}$; $\quad u_{t,j} = \langle \mathbf{e}_{t,\cdot,j}, \mathbf{1}_S \rangle$

Return: $\mathrm{Vit}_{\Omega}(\theta) = \nu_{T+1,1}$

$\nabla \mathrm{Vit}_{\Omega}(\theta) = (e_{t-1,i,j})_{t=1,i,j=1}^{T,S,S}$

Intermediary computations for Alg. 8:

$\mathbf{Q} \triangleq (q)_{t=1,i,j=1}^{T+1,S,S}, \mathbf{U} \triangleq (u)_{t=1,j=1}^{T+1,S}$

Algorithm 8 Compute $\langle \nabla \mathrm{Vit}_\Omega(\theta), \mathbf{Z} \rangle$ and $\nabla^2 \mathrm{Vit}_\Omega(\theta)\mathbf{Z}$

Input: $\mathbf{Z} \in \mathbb{R}^{T \times S \times S}, \theta \in \mathbb{R}^{T \times S \times S}$
Call Alg. 7 with input θ to get $\mathbf{U}$, $\mathbf{Q}$
▷ Forward pass
$\dot{v}_0 = 0_S$
for $t \in [1, \dots, T], i \in [S]$ **do**
  $\dot{v}_{t,i} = \langle \mathbf{q}_{t,i}, \mathbf{z}_{t,i} + \dot{v}_{t-1} \rangle$
  $\dot{\mathbf{q}}_{t,i} = \mathbf{J}_\Omega(\mathbf{q}_{t,i})\ (\mathbf{z}_t + \dot{v}_{t-1})$
$\dot{v}_{T+1,1} = \langle \mathbf{q}_{T+1,1}, \dot{v}_T \rangle; \quad \dot{\mathbf{q}}_{T+1,1} = \mathbf{J}_\Omega(\dot{\mathbf{q}}_{T+1,1})\ \dot{v}_T$
▷ Backward pass
$\dot{\mathbf{u}}_{T+1} = 0_S; \quad \dot{\mathbf{Q}}_{T+1} = 0_{S \times S}$
for $t \in [T, \dots, 0], j \in [S]$ **do**
  $\dot{\mathbf{e}}_{t,\cdot,j} = \mathbf{q}_{t+1,\cdot,j} \circ \dot{\mathbf{u}}_{t+1} + \dot{\mathbf{q}}_{t+1,\cdot,j} \circ \mathbf{u}_{t+1}$
  $\dot{\mathbf{u}}_{t,j} = \langle \dot{\mathbf{e}}_{t,\cdot,j}, \mathbf{1}_S \rangle$
Return: $\langle \mathrm{Vit}_\Omega(\theta), \mathbf{Z} \rangle = \dot{v}_{T+1}$
$\nabla^2 \mathrm{Vit}_\Omega(\theta)\mathbf{Z} = (\dot{\mathbf{e}}_{t-1,i,j})_{t=1,i,j=1}^{T,S,S}$

B.2.3 *Monotonic alignment prediction with the smoothed DTW*

COMPUTATIONAL GRAPH. As illustrated in Section 8.4, the DAG contains a start node and $N_A N_B$ nodes. Therefore, the number of vertices $|\mathcal{V}|$ is $N_A N_B + 1$. Due to the monotonic constraint, each node may only be connected with at most 3 other nodes. The cardinality of $\mathcal{Y}$ is the Delannoy number $(N_A - 1, N_B - 1)$, as studied by Banderier and Schwer (2005) and Sulanke (2003). That number grows exponentially with N_A and N_B.

REPRESENTATION. We follow the notation of Section 8.4, *i.e.* we represent $\mathbf{Y}$ and θ as $N_A \times N_B$ matrices. We represent $\mathbf{Y}$ as a binary matrix such that $y_{i,j} = 1$ if $\mathbf{a}_i$ is aligned with $\mathbf{b}_j$, and $y_{i,j} = 0$ otherwise. Likewise, we represent θ as a real matrix such that $\theta_{i,j}$ is a measure of "discrepancy" between $\mathbf{a}_i$ and $\mathbf{b}_j$.

ALGORITHMS. Following the DTW literature (Sakoe and Chiba, 1978), we seek an alignment with *minimal* cost. For that reason, we introduce the smoothed min operator, its gradient and its Hessian as follows

$$\begin{aligned} \min_\Omega(\mathbf{x}) &\triangleq -\max_\Omega(-\mathbf{x}) \\ \nabla \min_\Omega(\mathbf{x}) &= \nabla \max_\Omega(-\mathbf{x}) \\ \nabla^2 \min_\Omega(\mathbf{x}) &= -\nabla^2 \max_\Omega(-\mathbf{x}) \\ &= -\mathbf{J}_\Omega(\nabla \max_\Omega(-\mathbf{x})) \\ &= -\mathbf{J}_\Omega(\nabla \min_\Omega(\mathbf{x})). \end{aligned}$$

Applying (8.6) to the DTW DAG gives rise to a smoothed version of the algorithm. Let $v_{i,j}(\theta)$ be the alignment cost up to cell (i, j). Then the smoothed DTW recursion is

$$v_{i,j}(\theta) = \theta_{i,j} + \min_\Omega(v_{i,j-1}(\theta), v_{i-1,j-1}(\theta), v_{i-1,j}(\theta))$$

The value $\mathrm{DTW}_\Omega(\theta) \triangleq \nu_{N_A,N_B}(\theta)$ can be computed in $\mathcal{O}(N_A N_B)$ time. Applying the derivations of Section 8.3.3 and Section 8.3.4 to this specific DAG, we can compute $\nabla\mathrm{DTW}_\Omega(\theta)$, $\langle \nabla\mathrm{DTW}_\Omega(\theta), Z\rangle$ and $\nabla^2\mathrm{DTW}_\Omega(\theta)Z$ with the same complexity. The procedures, with appropriate handling of the edge cases, are summarized in Algorithm 9 and 10, respectively.

Note that when Ω is the negative entropy, $\mathrm{DTW}_\Omega(\theta)$ is known as soft-DTW (Cuturi and Blondel, 2017). While the DP computation of $\mathrm{DTW}_\Omega(\theta)$ and of its gradient were already known, the generalization to any strongly convex Ω and the computation of $\nabla^2\mathrm{DTW}_\Omega(\theta)Z$ are new. From Proposition 8.2 property 1, $\mathrm{DTW}_\Omega(\theta)$ is a *concave* function of the discrepancy matrix θ for any Ω. With respect to time-series, DTW_Ω is neither convex nor concave.

Algorithm 9 Compute $\mathrm{DTW}_\Omega(\theta)$ and $\nabla\mathrm{DTW}_\Omega(\theta)$

Input: Distance matrix $\theta \in \mathbb{R}^{N_A \times N_B}$

▷ Forward pass

$\nu_{0,0} = 0$; $\nu_{i,0} = \nu_{0,j} = \infty$, $i \in [N_A], j \in [N_B]$

for $i \in [1, \dots, N_A], j \in [1, \dots, N_B]$ **do**

$\nu_{i,j} = d_{i,j} + \min_\Omega(\nu_{i,j-1}, \nu_{i-1,j-1}, \nu_{i-1,j})$

$q_{i,j} = \nabla\min_\Omega(\nu_{i,j-1}, \nu_{i-1,j-1}, \nu_{i-1,j}) \in \mathbb{R}^3$

▷ Backward pass

$q_{i,N_B+1} = q_{N_A+1,j} = 0_3$, $i \in [N_A], j \in [N_B]$

$e_{i,N_B+1} = e_{N_A+1,j} = 0$, $i \in [N_A], j \in [N_B]$

$q_{N_A+1,N_B+1} = (0, 1, 0)$; $e_{N_A+1,N_B+1} = 1$

for $j \in [N_B, \dots, 1], i \in [N_A, \dots, 1]$ **do**

$e_{i,j} = q_{i,j+1,1}\, e_{i,j+1} + q_{i+1,j+1,2}\, e_{i+1,j+1} + q_{i+1,j,3}\, e_{i+1,j}$

Return: $\mathrm{DTW}_\Omega(\theta) = \nu_{N_A,N_B}$

$\nabla\mathrm{DTW}_\Omega(\theta) = (e)_{i,j=1}^{N_A,N_B}$

Intermediate computations for Algo. 10:

$Q \triangleq (q)_{i,j,k=1}^{N_A+1,N_B+1,3}$; $E \triangleq (e)_{i,j=1}^{N_A+1,N_B+1}$

Algorithm 10 Compute $\langle \nabla \mathrm{DTW}_\Omega(\theta), \mathbf{Z} \rangle$, $\nabla^2 \mathrm{DTW}_\Omega(\theta)\ \mathbf{Z}$

Input: $\theta \in \mathbb{R}^{N_A \times N_B}, \mathbf{Z} \in \mathbb{R}^{N_A \times N_B}$
Call Algo. 9 with input θ to retrieve $\mathbf{Q}$ and $\mathbf{E}$
▷ Forward pass
$\dot{v}_{i,0} = \dot{v}_{0,j} = 0,\ i \in [0, \dots, N_A], j \in [N_B]$
for $i \in [1, \dots, N_B], j \in [1, \dots, N_A]$ **do**
  $\dot{v}_{i,j} = z_{i,j} + q_{i,j,1}\ \dot{v}_{i,j-1} + q_{i,j,2}\ \dot{v}_{i-1,j-1} + q_{i,j,3}\ \dot{v}_{i-1,j}$
  $\dot{q}_{i,j} = -J_\Omega(q_{i,j})\ (\dot{v}_{i,j-1}, \dot{v}_{i-1,j-1}, \dot{v}_{i-1,j}) \in \mathbb{R}^3$
▷ Backward pass
$\dot{q}_{i,N_B+1} = \dot{q}_{N_A+1,j} = 0_3,\ i \in [0, \dots, N_A], j \in [N_B]$
$\dot{e}_{i,N_B+1} = \dot{e}_{N_A+1,j} = 0,\ i \in [0, \dots, N_A], j \in [N_B]$
for $j \in [N_B, \dots, 1], i \in [N_A, \dots, 1]$ **do**
  $\dot{e}_{i,j} = \dot{q}_{i,j+1,1}\ e_{i,j+1} + q_{i,j+1,1}\ \dot{e}_{i,j+1} +$
    $\dot{q}_{i+1,j+1,2}\ e_{i+1,j+1} + q_{i+1,j+1,2}\ \dot{e}_{i+1,j+1} +$
    $\dot{q}_{i+1,j,3}\ e_{i+1,j} + q_{i+1,j,3}\ \dot{e}_{i+1,j}$
Return: $\langle \nabla \mathrm{DTW}_\Omega(\theta), \mathbf{Z} \rangle = \dot{v}_{N_A,N_B}$
$\nabla^2 \mathrm{DTW}_\Omega(\theta)\ \mathbf{Z} = (\dot{e})_{i,j=1}^{N_A,N_B}$

B.3 EXPERIMENTAL DETAILS AND FURTHER RESULTS

Finally, we provide details on the architecture used in experiments, with additional figures.

B.3.1 *Named entity recognition (Section 8.5.2)*

Our model extracts word embedding from a 300-dimensional look-up table concatenated with a 50-dimensional character embedding. This character embedding corresponds to the concatenation of the last hidden unit of a bi-directional character LSTM, as in Lample et al. (2016). Character embedding size is set to 50. A word LSTM then produces sentence-aware features for each word. This LSTM is bi-directional with 100-dimensional hidden units per direction. The final features $\mathbf{X}$ used to build the potential tensor θ are thus 200-dimensional. Note that, in contrast with Lample et al. (2016):

— The look-up table is initialized with 300-dimensional embeddings from *FastText* (Joulin et al., 2016), trained on Wikipedia corpus.
— We do not pad letters prior to feeding the character LSTM as it is not principled.
— We do not train the unknown word embedding as we found it had no effect.

We convert tags to the IOBES (Inside-Outside-Begin-End-Stop) scheme to build a richer Vit_Ω model than if we used the simpler IOB (Inside-Outside-Begin) scheme, that has a lower number of tags. We performed a small grid-search to select the step-size and batch-size used for optimization: $s \in \{0.005, 0.01, 0.02\}$, $b \in \{8, 32, 128\}$. For each language and each loss, we select the highest-scoring model on the validation set, and report the test score.

The model is strongly subject to overfitting using the convex surrogate loss and the log likelihood. We have to use a small batch size ($b = 8$) and vanilla SGD with large step size ($s = 0.01$) to avoid this overfitting issue. For all losses, accelerated stochastic optimizers have all lower generalization performance than SGD. This was also noticed by Lample et al. (2016) when using the classical negative log-likelihood as a loss.

B.3.2 *Supervised audio-to-score transcription (Section 8.5.3)*

Audio sequences, sampled at 22.05 kHz, are split into frames of 512 samples. We extract the following features from these sequences: energy, spectral centroid, spectral bandwidth, and the 5 first Mel-frequency cepstral coefficients (MFCC) features. All features are centered around the median and normalized. The $\nabla \mathrm{DTW}_\Omega$ layer is written in *Cython* [1], and hence run on CPU. This technical choice was suggested by the fact that we have to write explicit loops to specify the topological and reverse topological pass over the DTW computation graph (see Algorithm 9). However, it is possible to use only contiguous vector operations and thus take advantage of GPU computations — this is left for future work. We use *SciPy*'s [2] LBFGS-B solver to perform end-to-end training and multinomial regression. We use a ℓ_2^2 regularization on the weight $\boldsymbol{W}_{:}$ we selected it using a grid search over $\{10^{-5}, 10^{-4}, \ldots, 1\}$ and selected 10^{-3}.

B.3.3 *Structured and sparse attention (Section 8.6)*

We use *OpenNMT-py* library [3] to fit our structured attention model. Model architecture and optimization details are as follow:

- We use a bidirectional LSTM encoder and decoder, with 500 units in each direction and a depth of 2 layers .
- The decoder is fed with the input representation as in Luong et al. (2015).
- SGD training with $s = 1$ learning rate, decaying from epoch 8 to epoch 15 with rate 0.65, batch size of size 256.
- Training sentence of lengths superior to 50 are ignored, and translated sentence are forced to a length inferior to 100.
- The temperature parameter is set to $\gamma = 2$ for entropy, and $\gamma = 10$ for ℓ_2^2. Performance is not affected much by this parameter, provided that it is not set too low in the ℓ_2^2 case — with a too small γ, Vit_Ω reduces to unregularized MAP estimation and $\nabla \mathrm{Vit}_\Omega$ has zero derivatives.

We use a 1-million sentence subject of WMT14 English-to-French corpus, available at http://nmt-benchmark.net/. We use Moses tokenizer and do not perform any post-processing, before computing BLEU score on detokenized sentences (*multi_bleu.perl* script).

1. http://cython.org/
2. http://scipy.org/
3. http://opennmt.net/

Table B.1 – Detokenized BLEU score on newstest2014 data, comparing softmax attention with structured attention.

Attention model	WMT14 1M fr→en	WMT14 en→fr
Softmax	**27.96**	**28.08**
Entropy regularization	**27.96**	27.98
ℓ_2^2 reg.	27.21	27.28

IMPLEMENTATION. We implemented a batch version of the ∇Vit_Ω layer on GPU, using the *PyTorch* tensor API. Model with negentropy-regularized attention mechanism runs 1/2 as fast as the softmax attention mechanism (approximately 7500 tokens/s vs 15000 tokens/s on a single Nvidia Titan X Pascal). With ℓ_2^2 regularization, it is only 1/3 as fast: approximately 5000 tokens/s. Although this remains reasonable, it could certainly be optimized by rewriting kernels using lower-level languages (*e.g.,* using *ATen* API from *PyTorch*.)

FURTHER RESULTS. Table B.1 provides BLEU scores for both translation directions on the 1 million sentence subset of WMT14 we used. We observe that the introduction of structure and sparsity does not hinder the general performance of the model. We provide several examples of attention maps in Figure B.2, that illustrate the sparsity patterns ℓ_2^2 regularization uncovers.

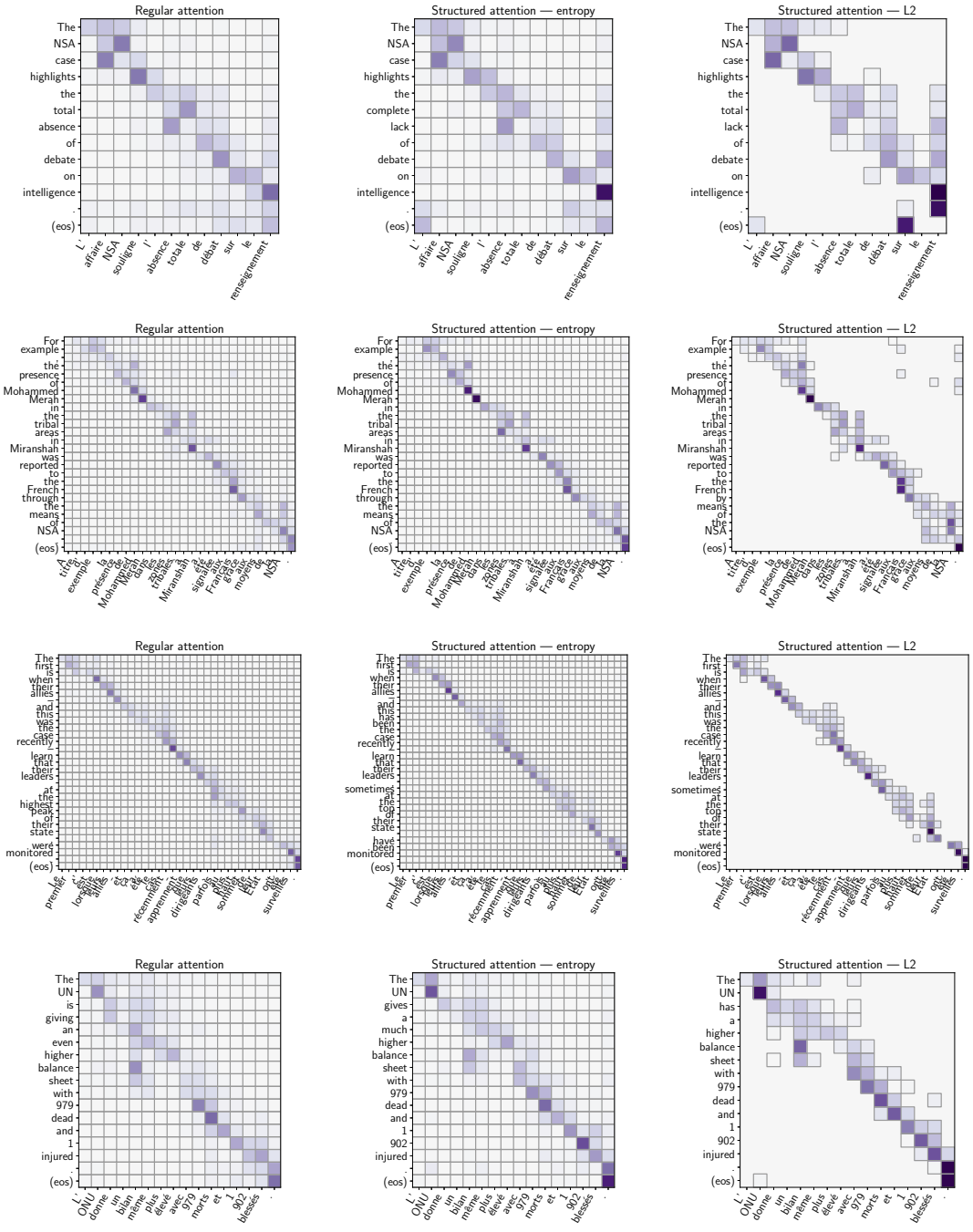


Figure B.2: Attention on test samples from *Newstest2014*. Borders indicate non-zero cells. Translations (y-axis) are often qualitatively equivalent, while attentions maps are sparse in the ℓ_2^2 case.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Apprentissage de représentation en imagerie fonctionnelle

Mots clés : Apprentissage, imagerie fonctionnelle, factorisation de matrice, dictionnaire, optimisation

Résumé : Grâce aux avancées technologiques dans le domaine de l'imagerie fonctionnelle cérébrale, les neurosciences cognitives accumulent une grande quantité de cartes spatiales décrivant de manière quantitative l'activité neuronale suscitée dans le cerveau humain en réponse à des tâches ou des stimuli spécifiques, ou de manière spontanée. Dans cette thèse, nous nous intéressons particulièrement aux données issues de l'imagerie par résonance magnétique fonctionnelle (IRMf), que nous étudions dans un cadre d'apprentissage statistique. Notre objectif est d'apprendre des modèles d'activité cérébrale à partir des données. Nous proposons différentes nouvelles manières de profiter de la grande quantité de données IRMf disponible. Tout d'abord, nous considérons les données d'IRMf de repos, que nous traitons grâce à des méthodes de factorisation de matrices. Nous présentons de nouvelles méthodes pour calculer en un temps raisonnable une factorisation parcimonieuse de matrices constituées de centaines d'enregistrements d'IRMf. Cela nous permet d'extraire des réseaux fonctionnels à partir de données d'une envergure inédite. Notre méthode principale introduit une réduction aléatoire de la dimension des données dans une boucle d'apprentissage en ligne. L'algorithme proposé converge plus de 10 fois plus vite que les meilleures méthodes existantes, pour différentes configurations et sur plusieurs jeux de données. Nous effectuons une vaste validation expérimentale de notre approche de sous-échantillonnage aléatoire. Nous proposons une étude théorique des propriétés de convergence de notre algorithme. Dans un second temps, nous nous intéressons aux données d'IRMf d'activation. Nous démontrons comment agréger différents études acquises suivant des protocoles distincts afin d'apprendre des modèles joints de décodage plus justes et interprétables. Notre modèle multi-études apprend à réduire la dimension des images cérébrales en entrée en même temps qu'il apprend à les classifier, pour chacune des études, à partir de leurs représentations réduites. Cela suscite un transfert d'information entre les études. En conséquence, notre modèle multi-étude est plus performant que les modèles de décodage appris sur chaque étude séparément. Notre approche identifie une représentation universellement pertinente de l'activité cérébrale, supportée par un petit nombre de réseaux optimisés pour l'identification de tâches.

Title: Learning representations from functional MRI data

Keywords: Machine learning, functional imaging, matrix factorization, dictionary, optimization, deep learning

Abstract: Thanks to the advent of functional brain-imaging technologies, cognitive neuroscience is accumulating maps of neural activity responses to specific tasks or stimuli, or of spontaneous activity. In this work, we consider data from functional Magnetic Resonance Imaging (fMRI), that we study in a machine learning setting: we learn a model of brain activity that should generalize on unseen data. After reviewing the standard fMRI data analysis techniques, we propose new methods and models to benefit from the recently released large fMRI data repositories. Our goal is to learn richer representations of brain activity. We first focus on unsupervised analysis of terabyte-scale fMRI data acquired on subjects at rest (resting-state fMRI). We perform this analysis using matrix factorization. We present new methods for running sparse matrix factorization/dictionary learning on hundreds of fMRI records in reasonable time. Our leading approach relies on introducing randomness in stochastic optimization loops and provides speed-up of an order of magnitude on a variety of settings and datasets. We provide an extended empirical validation of our stochastic subsampling approach, for datasets from fMRI, hyperspectral imaging and collaborative filtering. We derive convergence properties for our algorithm, in a theoretical analysis that reaches beyond the matrix factorization problem. We then turn to work with fMRI data acquired on subject undergoing behavioral protocols (task fMRI). We investigate how to aggregate data from many source studies, acquired with many different protocols, in order to learn more accurate and interpretable decoding models, that predicts stimuli or tasks from brain maps. Our multi-study shared-layer model learns to reduce the dimensionality of input brain images, simultaneously to learning to decode these images from their reduced representation. This fosters transfer learning in between studies, as we learn the undocumented cognitive common aspects that the many fMRI studies share. As a consequence, our multi-study model performs better than single-study decoding. Our approach identifies universally relevant representation of brain activity, supported by a few task-optimized networks learned during model fitting. Finally, on a related topic, we show how to use dynamic programming within end-to-end trained deep networks, with applications in language.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France