Algorithmes décentralisés et asynchrones pour l'apprentissage statistique large échelle et application à l'indexation multimédia

Jérôme Fellus




Université Paris Seine - École doctorale Sciences et Ingénierie

THÈSE

présentée pour obtenir le titre de docteur en Sciences et Technologies de l'Information et de la Communication

---

ALGORITHMES DÉCENTRALISÉS ET ASYNCHRONES POUR L'APPRENTISSAGE STATISTIQUE LARGE ÉCHELLE ET APPLICATION À L'INDEXATION MULTIMÉDIA

---

par

Jérôme Fellus

ETIS UMR 8051, Université Paris Seine, Université de Cergy-Pontoise, ENSEA, CNRS
6 avenue du Ponceau, 95014 Cergy-Pontoise, France

Soutenue le 03 octobre 2017 devant le jury composé de :

ALAIN RAKOTOMAMONJY, Professeur des Universités, LITIS EA 4108 - Université de Rouen.	Rapporteur
FRÉDÉRIC PRECIOSO, Professeur des Universités, I3S - UMR CNRS 7271 - Université de Nice Sophia Antipolis.	Rapporteur
FRANCIS BACH, Chercheur, INRIA - équipe-projet SIERRA - École Normale Supérieure.	Examinateur
NICOLAS THOME, Professeur des Universités, CEDRIC - Conservatoire National des Arts et Métiers.	Examinateur
ELSA DUPRAZ, Maître de conférences, Lab-STICC - Telecom Bretagne.	Examinatrice
DAVID PICARD, Maître de conférences, ETIS - ENSEA - Université de Cergy-Pontoise.	Co-Encadrant
PHILIPPE-HENRI GOSSELIN, Professeur des Universités, ETIS - ENSEA - Université de Cergy-Pontoise.	Directeur de thèse

Table des matières

Introduction **9**

1 Fondations théoriques : Protocoles Gossip **17**

1.1 Apprentissage statistique et optimisation . . . 18
1.2 Optimisation distribuée . . . 19
1.2.1 Modèle de temps . . . 20
1.2.2 Optimisation par consensus . . . 21
1.3 Calcul distribué de moyennes . . . 23
1.4 Protocoles par consensus . . . 26
1.4.1 Théorie de Perron-Frobenius et ergodicité . . . 27
1.4.2 Structure du réseau et vitesse de convergence . . . 30
1.4.3 Exemples . . . 31
1.5 Protocoles Gossip . . . 33
1.5.1 Ergodicité forte, ergodicité faible . . . 35
1.5.2 Protocoles push-pull (doublement-stochastiques) . . . 36
1.5.3 Exemple : Protocole Newscast . . . 37
1.5.4 Problèmes de synchronisation implicite . . . 39
1.6 Protocoles Gossip asynchrones . . . 40
1.6.1 Stochasticité seulement en espérance . . . 41
1.6.2 Protocoles Sum-Weight . . . 43
1.6.3 AGAVG : Asynchronous Gossip Averaging . . . 46
1.6.4 Temps de convergence non-asymptotique d'AGAVG . . . 49
1.7 Protocoles Gossip non-linéaires . . . 49
1.7.1 Stratégies par pré-encodage . . . 49
1.7.2 Orchestration centralisée . . . 50
1.7.3 Protocoles Gossip perturbés . . . 51
1.8 Réseaux à topologie dynamique . . . 53
1.8.1 Ajout/suppression de liens . . . 53
1.8.2 Rupture de connexité . . . 54
1.8.3 Ajout/suppression de nœuds . . . 57
1.9 Conclusion . . . 58

2 Catégorisation non supervisée et estimation de densité **61**

2.1 Catégorisation en environnement centralisé . . . 62
2.1.1 Quantification vectorielle : K-means . . . 62
2.1.2 Catégorisation probabiliste . . . 64
2.1.3 Sensibilité aux conditions initiales et minima locaux . . . 67
2.2 Catégorisation distribuée . . . 68
2.3 État de l'art . . . 69
2.4 AGKM : Asynchronous Gossip K-Means . . . 72
2.4.1 Analyse Théorique . . . 78
2.5 AGEM . . . 82
2.6 Résultats Expérimentaux . . . 84
2.7 Conclusion . . . 91

3 Réduction de dimension **93**
3.1 Réduction de dimension centralisée . . . 94
3.1.1 Réduction de dimension linéaire . . . 94
3.1.2 Réduction de dimension non-linéaire . . . 96
3.1.3 Resolution des problèmes de valeurs propres . . . 100
3.1.4 Pertinence de l'Analyse en Composantes Principales . . . 102
3.2 Réduction de dimension distribuée . . . 103
3.2.1 Information a priori sur les échantillons . . . 103
3.2.2 Décentralisation et asynchronisme . . . 104
3.2.3 Agrégation de modèles *versus* optimisation itérative . . . 105
3.3 Late PCA . . . 105
3.4 AGPCA pour échantillons distribués (ED) . . . 107
3.4.1 Relation duale entre les décompositions de la matrices de covariance et de la matrice de Gram . . . 110
3.5 AGPCA pour coordonnées distribuées (CD) . . . 111
3.5.1 Projection d'observations futures . . . 113
3.6 Analyse Théorique . . . 114
3.7 Résultats Expérimentaux . . . 116
3.7.1 Scénarios ED . . . 119
3.7.2 Scénarios CD . . . 120
3.7.3 Influence de la topologie du réseau . . . 122
3.8 Conclusion . . . 123

4 Classification supervisée et optimisation convexe **125**
4.1 Classification supervisée par Machines à Vecteurs Supports (SVM) . . . 126
4.1.1 Définition du problème . . . 126
4.1.2 Machines à Vecteurs Supports (SVM) . . . 127
4.1.3 Pertinence de la formulation primale . . . 128
4.2 Résolution par descente de gradient . . . 129
4.2.1 Descente de Gradient Stochastique (SGD) . . . 130
4.2.2 Approches par gradient moyenné . . . 131
4.3 Optimisation convexe distribuée . . . 133
4.3.1 Définition du problème . . . 133
4.3.2 Approches géométriques . . . 133

4.3.3 Approches duales . . . 134
4.3.4 Approches primales . . . 135
4.4 Short-Term Averaged Gradient (STAG) . . . 136
4.5 AGSTAG . . . 138
4.5.1 Principe de fonctionnement . . . 138
4.5.2 Outils d'analyse théorique . . . 140
4.5.3 AGSVM . . . 141
4.6 Résultats Experimentaux . . . 143
4.6.1 STAG . . . 143
4.6.2 AGSTAG . . . 145
4.7 Conclusion . . . 146

5 Application à l'indexation multimédia par le contenu 149
5.1 Recherche par le contenu : état de l'art . . . 152
5.1.1 Approches à base de vote . . . 153
5.1.2 Noyaux sur sacs . . . 153
5.1.3 Approches fréquentistes . . . 154
5.1.4 Approches par codage . . . 154
5.1.5 Approches par déviation à un modèle . . . 155
5.1.6 Réseaux convolutifs profonds . . . 156
5.2 Système d'indexation décentralisé et asynchrone proposé . . . 157
5.2.1 Architecture générale . . . 159
5.2.2 Scénario incrémental et limitations . . . 160
5.2.3 Protocole expérimental . . . 160
5.3 Apprentissage de dictionnaire visuel . . . 161
5.4 Classification d'images . . . 164
5.5 Conclusion . . . 165

Conclusion 167

Annexe A : Démonstrations 171

Notations

i, j, k	entiers naturels
x, y, z, t	réels
$\mathbb{N}$	ensemble des naturels
$\mathbb{Z}$	ensemble des entiers relatifs
$\mathbb{R}$	ensemble des réels
$\mathbf{x}, \mathbf{y}, \mathbf{z}$	vecteurs colonne
x_i	i-ème élément du vecteur $\mathbf{x}$
$\mathbf{1}_D$	vecteur de dimension D dont toutes les entrées sont 1 (D omis si non-ambiguë)
$\mathbf{0}_D$	vecteur nul de dimension D (D omis si non-ambiguë)
$\mathbf{e}_i$	i-ème vecteur canonique, nul sauf en $\mathbf{e}_i[i] = 1$ (dimension implicite)
$\|\mathbf{x}\|_2$	norme Euclidienne (ℓ_2) de $\mathbf{x}$
$\mathbf{x} \odot \mathbf{y}$	produit de Hadamard (terme à terme) de $\mathbf{x}$ et $\mathbf{y}$
$\frac{\mathbf{x}}{\mathbf{y}}$	division terme à terme de $\mathbf{x}$ par $\mathbf{y}$
$\mathbf{A}, \mathbf{B}, \mathbf{C}$	matrices
$\mathbf{A}_{(i,j)}$	élément en i-ème ligne et j-ème colonne de la matrice $\mathbf{A}$
$\mathbf{A}^\top$	transposée de la matrice $\mathbf{A}$
$\mathbf{I}_D$	matrice identité $D \times D$ (D omis si non-ambigu)
$\|\mathbf{A}\|$	déterminant de la matrice carrée $\mathbf{A}$
$\text{diag}(\mathbf{A})$	diagonale de la matrice carrée $\mathbf{A}$
$\text{diag}(\mathbf{x})$	matrice diagonale dont la diagonale est $\mathbf{x}$
$\|\mathbf{A}\|_F$	norme de Frobenius de $\mathbf{A}$
$\mathcal{A}, \mathcal{B}, \mathcal{C}$	ensembles
$\|\mathcal{A}\|$	cardinal de l'ensemble $\mathcal{A}$
$\mathcal{V}$	espace vectoriel
$\text{QR}(\mathbf{A})$	décomposition QR économique (thin) de la matrice $\mathbf{A}$
$\text{eig}(\mathbf{A})$	décomposition en vecteurs et valeurs propres de la matrice carrée $\mathbf{A}$
$\text{eig}_q(\mathbf{A})$	décomposition en vecteurs et valeurs propres de $\mathbf{A}$ tronquée aux q paires dominantes
$\mathbf{A} \geq \mathbf{0}$	$\mathbf{A}$ est un matrice non-négative
$\mathbf{A} > \mathbf{0}$	$\mathbf{A}$ est un matrice strictement positive
$\mathbb{P}[X]$	probabilité de l'événement aléatoire X
$\mathbb{E}[X]$	espérance de la variable aléatoire X
$X \sim \mathcal{D}$	variable aléatoire X suivant la loi de probabilité $\mathcal{D}$
$X \sim \mathcal{U}_\mathcal{A}$	la variable aléatoire X est uniformément distribuée sur $\mathcal{A}$
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	loi normale multivariée de moyenne $\boldsymbol{\mu}$ et matrice de covariance $\boldsymbol{\Sigma}$
$\mathbf{g} = \nabla f(\mathbf{x})$	$\mathbf{g}$ est le gradient de la fonction dérivable f en $\mathbf{x}$
$\mathbf{g} \in \partial f(\mathbf{x})$	$\mathbf{g}$ est un sous-gradient de f en $\mathbf{x}$

Introduction

L'augmentation spectaculaire des volumes de données multimédias produits et échangés sur les réseaux, particulièrement catalysée ces vingt dernières années par la croissance d'Internet, des réseaux sociaux, puis des appareils numériques connectés (téléphones mobiles, tablettes, réseaux de capteurs) a ouvert de nouveaux champs applicatifs pour les techniques de traitement de l'information. Sur son versant directement bénéfique, elle a entraîné entre autres une mutation et une maturation accélérée des outils de reconnaissance de formes, de fouille de données et d'extraction de connaissance, notamment via l'introduction d'approches statistiques profitant de la disponibilité de ces grands amas de données. Sur un versant opposé, ce phénomène qualifié populairement de "big data" met à mal les fondations méthodologiques et technologiques des systèmes de traitement de l'information, jusqu'à remettre en question l'hégémonie du modèle de calcul traditionnel hérité de Turing et Von Neumann. Le début du 21$^{\text{ème}}$ siècle a notamment vu se métamorphoser la notion même de valeur associée à un système de traitement de l'information (dans son acception utilitariste aussi bien qu'économique). Autrefois centrée sur l'application (son architecture, sa conception), celle-ci s'est progressivement déportée sur la donnée qui la nourrit (ses dimensions, ses propriétés statistiques, son hétérogénéité, sa représentativité, son évolution dans le temps et l'espace). Savoir interpréter à temps les données entrantes dans toute leur complexité est ainsi devenu un enjeu prioritaire, supplantant le raffinement et la multiplication des fonctionnalités. Sur cette métamorphose s'est développé un domaine de recherche pluri-disciplinaire à part entière, sous l'appellation "science des données" (*data science*).

Limitations du modèle de calcul séquentiel

Dans ce contexte, prendre en compte et exploiter la nature des données est une nécessité fondamentale. Malheureusement, le modèle de calcul traditionnel, qui formalise le traitement comme une séquence d'instructions (*i.e.*, une suite d'opérations atomiques exécutées strictement l'une après l'autre) n'intègre pas une des caractéristique essentielle de ces données : leur *simultanéité*. En effet, les données ne parviennent pas à un système en une unique suite d'informations unitaires. En réalité, un système capture généralement de multiples modalités simultanées, chaque modalité étant elle même formée de multiples dimensions caractérisant un même instant. Les processeurs modernes, largement fondés sur le modèle Turing-Von Neumann dans la lignée de l'architecture de Harvard, contournent ce problème par une augmentation de leur fréquence de fonctionnement. Ainsi si n informations parvenant simultanément à une fréquence f sont traitées une par une à une fréquence nf, la nature séquentielle du processus est transparente. Toutefois, cela repose sur l'hypothèse que la fréquence de fonctionnement d'un processeur peut être augmentée à volonté pour répondre à la multiplication des entrées simultanées, hypothèse

qui se révèle théoriquement et pratiquement erronée. Si durant plusieurs décennies cette limitation a pu être atténuée par l'usage d'artifices techniques divers (augmentation graduelle de la densité d'intégration des circuits, hiérarchisation des accès mémoires par l'usage de caches rapides, etc), le paradigme du calcul séquentiel montre ses limites strictes dans de nombreux cas d'usages récents, sous l'influence de trois principaux facteurs :

- L'augmentation constante du nombre de dimensions simultanées produites par un appareil donné, liée notamment à l'amélioration de la résolution et de la finesse de capture.
- L'explosion de la quantité d'information de nature similaire produite simultanément en chaque point du monde, et présentant donc un intérêt potentiel équivalent pour une application donnée.
- La croissance du nombre d'appareils connectés en réseau et produisant des données de grande dimension, ainsi que la création permanente de nouvelles connections entre appareils.

Lorsqu'un contexte applicatif réunit ces trois facteurs, la méthodologie de calcul doit être repensée pour passer à l'échelle.

Dans cette thèse, nous nous intéressons à une instance typique de ce problème : la recherche multimédia par le contenu (*Content-Based Retrieval*, CBR). Extension de la recherche de documents textuels (*e.g.*, par mots clés dans les moteurs de recherche sur le web), la recherche multimédia par le contenu s'attache à trier, classifier, regrouper des documents multimédias (images, vidéos, sons, modèles 3D ...) sur la base de leur contenu brut, sans utiliser de métadonnées associées (titre, lieu, auteur ...). Dès lors que les trois facteurs de dimension, quantité et connectivité sont réunis, aucune méthode existante ne permet de réaliser ce type de tâche avec une précision convenable sans recourir à de multiples processeurs fonctionnant en parallèle.

De la parallélisation au calcul distribué

La parallélisation du calcul sur de multiples processeurs remonte au début des processeurs eux-mêmes, lorsque la fréquence de fonctionnement était très réduite et ne permettait de traiter que quelques données unitaires par seconde. Lorsque cela était possible, l'ensemble des données à traiter était divisé en lots (*batches*) confiés à une batterie de processeurs s'exécutant de manière indépendante, le défi consistant à produire un résultat agrégé unique à partir des résultats partiels produits par chaque machine.

Avec l'avènement de "l'ère des données", l'augmentation des performances individuelles des processeurs et des capacités de stockage est largement dépassée par la double tendance de multiplication des objets numériques connectés et d'explosion des quantités de données produites par chacun d'eux. Investir un nombre croissant de processeurs semble alors l'unique alternative pour passer à l'échelle. En s'appuyant en premier lieu sur le succès établi des coprocesseurs graphiques massivement parallèles (GPU, *Graphical Processing Units*), de nombreux outils de reconnaissance de formes (*e.g.*, filtrage) ont trouvé une implémentation parallèle performante sur plusieurs centaines de cœurs de calcul (*manycores*). Par la suite se sont développés les réseaux sur puce (*Network-on-Chip*, NoC) où les unités peuvent échanger des données sur des canaux dédiés, sans transiter via un bus central. Ces architectures se rapprochent ainsi progressivement des réseaux conventionnels (LAN, Internet, réseaux mobiles, réseaux de capteurs, ...), certains

prototypes implémentant même un réseau sans fil (RF-NoC). Nous assistons ainsi, au sein même de nos processeurs, à une sortie graduelle du paradigme de programmation séquentielle au profit d'une algorithmique dite "distribuée".

Dans un système dit distribué, chaque unité de calcul (ou noeud) est associée à une mémoire locale distincte et exécute une procédure propre. Ainsi l'algorithmique distribuée se distingue de l'algorithmique parallèle par la nécessité d'expliciter les opérations de communication entre nœuds, qui ne partagent a priori aucune variable en commun (mémoires locales). Jusqu'à présent, la conception d'un algorithme distribué a été largement calquée sur l'algorithmique parallèle (spécification des étapes globales, puis distribution de chaque étape par échange de données entre unités lorsque nécessaire). Les opérations d'échange de données sont souvent restées "discrètes" voire "transparentes". Or, plus un système distribué comporte d'unités de calcul, plus la complexité des opérations locales exécutées en chaque noeud diminue au profit de la quantité d'échanges entre nœuds. En poussant cette idée à l'extrême, chaque noeud n'exécute plus qu'une seule opération locale élémentaire et échange massivement de l'information sur le réseau. L'analyse de performance d'un algorithme massivement distribué ne peut donc faire l'économie d'une étude des interactions entre nœuds au sein du réseau de calcul. D'où la nécessité d'utiliser des outils spécifiques de théorie des réseaux et des systèmes d'interactions dynamiques.

Problèmes intrinsèquement distribués

Si l'algorithmique distribuée est née de la volonté d'augmenter les capacités de calcul et de stockage des systèmes d'information existants, elle a trouvé plus une pertinence nouvelle dans le cas des problèmes intrinsèquement distribués. Par exemple, il est devenu dérisoire de collecter et traiter l'ensemble des images présentes sur l'Internet en un site unique, y compris pour des traitements très simples.

Traiter de telles masses de données implique donc d'accepter leur nature définitivement distribuée. Ces données sont générées, acquises et stockées par une multitudes d'appareils qui sont souvent dotés de capacités de calcul et de communication propres. Exploiter ces capacités locales apparaît donc comme une approche alternative qui permet le passage à l'échelle, mais dont la mise en œuvre relève de l'algorithmique distribuée.

Dans ce contexte, nous devons également accepter que nous ne disposons que d'un contrôle limité sur les unités de calculs, le plus souvent dispersées à l'échelle mondiale. Fiabilité, disponibilité, vitesse, et coopération ne sont alors plus des variables maîtrisées et observables.

Décentralisation et asynchronisme

Dans un système massivement distribué, la dimension "calcul", vue comme une séquence d'opérations locales, s'efface donc irrémédiablement devant la dimension "réseau". La prise en compte des facteurs clés qui influencent le comportement des réseaux acquiert alors un rôle prépondérant. Dans cette thèse, nous nous intéressons à deux de ces facteurs : la décentralisation et l'asynchronisme. Ces deux concepts, qui n'apparaissent pas dans un système séquentiel à processeur unique, jouent un rôle crucial dans un système massivement distribué. Ils se définissent essentiellement par opposition aux pratiques intuitives héritées des systèmes monoprocesseurs, qui considèrent implicitement et en toute logique que la coordination du système est confiée à une seule entité centrale (centralisation) et que l'ordonnancement temporel des opérations est contrôlé par une horloge unique (synchronisation). Si ces deux hypothèses semblent naturelles

dans un système monoprocesseur, elles ne font leur apparition dans un système distribué que par l'ajout de mécanismes spécifiques :

- Pour assurer une coordination centrale des calculs, une unité particulière doit être identifiée comme noeud central (on parle de noeud maître). Ce noeud devient alors un élément critique du réseau : tout dysfonctionnement du noeud maître est directement répercuté sur tous les autres nœuds qui en sont par conception dépendants.

- Deux mécanismes artificiels peuvent introduire une synchronisation dans le réseau : (i) l'usage d'une horloge globale commune, (ii) l'attente mutuelle entre nœuds. Dans le premier cas, on retrouve le paradigme des architectures SIMD où plusieurs opérations (souvent homogènes) sont réalisées dans un même cycle global sur des données différentes. Le deuxième cas est plus subtil, car la synchronisation est seulement locale aux groupes de nœuds qui échangent des données, information qui est portée par les connections elles-mêmes. Si une opération locale à un noeud nécessite la réception d'une information depuis un autre noeud *préalablement* à son exécution, le couple émetteur-récepteur entretient de fait une relation de synchronisation locale. Or, les conséquences d'une synchronisation entre un simple couple de nœuds ne se limitent pas à ce couple, mais font intervenir l'ensemble des relations que ces deux nœuds entretiennent eux-mêmes avec leur voisins. Les effets de cascade résultants révèlent une propriété fondamentale de tout réseau :

> Toute synchronisation locale *explicite*
> influence la synchronisation globale de manière *implicite* (1)

Si les propriétés bénéfiques de la décentralisation sont généralement reconnues dans diverses disciplines, le rôle de l'asynchronisme est quant à lui largement négligé. L'introduction de synchronisations entre couples de nœuds constitue pourtant un risque de conception majeur. Ce risque est aggravé par la nature émergente de la propriété (1), c'est à dire que le phénomène de synchronisation globale ne se manifeste non pas au moment de son implémentation, mais durant son fonctionnement, ceci en fonction d'un contexte d'exécution (la structure du réseau) totalement inconnu *a priori*. Ainsi, une application très performante sur un certain réseau peut soudain devenir extrêmement lente et/ou instable par la simple addition ou suppression d'un lien, d'un noeud ou d'une opération locale.

Ces considérations nous amènent à questionner la notion même de *programmabilité* de tels systèmes, dans un schéma classique de conception des systèmes où l'expression du problème, la formulation d'une solution algorithmique, son implémentation et son instanciation sur une infrastructure quelconque constituent des phases distinctes et traditionnellement étanches.

Dans cette thèse, nous nous fixons donc comme contraintes une décentralisation et un asynchronisme strict. Dans le chapitre 1, nous donnons une définition formelle de ces contraintes et analysons leurs conséquences théoriques. Nous en donnons ici une définition informelle qui nous servira de base de réflexion. Nous définissons la décentralisation par le biais de trois hypothèses :

Définition : Décentralisation

1. Dans un système décentralisé, toutes les unités jouent le même rôle et sont totalement interchangeables.
2. Dans un système décentralisé, toutes les unités n'ont qu'une connaissance locale du réseau, limitée à leur voisinage.
3. Le fonctionnement d'un système décentralisé est indépendant de la structure du réseau. Notamment, il n'est pas impacté par les changements de connectivité en cours de fonctionnement.

De son côté la contrainte d'asynchronisme interdit toute synchronisation des nœuds aussi bien à l'échelle globale qu'à l'échelle locale :

Définition : Asynchronisme

1. Dans un système asynchrone, chaque noeud possède sa propre horloge indépendante, dont la fréquence est inconnue des autres nœuds et peut être différente en chaque noeud.
2. Dans un système asynchrone, aucune opération élémentaire ne doit induire d'attente entre nœuds. Un noeud ne peut donc pas attendre la réception d'une donnée sur une de ses connections. Au contraire toute information provenant d'un de ses liens entrants doit être traitée immédiatement, stockée ou ignorée.
3. Dans un système asynchrone, le fonctionnement d'un noeud est donc guidé non seulement par son horloge interne, mais également par les événements de réception de messages qui peuvent survenir à tout moment de manière concurrente.

Apprentissage statistique large échelle et optimisation distribuée

Au sein des nouvelles "sciences des données", l'apprentissage statistique (ou *machine learning*) occupe une position de premier plan. En effet, l'explosion des volumes de données accessibles a conduit à l'apparition de modèles de traitement de l'information dont les paramètres sont automatiquement estimés à partir des données d'entrée. Ces quantités de données se révèlent aujourd'hui suffisamment significatives pour que ces modèles statistiques surpassent les performances de modèles spécifiquement conçus par des experts humains. A la rigidité et la spécificité de ces modèles "faits-main" (*handcrafted models*) s'oppose donc la généricité de ces modèles "appris", qui permet leur application à des problèmes souvent inaccessibles à l'expertise humaine (prédiction, classification, régression, compression, catégorisation, détection, recommandation,...).

De nombreuses applications industrielles désormais matures font appel à des méthodes d'apprentissage statistique, souvent couplées à des systèmes de calcul distribué. Ces méthodes correspondent le plus souvent à la maximisation d'un critère de qualité de modèle ou la minimisation d'une erreur de prédiction, sur la base d'un grand ensemble de données d'entraînement (*training dataset*) supposé représentatif du phénomène observé. Elles appartiennent ainsi à la famille plus large des techniques d'optimisation numérique. En dépit d'une histoire longue et

prolifique des techniques d'optimisation distribuées, le cadre algorithmique dans lequel elles ont été développées est poussé à ses limites par la quantité et la dimensionnalité des données investies par les applications contemporaines de l'apprentissage statistique.

Par exemple, aucun moteur de recherche actuel ne permet d'indexer l'ensemble des contenus non-textuels (*e.g.* multimédias) présents sur Internet avec une précision acceptable, ceci malgré la croissance permanente de la performance des calculateurs et des énormes moyens financiers investis par les grandes firmes dans la distribution des calculs. De plus, le fourmillement d'études empiriques récentes (dans le cadre aussi bien de compétitions académiques que d'applications industrielles pionnières) suggèrent une tendance de plus en plus claire : les systèmes les plus performants sont ceux qui exploitent la plus grande quantité de données. En particulier, les approches statistiques n'offrent de bons résultats que lorsque la quantité de données fournie est significative. L'accès à d'immenses masses de données est donc une condition *nécessaire* au succès de ces modèles. Cette propriété met en évidence la deuxième facette du phénomène "big data" : l'augmentation de la masse de données disponible est à la fois un phénomène *subi* et *désiré*.

Un dernier aspect important est que l'information pertinente est produite en continu à un rythme croissant. Le temps de calcul devient donc un critère prédominant. Pour contrer cette limitation, les méthodes d'optimisation distribuées sont déployées sur des réseaux toujours plus grands. Analyser et prédire les phénomènes inhérents aux larges réseaux apparaît donc comme une préoccupation indispensable.

Dans cette thèse nous défendons que la décentralisation et l'asynchronisme, tels que définis plus haut, sont une des clés du passage à l'échelle des méthodes d'apprentissage statistique. Les méthodologies algorithmiques classiques doivent alors être repensées pour tenir compte des contraintes associées à la décentralisation et l'asynchronisme.

Mentionnons enfin que nos travaux s'inscrivent dans un contexte sociétal de préoccupation croissante envers la centralisation de l'information numérique par une poignée d'acteurs influents. Ces préoccupations concernent principalement le respect de la vie privée et de la confidentialité. De nombreuses institutions se sont prononcées pour une moralisation et une régulation de ces pratiques via des mesures éthiques et/ou juridiques. Nous pensons que l'approche décentralisée présentée dans cette thèse, complémentaire, apporte une réponse technique à un problème d'abord d'ordre *systémique*.

Organisation du manuscrit

Cette thèse est articulée en trois parties :

- Dans le chapitre 1, nous présentons le paradigme théorique sur lequel s'appuient nos contributions algorithmiques, à savoir les protocoles Gossip.
- Dans les chapitres 2,3 et 4, nous introduisons trois méthodes d'apprentissage originales décentralisées et asynchrones, respectivement dédiées à la catégorisation non supervisée, la réduction de dimension et l'optimisation convexe. Tous les algorithmes présentés dans ces chapitres sont construit sur le protocole Gossip défini dans le chapitre 1.
- Dans le chapitre 5, nous illustrons l'intérêt applicatif des algorithmes proposés en évaluant leur pertinence dans une chaîne d'indexation multimédia combinant plusieurs méthodes d'apprentissage.

Publications

Les contributions présentées dans ce manuscrit ont fait l'objet de diverses publications dans des revues et conférences nationales et internationales, listées ci-dessous :

Revues

- Jérôme Fellus, David Picard et Philippe-Henri Gosselin. *Asynchronous Gossip Principal Components Analysis*. in Neurocomputing, 2015.
- Jérôme Fellus, David Picard et Philippe-Henri Gosselin. *Indexation multimédia par dictionnaires visuels en environnement décentralisé : Une approche par protocoles Gossip*. in Traitement du Signal, 2015.

Conférences

- Jérôme Fellus, David Picard et Philippe-Henri Gosselin. *Asynchronous decentralized convex optimization through short-term gradient averaging*. in European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN). Avril 2015, Bruges, Belgique
- Jérôme Fellus, David Picard et Philippe-Henri Gosselin. *Dimensionality reduction in decentralized networks by Gossip aggregation of principal components analyzers*. in European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN). Avril 2014, Bruges, Belgique.
- Jérôme Fellus, David Picard et Philippe-Henri Gosselin. *Decentralized K-means using randomized Gossip protocols for clustering large datasets*. in ICDM Workshop on Knowledge Discovery Using Cloud and Distributed Computing Platforms, Dec 2013, Dallas, Texas, États-Unis.
- Jérôme Fellus, David Picard et Philippe-Henri Gosselin. *Calcul décentralisé de dictionnaires visuels pour l'indexation multimédia dans les bases de données réparties sur les réseaux*. in Orasis, Congrès des jeunes chercheurs en vision par ordinateur (ORASIS), Juin 2013, Cluny, France.

CHAPITRE 1

Fondations théoriques : Protocoles Gossip

Ce chapitre présente les concepts théoriques qui servent de base aux contributions algorithmiques des chapitres suivants. Dans un premier temps, il introduit les bases de l'apprentissage statistique (*machine learning*) et de l'optimisation numérique et installe le cadre spécifique de l'optimisation distribuée. La section 1.3 s'intéresse au calcul distribué de moyennes, composante essentielle des méthodes développées dans cette thèse. Les protocoles par consensus, paradigme itératif qui permet le calcul décentralisé de moyennes à travers un réseau de connectivité arbitraire, sont définis dans la section 1.4. Les protocoles Gossip, extension stochastique des protocoles par consensus, sont introduits en section 1.5. La section 1.6 lève les contraintes de synchronisation inhérentes aux protocoles Gossip classiques (dits symétriques) en proposant et en analysant des protocoles Gossip *asynchrones*. Nous y présentons trois contributions originales :

- Un protocole Gossip asynchrone basé sur des matrices de communication seulement contraintes en espérance (les matrices de communication aléatoires ne sont pas nécessairement symétriques, seule leur espérance doit être doublement stochastique).

- Une analyse de convergence non-asymptotique des protocoles Gossip asynchrones de type *Sum-Weight*, qui raffine notre connaissance théorique des processus stochastiques correspondants et offre un outil de prédiction du comportement précieux pour le développement d'algorithmes Gossip asynchrones plus complexes (*e.g.*, non-linéaires) tels que ceux présentés dans cette thèse.

- Un protocole Gossip totalement asynchrone nommé AGAVG (Asynchronous Gossip Averaging) qui forme la composante de base des algorithmes développés dans les chapitres suivants.

Les sections 1.7 et 1.8 étudient respectivement des extensions non-linéaires de nos protocoles Gossip asynchrones et le comportement de ceux-ci sur des réseaux dont la topologie évolue dans le temps.

La section 1.9 présente finalement la synthèse de ces bases théoriques.

1.1 Apprentissage statistique et optimisation

Une large majorité de problèmes d'apprentissage statistique se présente sous la forme d'une fonction d'erreur à minimiser ou une fonction de vraisemblance à maximiser, qui qualifient la fidélité d'un modèle vis-à-vis d'un phénomène d'intérêt supposé inconnu. Ce phénomène est représenté par une variable aléatoire X, dont on ne possède qu'un ensemble fini de n réalisations (ou observations) $\{x_i\}_{i=1}^n$. L'objectif consiste généralement à trouver le modèle $\boldsymbol{\theta}^\star$ qui minimise l'espérance de la fonction de coût sur toutes les réalisations possibles de X, parmi une famille de modèles indexée par un ensemble de paramètres $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^\star \stackrel{\text{def}}{=} \arg\min_{\boldsymbol{\theta}} \mathbb{E}_X[J(X, \boldsymbol{\theta})] \tag{1.1}$$

Dans le cas où l'objectif est de maximiser une vraisemblance $\mathcal{L}$, il suffit de considérer $J(X, \boldsymbol{\theta}) = -\mathcal{L}(X|\boldsymbol{\theta})$ pour obtenir un problème de minimisation et retomber sur (1.1). En guise d'illustration, le problème de régression consiste à estimer une fonction f inconnue liant deux variables X et Y (*i.e.*, $Y = f(X)$), étant donné un ensemble d'échantillons $(x_i, y_i)_{i=1}^n$. Dans ce cadre, la régression au sens des moindres carrés (*Least Squares*, LS) utilise comme fonction de coût l'erreur quadratique entre la valeur prédite par un certain modèle $\hat{f}_{\boldsymbol{\theta}}(X)$ et la valeur réelle $f(X) = Y$:

$$J(X, \boldsymbol{\theta}) \stackrel{\text{def}}{=} \Big[\hat{f}_{\boldsymbol{\theta}}(X) - f(X)\Big]^2 \tag{1.2}$$

Par exemple, la régression linéaire considère les modèles du type $\hat{f}_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}$ où $\mathbf{x}$ et $\boldsymbol{\theta}$ sont des vecteurs de $\mathbb{R}^D$. L'objectif (1.1) peut alors réécrit comme

$$\boldsymbol{\theta}^\star \stackrel{\text{def}}{=} \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^D} \mathbb{E}_{\mathbf{x}}\big[f(\mathbf{x}) - \boldsymbol{\theta}^\top \mathbf{x}\big]^2 \tag{1.3}$$

Puisqu'en pratique on ne dispose que d'un ensemble fini d'échantillons $(\mathbf{x}_i, y_i)_{i=1}^n$, on estime $\boldsymbol{\theta}^\star$ en remplaçant l'espérance inconnue dans (1.3) par la moyenne empirique de la fonction d'erreur sur le jeu d'échantillons, appelée risque empirique [241]) :

$$\boldsymbol{\theta}^\star \stackrel{\text{def}}{=} \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^D} \frac{1}{n} \sum_{i=1}^n \big[y_i - \boldsymbol{\theta}^\top \mathbf{x}_i\big]^2 \tag{1.4}$$

D'après la loi des grands nombres, si les échantillons sont effectivement issus d'un même processus (*i.e.*, sont identiquement distribués), plus le nombre d'échantillons n est grand plus l'approximation (1.6) se rapproche du véritable optimum (1.3) :

$$\lim_{n \to +\infty} \mathbb{P}\left[\frac{1}{n} \sum_{i=1}^n \big[y_i - \hat{f}_{\boldsymbol{\theta}}(\mathbf{x}_i)\big]^2 = \mathbb{E}_{\mathbf{x}}\big[f(\mathbf{x}) - \hat{f}_{\boldsymbol{\theta}}(\mathbf{x})\big]\right] = 1 \tag{1.5}$$

En l'occurrence, la régression linéaire au sens des moindres carrés trouve une solution analytique (*i.e.*, directement calculable par une formule explicite) [178]. En notant $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$ et $\mathbf{y} = (y_1, \ldots, y_n)^\top$, on a en effet

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \big\|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\big\|_2^2 = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \tag{1.6}$$

Dans ce cas spécifique, il suffit donc de calculer l'inverse, lorsqu'elle existe, de $\mathbf{X}^\top\mathbf{X}$ (appelée matrice de Gram).

Toutefois, un grand nombre de problèmes d'apprentissage statistique n'ont pas de solution analytique en forme close telle que (1.6). C'est de manière générale le cas des problèmes non-linéaires qui peuvent avoir de multiples optimums locaux sans offrir de formule explicite pour l'optimum global. Pour ces familles de problèmes, on doit recourir à des méthodes d'optimisation itératives.

Une approche itérative typique consiste à mettre à jour les paramètres $\boldsymbol{\theta}$ pas à pas en approximant J au voisinage de $\boldsymbol{\theta}$ (*e.g.*, via son développement de Taylor) de sorte à ce que la résolution du problème approximé à chaque pas soit "facile". Autrement dit, à partir de paramètres courants $\boldsymbol{\theta}(t)$, on cherche de nouveaux paramètres $\boldsymbol{\theta}(t+1)$ calculables par une expression analytique et tels que $J(\boldsymbol{\theta}(t+1)) \leq J(\boldsymbol{\theta})$. La convergence vers un minimum de J est alors garantie (mais celui-ci peut n'être qu'un minimum local pour un problème non-convexe).

1.2 Optimisation distribuée

Lorsque le nombre d'échantillons n et leur dimension D restent raisonnables (quelques milliers), la plupart des problèmes d'optimisation peuvent être résolus efficacement par des algorithmes séquentiels qui ne manipulent qu'un échantillon à la fois et supposent que tous les échantillons sont accessibles depuis une mémoire centrale unique.

Or, pour les raisons à la fois subies et désirées évoquées en introduction de cette thèse, les méthodes d'apprentissage statistique sont amenées à considérer des ensembles d'entraînement de très grande taille et de très grande dimension, entraînant trois conséquences :

- L'ensemble d'entraînement ne peut plus être stocké dans une unique mémoire centrale, mais doit être réparti sur plusieurs sites de stockage.
- Le nombre d'opérations à réaliser pour obtenir le résultat augmentant avec n et D, le temps de calcul explose.
- La taille des données intermédiaires nécessaires aux calculs, éventuellement liées à n et/ou D, dépassent la capacité mémoire d'une machine seule.

Ces limitations ont conduit à la formulation d'algorithmes distribués pour la résolution de problèmes d'optimisation large échelle. Ces algorithmes font généralement l'hypothèse que les données sont dispersées sur un ensemble de nœuds de stockage, et que l'on dispose de multiples nœuds de calcul, chaque noeud de calcul étant couplé à une ou plusieurs unités de stockage et pouvant au besoin échanger des données avec certains nœuds voisins par le biais d'un protocole de communication. Cette configuration, spécifiée indépendamment de l'algorithme lui-même, correspond donc à la définition d'un réseau de N noeuds connectées selon un certain graphe orienté $\mathcal{G}$. Chaque noeud est doté de capacités de calcul et d'une mémoire locales et héberge un sous-ensemble $\mathbf{X}_i$, $1 \leq i \leq N$ des données d'entraînement $\mathbf{X}$. On suppose de plus que $\bigcup_i \mathbf{X}_i = \mathbf{X}$ et que $\forall i, j, \mathbf{X}_i \cap \mathbf{X}_j = \emptyset$, c'est à dire que $\{\mathbf{X}_i\}_{i=1}^N$ est une partition de $\mathbf{X}$. Un algorithme distribué doit alors garantir l'obtention du résultat *quels que soient* le graphe $\mathcal{G}$ et l'allocation des données $\{\mathbf{X}_i\}$. On cherche en effet à ce que l'influence du réseau soit la plus transparente possible.

1.2.1 Modèle de temps

Formellement, un processus de calcul distribué est un système dynamique et doit être analysé en tant que tel. La définition du modèle de temps dans lequel ce système dynamique évolue est une hypothèse importante mais souvent implicite.

Modèle de temps globalement synchrone

De nombreuses méthodes de calcul distribué supposent qu'à chaque pas de temps $t \in \mathbb{N}$, chaque noeud effectue une certaine opération. Cette hypothèse induit de fait une synchronisation globale, dont la mise en œuvre pratique requiert *(i)* une horloge commune à tous les nœuds qui, à chaque pas de temps, déclenche une opération et une seule en chaque noeud, *(ii)* que l'instant $t + 1$ ne survient que lorsque toutes les opérations au temps t sont terminées. Une conséquence directe est que les opérations en tout noeud s'effectuent aussi lentement que le noeud le plus lent. Plus grave, si une communication entre nœuds est nécessaire à un instant donné, les nœuds non impliqués dans cette opérations doivent attendre que les données soient transmises avant de procéder à leur propres calculs et/ou échanges. Une méthode distribuée s'appuyant sur un modèle de temps globalement synchrone est donc fondamentalement sous-efficiente. La principale justification de ce type d'approches synchrones réside dans la facilitation des preuves théoriques de convergence et l'analyse asymptotique, qui peut alors s'appuyer sur un formalisme purement itératif où l'évolution du système est globalement maîtrisée à chaque étape.

Modèle de temps partiellement asynchrone

Au contraire, un système distribué privé d'horloge globale est intrinsèquement asynchrone. Chaque noeud est guidé par sa propre horloge qui peut fonctionner à une fréquence très différente selon le nœud considéré. Dans un modèle de temps asynchrone, on considère un temps réel $t \in \mathbb{R}$ fondamentalement continu, au sein duquel surviennent des événements de "tics" d'horloges épars. Une approche classique est de modéliser ces horloges entières comme des processus stochastique de comptage ponctuels [39]. Formellement, les horloges sont définies comme des processus de Poisson superposés et indépendants. Dans le cas où les fréquences d'horloge des nœuds sont identiques, disons égales à λ, le processus global d'activation des horloges à travers le réseau est lui-même un processus de Poisson de taux $N\lambda$.

Il est alors commode de supposer que l'état d'un noeud évolue instantanément au moment où son horloge s'active. On peut ainsi assimiler indifféremment t au temps réel ($t \in \mathbb{R}$) ou au processus de comptage global ($t \in \mathbb{N}$) sans perte de généralité.

En réalité, une opération ne s'effectue jamais instantanément. De plus, un noeud ne peut exécuter plusieurs opérations simultanément, en tant qu'unité de calcul séquentielle. L'éventualité qu'un noeud soit "occupé" ne doit donc pas être ignoré dans l'analyse. Ainsi, les horloges locales ne peuvent pas être rigoureusement modélisées par des processus de Poisson indépendants (qui associent une probabilité non-nulle à tout intervalle non-nul). Lorsqu'un noeud envoie un message à un noeud occupé, il nous faut donc expliciter le comportement du couple émetteur-récepteur.

Dans un modèle partiellement asynchrone [39], on suppose implicitement que l'émetteur attend que le récepteur ait reçu son message. Cette hypothèse a deux conséquences néfastes : *(i)* l'analyse théorique ne tient pas compte des effets émergents de cette synchronisation locale

(*e.g.*, synchronisation globale implicite), *(ii)* l'analyse ne prédit pas le comportement du système lorsque cette synchronisation locale ne peut pas être techniquement honorée (délais réseau, pertes de messages).

Modèle de temps totalement asynchrone

Afin d'ôter toute notion cachée de synchronisation ou attente entre nœuds susceptible d'altérer de manière implicite l'évolution temporelle de leur état, les modèles totalement asynchrones reposent sur deux hypothèses architecturales :

1. Les opérations locales sont atomiques. L'horloge locale à un noeud ne peut s'activer que si celui-ci a terminé son opération précédente.

2. Lorsqu'une opération en un noeud i implique une transmission d'information vers un noeud j, l'état du noeud j n'est pas mis à jour immédiatement, mais à la prochaine activation de sa propre horloge. Pour ce faire on suppose l'existence d'une mémoire à court terme locale à j, capable de stocker la donnée entrante à l'instant où elle lui parvient (éventuellement après un délai inconnu de transmission). Cette mémoire est supposée de capacité finie (les messages dépassant cette capacité sont supposés perdus). Ainsi, l'envoi du message est asservi à l'horloge de i, tandis que sa réception et son traitement est asservi à l'horloge de j. Il y a donc désynchronisation des échanges entre nœuds (au prix d'un surcoût mémoire supposé tolérable).

D'un point de vue théorique, l'ensemble des mécanismes de désynchronisation par mémoire à court terme associés à chaque noeud s'apparente à un système de files d'attente, dont l'analyse sort du cadre de cette thèse, mais pour lequel il existe de nombreux résultats théoriques déterminant les conditions de stabilité [263]. Dans cette thèse, nous supposons que l'infrastructure réseau est construite de sorte à respecter ces conditions de stabilité.

Toutes les méthodes présentées dans cette thèse s'appuient sur ce modèle de temps totalement asynchrone.

1.2.2 Optimisation par consensus

Dès lors que chaque noeud possède ses propres capacités de calcul, sa propre mémoire locale et son propre jeu d'échantillons, une approche naturelle pour résoudre un problème d'optimisation distribué est de confier à chaque noeud la résolution du sous problème associé à son jeu de données local.

Par exemple, si l'on étend le problème de la régression linéaire par moindre carrés (1.6) au cas où $f : \mathbb{R}^D \mapsto \mathbb{R}^m$, c'est à dire où $\mathbf{y} = f(\mathbf{x})$ est un vecteur de dimension m, les paramètres à optimiser deviennent une matrice $\boldsymbol{\Theta} \in \mathbb{R}^{D\times m}$, et on a

$$\boldsymbol{\Theta}^\star \overset{\text{def}}{=} \underset{\boldsymbol{\Theta}\in\mathbb{R}^{m\times D}}{\arg\min} \left\|\mathbf{Y} - \mathbf{X}\boldsymbol{\Theta}\right\|_F^2, \tag{1.7}$$

où $\mathbf{Y} \in \mathbb{R}^{n\times m}$ et $\mathbf{X} \in \mathbb{R}^{n\times D}$. La résolution de (1.7) peut être distribuée en répartissant les m colonnes de $\mathbf{Y}$ sur m nœuds, en leur affectant chacun une copie de $\mathbf{X}$, et en confiant à chaque noeud i la résolution du sous-problème

$$\boldsymbol{\Theta}^\star_{(i,\cdot)} \overset{\text{def}}{=} \underset{\boldsymbol{\theta}\in\mathbb{R}^{D}}{\arg\min} \left\|\mathbf{Y}_{(\cdot,i)} - \mathbf{X}\boldsymbol{\theta}\right\|_F^2, \tag{1.8}$$

Remarquons que pour obtenir $\boldsymbol{\Theta}^\star$, il n'est pas nécessaire d'échanger d'information entre les nœuds, puisque chacun en calcule une ligne distincte i et possède les données nécessaires $\mathbf{Y}_{(\cdot,i)}$ et $\mathbf{X}$. De manière générale, un problème ou chaque noeud peut calculer une composante distincte du résultat sans échanger d'information avec ses voisins est qualifié de "*séparable*". Ces problèmes très simples ont très tôt trouvé des solutions efficaces, logicielles comme matérielles (*e.g.*, implémentations sur GPU).

Si au lieu de distribuer les m colonnes de $\mathbf{Y}$ on distribue les n lignes de $\mathbf{Y}$ et $\mathbf{X}$ sur n nœuds, (*i.e.*, on affecte un échantillon $(\mathbf{x}_i, \mathbf{y}_i)$ à chaque noeud i), le sous-problème associé aux données locales à un noeud i devient

$$\boldsymbol{\Theta}^{(i)} \stackrel{\text{def}}{=} \underset{\boldsymbol{\Theta}\in\mathbb{R}^{D\times m}}{\arg\min} \left\|\mathbf{y}_i - \mathbf{x}_i\boldsymbol{\Theta}\right\|_2^2, \tag{1.9}$$

On constate que contrairement à (1.8), on ne peut pas trivialement obtenir $\boldsymbol{\Theta}^\star$ à partir des résultats locaux $\boldsymbol{\Theta}^{(i)}$ obtenus en chaque noeud. Le problème global ne peut donc pas être réduit à la résolution de sous-problèmes locaux indépendants : on qualifie le problème de "*non-séparable*".

Pour s'assurer que les nœuds résolvent des problèmes locaux équivalents au problème global grâce à leurs seuls échantillons locaux, on doit formuler des contraintes explicites sur les modèles locaux $\boldsymbol{\Theta}^{(i)}$:

$$\begin{aligned}(\boldsymbol{\Theta}^{(1)}, \dots, \boldsymbol{\Theta}^{(n)}) \stackrel{\text{def}}{=} \underset{(\hat{\boldsymbol{\Theta}}_i)_{i=1}^n}{\arg\min} \sum_{i=1}^{n} \left\|\mathbf{y}_i - \mathbf{x}_i\hat{\boldsymbol{\Theta}}_i\right\|_2^2 \\ \text{sous la contrainte} \quad \boldsymbol{\Theta}^{(1)} = \dots = \boldsymbol{\Theta}^{(n)}\end{aligned} \tag{1.10}$$

On s'aperçoit sans difficulté que (1.10) est rigoureusement équivalente à (1.7) et que $\forall i, \boldsymbol{\Theta}^{(i)} = \boldsymbol{\Theta}^\star$. Chaque noeud i obtient donc une instance locale du modèle optimal. Cela nous a néanmoins obligés à ajouter une contrainte d'égalité sans laquelle (1.10) reviendrait simplement à (1.9), *i.e.*, les nœuds n'optimiseraient que leur fonction objective locale sans tenir compte de l'objectif global. Cette contrainte est nommée "*contrainte de consensus*", dans le sens où elle force les nœuds à trouver un compromis entre leurs objectifs respectifs sous la forme d'une estimée commune à tous.

De manière générale, un problème d'*optimisation par consensus* a la forme suivante :

$$\begin{aligned}\text{minimiser } J(\mathbf{X}, \boldsymbol{\theta}) = \sum_{i=1}^{N} J_i(\mathbf{X}_i, \boldsymbol{\theta}_i) \\ \text{sous la contrainte} \quad \boldsymbol{\theta}_1 = \dots = \boldsymbol{\theta}_N\end{aligned} \tag{1.11}$$

où N est le nombre de nœuds du réseau, et $\mathbf{X}_i$, $\boldsymbol{\theta}_i$ et J_i sont respectivement le jeu de données local, les paramètres du modèle local et la fonction de coût locale au noeud i. Chaque noeud s'efforce de minimiser son terme $J_i(\mathbf{X}_i, \boldsymbol{\theta}_i)$ indépendamment des autres, tandis que la contrainte d'égalité force le consensus. Par le biais de ces deux "forces" contradictoires, le système distribué trouve le minimum de la fonction de coût globale $J(X, \boldsymbol{\theta}) \equiv \sum_i^N J_i(\mathbf{X}_i, \boldsymbol{\theta})$.

L'optimisation par consensus, notamment popularisée par son usage intensif dans la coordination de systèmes multi-agents et l'allocation de ressource en théorie des réseaux, a trouvé de nombreuses applications dans des domaines aussi variés que la météorologie, l'économie, la biologie cellulaire, ou les systèmes de transports.

Une propriété importante et souvent peu mise en avant du problème général de consensus (1.11) est que la composante *non-séparable* de ce problème est *linéaire*. Dans cette thèse, nous exploitons cette propriété pour formuler des algorithmes d'optimisation itérative dont les opérations non-linéaires sont toujours strictement locales et où les échanges entre nœuds n'impliquent que des combinaisons linéaires.

Intuitivement, pour peu que l'on dispose d'une fonction f localement calculable en tout noeud i telle que $J_i(\mathbf{X}_i, f(\boldsymbol{\theta}_i(t))) \leq J_i(\mathbf{X}_i, \boldsymbol{\theta}_i(t))$, alors la règle de mise à jour suivante

$$\boldsymbol{\theta}_i(t+1) = \frac{1}{N} f(\boldsymbol{\theta}_i(t)) + \frac{1}{N} \sum_{j \neq i}^{N} \boldsymbol{\theta}_j(t) \tag{1.12}$$

attirera tous les $\boldsymbol{\theta}_i(t)$ vers une solution de (1.11). Le premier terme de (1.12) est localement calculable en i, tandis que le deuxième est la moyenne des estimées des nœuds voisins. La résolution d'un problème d'optimisation distribuée se trouve donc réduite au calcul d'une fonction localement calculable f et d'une moyenne distribuée. Dans la section suivante, nous nous concentrons sur le calcul d'une telle moyenne distribuée.

1.3 Calcul distribué de moyennes

Dans cette section, nous considérons le calcul distribué de moyennes, éventuellement pondérées. Soient N estimées $\{x_i\}_{i=1}^n$ appartenant à un même espace vectoriel $\mathcal{V}$ quelconque et N coefficients réels $\{w_i\}_{i=1}^n$, chaque couple (x_i, w_i) étant hébergé par un noeud i du réseau. Notre objectif est de fournir *en tout noeud* la moyenne $\bar{x}$ définie par

$$\bar{x} = \frac{\sum_{i=1}^{N} w_i x_i}{\sum_{i=1}^{N} w_i} \tag{1.13}$$

Approche 1 : Agrégation centralisée

Une première solution à ce problème est de transmettre tous les couples (x_i, w_i) à un noeud déterminé qui, en possession de toutes les données nécessaires peut calculer directement le résultat et le renvoyer à tous les autres nœuds. Ce noeud particulier est alors qualifié de "maître". L'avantage évident de cette approche est que chaque noeud n'envoie qu'un seul message contenant son estimée et reçoit un seul message contenant le résultat $\bar{x}$. Cette solution est donc optimale en coût de communications pour l'immense majorité des nœuds. Le noeud maître doit quant à lui recevoir, agréger et renvoyer N estimées. De par sa simplicité, cette approche canonique est très répandue dans les systèmes de calcul à architecture clients-serveur, dans de nombreuses solution de calcul en nuage (*cloud computing*, *e.g.* MapReduce) ou en tant que primitive standard (opération de réduction dans les standards MPI, OpenMP, etc).

Selon notre modèle de temps asynchrone, à chaque cycle d'horloge locale, le noeud maître n'effectue qu'une seule opération (traitement d'un message entrant, calcul local ou envoi d'un message). Bien que tous les autres nœuds n'aient besoin que d'un cycle d'horloge pour réaliser leur tâche, le noeud maître nécessite de son côté $\mathcal{O}(N)$ opérations, puisque N messages lui parviennent. Ceci soulève de sérieuses limitations :

- Soit le noeud maître est N fois plus rapide que les autres, soit le calcul global est N fois plus lent que le noeud le plus rapide.

- Le temps de calcul de la moyenne croît linéairement avec la taille du réseau.
- Le processus est donc par construction globalement synchrone : chaque noeud doit attendre le résultat envoyé par le noeud maître avant d'avoir une quelconque estimée de la moyenne.
- En cas de dysfonctionnement du noeud maître, l'ensemble du réseau est impacté. Notamment, si le noeud maître quitte le réseau, l'ensemble du calcul échoue.
- Tous les nœuds doivent être connectés au noeud maître.

Ces contraintes fortes sur le noeud maître impliquent que le succès de l'algorithme est étroitement lié à l'infrastructure réseau. Dans notre contexte, où la nature du réseau sur lequel est déployé la solution est supposée inconnue *a priori*, une telle approche par agrégation centralisée est donc inadaptée.

Approche 2 : Agrégation incrémentale asynchrone

Grâce à l'associativité de l'addition, on peut lever la contrainte de synchronisation globale en calculant la somme de manière incrémentale. Plus précisément, tout noeud i peut à tout moment décider d'envoyer sa valeur locale au noeud maître, qui l'intègre à son estimée courante de la moyenne et envoie cette dernière à tous les nœuds à chaque incrément :

$$\hat{s}(t+1) = \hat{s}(t) + x_i \quad \hat{w}(t+1) = \hat{w}(t) + w_i \quad \bar{x}(t+1) = \frac{\hat{s}(t+1)}{\hat{w}(t+1)} \tag{1.14}$$

Cette approche centralisée asynchrone (également qualifiée d'approche *blackboard* car tous les nœuds "écrivent" et "lisent" sur un même "tableau noir" sans coordination entre eux) est notamment utilisée dans [3] puis dans [66] pour l'entraînement de réseaux de neurones profonds (*Deep Neural Networks*).

Remarquons toutefois que l'asynchronisme est limité par l'engorgement du noeud maître, qui ne peut intégrer qu'une seule estimée à chaque cycle de son horloge locale. Ainsi, bien que les nœuds puissent obtenir une moyenne approximative dès les premiers cycles, il faut toujours un temps linéaire en N pour obtenir la moyenne exacte.

Approche 3 : Agrégation le long d'un arbre couvrant

Une approche alternative, qui exploite également l'associativité de l'addition, profite de la structure arborescente du réseau pour effectuer des agrégations partielles en chaque noeud et en propageant ces résultats intermédiaires le long d'un arbre couvrant le réseau. Les nœuds parents calculent la moyenne des estimées de leurs nœuds fils, transmettent le résultat à leurs propres parents qui en calculent la moyenne et ainsi de suite, jusqu'à la racine de l'arbre qui obtient finalement la moyenne globale. Ce résultat est alors propagé aux nœuds dans le sens inverse. Les avantages de cette approche par arbre couvrant sont multiples :

- Aucun noeud n'effectue plus de d opérations où d est son nombre de voisins.
- Le nombre de messages m échangés à travers le réseau est minimal : $m = 2(N-1)$.

- Quel que soit la structure du réseau, on peut toujours en construire un arbre couvrant dès lors que le réseau est connexe.

Ces avantages ont entraîné l'utilisation d'arbres couvrants dans de nombreux systèmes d'apprentissage [2, 180]. Malgré cette optimalité en termes de calculs locaux et de communications, les principaux défauts de cette approche arborescente sont sa robustesse et son caractère globalement synchrone :

- Si un noeud parent quitte brutalement le réseau, les contributions de l'ensemble de ses descendants sont perdues.
- De même, si un lien est rompu ou introduit des erreurs, toute confiance dans les contributions de ses descendants est perdue.
- Les liens redondants formant des cycles ne sont pas exploités.
- Les nœuds terminaux (*i.e.*, sans fils) sont désavantagés car ils ne reçoivent une estimée de la moyenne qu'après $2h$ opérations séquentielles, où h est la hauteur de l'arbre. Dans un réseau correctement équilibré cette hauteur est logarithmique donc faible devant N. Mais elle devient problématique dans les réseaux moins biens conditionnés (anneaux, grilles, architectures systoliques [261]).

Approche 4 : Diffusion entre voisins

Pour lever les limitations des approches arborescentes, une solution est d'exploiter l'ensemble des liens disponibles, en forcant les nœuds à communiquer avec *tous* leurs voisins. Ainsi, dans les protocoles par diffusion (*broadcast*), les nœuds adressent le même message simultanément à tous leurs voisins, sans distinction. Par exemple, dans un réseau à connectivité complète (clique), où chaque noeud est connecté à tous les autres, si chaque noeud envoie son estimée à tous ses voisins par diffusion, tous les nœuds reçoivent les termes nécessaires et peuvent donc calculer $\bar{x}$ localement, chacun de son coté.

Par commodité de comparaison avec les approches présentées dans la suite de cette section, ce protocole par diffusion peut être écrit sous forme matricielle. En notant respectivement $\mathbf{x} = (x_1, \dots, x_N)^\top$, $\mathbf{w} = (w_1, \dots, w_N)^\top$ et $\hat{\mathbf{x}} = (\hat{x}_1, \dots, \hat{x}_N)^\top$ les vecteurs colonne regroupant les estimées locales initiales, les poids initiaux et les estimations de $\bar{x}$ en chaque noeud i, on a

$$\hat{\mathbf{x}}^\top = \frac{\hat{\mathbf{s}}^\top}{\hat{\mathbf{w}}^\top} \quad \text{avec} \quad \begin{cases} \hat{\mathbf{s}}^\top = \left(\sum_{i=1}^{N} w_i x_i \right) \mathbf{1}^\top = (\mathbf{w}^\top \odot \mathbf{x}^\top) \mathbf{1}\mathbf{1}^\top \\ \hat{\mathbf{w}}^\top = \left(\sum_{i=1}^{N} w_i \right) \mathbf{1}^\top = \mathbf{w}^\top \mathbf{1}\mathbf{1}^\top \end{cases}, \tag{1.15}$$

où $\frac{\mathbf{a}}{\mathbf{b}}$ désigne la division composante à composante du vecteur $\mathbf{a}$ par le vecteur $\mathbf{b}$ et $\mathbf{a} \odot \mathbf{b}$ leur produit de Hadamard (composante à composante).

Le principal avantage du protocole par diffusion provient de son caractère décentralisé. En effet, contrairement aux approches précédentes (noeud maître et arbre couvrant), la disparition d'un noeud quelconque ou la rupture d'un lien n'entraîne jamais l'échec global de l'algorithme. Si un noeud quitte le réseau avant d'avoir diffusé sa valeur, celle-ci ne contribue à l'estimée

d'aucun autre noeud. Les nœuds restants estiment donc correctement la moyenne des nœuds restés actifs. Si un noeud quitte le réseau après avoir diffusé sa valeur, le résultat des nœuds restants n'est pas impacté.

Un tel protocole nécessite toutefois un réseau à connectivité complète, comme en témoigne la matrice $\mathbf{1}\mathbf{1}^\top$: toutes ses entrées étant non-nulles, chaque noeud doit envoyer son estimée à tous les autres. A moins de compter sur un protocole sous-jacent qui assure une communication un-vers-tous de voisin en voisin (*multi-hop overlay network*), on ne peut donc pas utiliser un protocole par diffusion sur un réseau quelconque.

Les protocoles par diffusion font souvent l'hypothèse que les canaux de communication fonctionnement eux-mêmes par diffusion (*broadcast channels*), c'est à dire que l'envoi d'un même message à de multiples voisins se fait en une seule opération de l'émetteur (*e.g.*, réseaux sans fils RF). Cette hypothèse n'est pas valide dans les réseaux points-à-points qui nécessitent $(N-1)$ envois sur $(N-1)$ liens distincts, d'où un coût de communication élevé en $\mathcal{O}(N^2)$.

Plus important, chaque noeud doit recevoir et calculer la somme de $(N-1)$ estimées, ce qui implique au mieux $\mathcal{O}(N)$ opérations locales en tout noeud, évidemment exécutées séquentiellement. Ainsi, malgré l'asynchronisme apparent de la diffusion, l'agrégation que chaque noeud doit réaliser a un coût calculatoire identique à un noeud maître. La duplication totale des émissions de messages et des calculs en chaque noeud fait ainsi de l'approche par diffusion une solution peu efficace.

1.4 Protocoles par consensus

Si la redondance maximale de l'approche par diffusion apparaît comme bénéfique à la décentralisation, celle-ci est obtenue au détriment de l'efficacité (chaque noeud envoie et reçoit un grand nombre de messages et effectue donc un grand nombre d'opérations locales). Les protocoles par consensus proposent un compromis entre décentralisation (qui requiert une plus grande redondance) et efficience (qui requiert une plus faible redondance), en s'appuyant sur un processus itératif.

A chaque étape t, chaque noeud i moyenne son estimée courante avec certains voisins déterminés :

$$\forall t, \forall i, \quad x_i(t+1) = \sum_{j=1}^{N} \kappa_{ji} x_j(t), \quad \text{avec} \quad \forall j, \sum_{i=1}^{N} \kappa_{ji} = 1 \quad \text{et} \quad \forall j, \kappa_{ij} \geq 0 \tag{1.16}$$

où $\kappa_{ij}(t) \neq 0$ si et seulement si le noeud i est connecté au noeud j. En notant $\mathbf{x}(t) = (x_1(t), \ldots, x_N(t))^\top$ et $\mathbf{K} = [\kappa_{ij}]_{i,j}^N$, ce processus prend une forme matricielle très simple :

$$\forall t, \quad \mathbf{x}(t+1)^\top = \mathbf{x}(t)^\top \mathbf{K} \quad \text{avec} \quad \mathbf{K} \geq \mathbf{0} \quad \text{et} \quad \mathbf{K}\mathbf{1} = \mathbf{1} \tag{1.17}$$

En développant la récursion, on peut également écrire

$$\forall t, \quad \mathbf{x}(t)^\top = \mathbf{x}(0)^\top \mathbf{K}^t \tag{1.18}$$

La matrice $\mathbf{K}$, de dimensions $N \times N$, est appelée "matrice de transition", "matrice de communication" ou "matrice de consensus". Sous les contraintes $\mathbf{K} \geq \mathbf{0}$ et $\mathbf{K}\mathbf{1} = \mathbf{1}$, elle est dite "stochastique en ligne" (*row-stochastic*) ou simplement "stochastique".

Les matrices non-négatives ($\mathbf{K} \geq \mathbf{0}$) reçoivent une attention particulière en algèbre linéaire de part les nombreuses propriétés, notamment spectrales, qui les caractérisent. La famille des matrices stochastiques ($\mathbf{K1} = \mathbf{1}$) en forme une sous-classe importante puisqu'elle correspond à l'ensemble des chaînes de Markov (homogènes) à N états. L'analyse du comportement du processus (1.16) relève ainsi de l'étude d'une chaîne de Markov homogène à temps et espace discrets. De nombreuses méthodologies d'analyse des chaînes de Markov ont été développées depuis le début du XX$^{\text{ième}}$ siècle [172, 218], notamment à travers les notions d'ergodicité et de distributions stationnaires.

La stochasticité de $\mathbf{K}$, qui implique que $\mathbf{1}$ est un vecteur propre droit de $\mathbf{K}$ associé à la valeur propre 1, est une condition nécessaire et suffisante à la conservation de la somme des estimées à travers le réseau (on dit que $\mathbf{K}$ conserve la masse) :

Propriété 1. *Conservation de masse.*

$$\mathbf{K1} = \mathbf{1} \quad \Leftrightarrow \quad \forall \mathbf{x}(0) \in \mathbb{R}^N, \sum_{i=1}^{N} x_i(t) = \sum_{i=1}^{N} x_i(0) \tag{1.19}$$

Démonstration. Par stochasticité de $\mathbf{K}$, on a

$$\forall t, \quad \sum_{i=1}^{N} x_i(t+1) = \mathbf{x}(t+1)^\top \mathbf{1} = \mathbf{x}(t)^\top \mathbf{K1} = \mathbf{x}(t)^\top \mathbf{1} = \sum_{i=1}^{N} x_i(t) \tag{1.20}$$

En développant la récurrence jusqu'à $t = 0$, on obtient trivialement (1.19). □

Cette propriété assure que si les nœuds convergent vers une même valeur c, celle-ci est égale à $\bar{x} \equiv \frac{1}{N}\sum_{i=1}^{N} x_i(0)$:

$$\mathbf{x}(t) = c\mathbf{1} \quad \Rightarrow \quad \mathbf{x}(t)^\top \mathbf{1} = c\mathbf{1}^\top \mathbf{1} \tag{1.21}$$

$$\mathbf{x}(0)^\top \mathbf{1} = cN \tag{1.22}$$

$$c = \bar{x} \tag{1.23}$$

1.4.1 Théorie de Perron-Frobenius et ergodicité

Pour garantir la convergence du processus (1.17), il nous faut fixer certaines contraintes sur $\mathbf{K}$, contraintes qui nous sont données par le théorème de Perron-Frobenius [196] :

Theorème 1. *Perron-Frobenius (pour les matrices non-négatives)*
Soit $\mathbf{A}$ une matrice $N \times N$ non-négative.
Si $\mathbf{A}$ est irréductible, $\mathbf{A}$ possède une valeur propre r réelle strictement positive telle que :

(a) r est associée à des vecteurs propres gauche et droit strictement positifs.

(b) Tout autre vecteur propre a au moins une composante négative.

(c) r est une racine simple de l'équation caractéristique de $\mathbf{A}$.

(d) Les espaces propres gauche et droit associés à r sont de dimension 1.

(e) Pour toute autre valeur propre λ, on a $|\lambda| \leq r$.
Si de plus $\mathbf{A}$ est primitive, $|\lambda| < r$.

(e) Pour toute matrice $\mathbf{B}$ telle que $\mathbf{0} \leq \mathbf{B} \leq \mathbf{A}$, toute valeur propre β de $\mathbf{B}$ satisfait $|\beta| \leq r$.
De plus, $|\beta| = r$ implique $\mathbf{B} = \mathbf{A}$.

Le théorème de Perron-Frobenius fait intervenir deux notions importantes : l'irréductibilité et la primitivité. Leurs définitions respectives sont données ci-dessous.

Définition 1. *Matrice irréductible.*
Une matrice non-négative $\mathbf{A}$ de dimensions $N \times N$ est dite "irréductible" si et seulement si

$$\forall (i,j) \in \{1,\dots,N\}^2, \quad \exists m \in \mathbb{N} : \quad \left(\mathbf{A}^m\right)_{ij} > 0 \tag{1.24}$$

Définition 2. *Matrice primitive.*
Une matrice non-négative $\mathbf{A}$ de dimensions $N \times N$ est dite "primitive" si et seulement si

$$\exists m \in \mathbb{N} : \quad \forall (i,j) \in \{1,\dots,N\}^2, \qquad \left(\mathbf{A}^m\right)_{ij} > 0 \tag{1.25}$$

On remarque qu'une matrice positive est trivialement primitive et qu'une matrice primitive est elle-même irréductible. Dans notre interprétation de $\mathbf{K}$ comme matrice de communication, dire que $\mathbf{K}$ est irréductible correspond à établir que tout noeud i communique une partie de son estimée à tout autre noeud j après m itérations du processus, où m *dépend* du lien (i,j). Dire que $\mathbf{K}$ est primitive fixe une contrainte plus forte : après m itérations, tous noeud i a communiqué une partie de son estimée à tout noeud j, où m *ne dépend pas* de (i,j).

Introduisons la matrice d'adjacence $\mathbf{A}$ du graphe $\mathcal{G}$ définie par $\mathbf{A}_{ij} = 1$ si il existe un lien du noeud i au noeud j et $\mathbf{A}_{ij} = 0$ sinon. On dit que $\mathbf{K}$ est "compatible" avec $\mathcal{G}$, si et seulement si $\mathbf{A}_{ij} = 0 \Rightarrow \mathbf{K}_{ij} = 0$, c'est à dire si seuls les coefficients κ_{ij} pour lesquels il existe un lien du noeud i au noeud j peuvent être non-nuls. Les coefficients non-nuls de $\mathbf{K}$ définissent quant à eux un sous-graphe $\mathcal{K}$ de $\mathcal{G}$, tel que l'arête (i,j) est dans $\mathcal{K}$ si et seulement si $\mathbf{K}_{ij} > 0$.

Sous cette définition, l'irréductibilité de $\mathbf{K}$ se traduit par l'existence dans $\mathcal{K}$ d'un certain chemin reliant tout noeud i à tout noeud j . On a alors la définition équivalente [218] :

Propriété 2. *Une matrice non-négative $\mathbf{A}$ est irréductible si et seulement si le graphe défini par $\mathbf{A}$ est fortement connexe.*

Plus contraignante, la primitivité de $\mathbf{K}$ correspond à la possibilité de relier tout noeud i à tout noeud j par des chemins de longueur identique. On peut montrer que $\mathbf{K}$ est primitive si et seulement si le graphe $\mathcal{K}$ est fortement connecté et admet au moins deux cycles de longueur premières entre elles [218] (on dit que $\mathbf{K}$ est apériodique). Étant donnée une matrice primitive $\mathbf{A}$, le plus petit m tel que $\mathbf{A}^m > \mathbf{0}$ est appelé "indice de primitivité" de $\mathbf{A}$. Notons que si $\mathbf{A} \in \mathbb{R}^{n \times n}$, son indice de primitivité est toujours inférieur à $n^2 - 2n - 2$, et que ce pire cas est atteint pour la matrice

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ & & & & \\ 0 & 0 & 0 & \ddots & 1 \\ 1/2 & 1/2 & 0 & \cdots & 0 \end{pmatrix} \tag{1.26}$$

et toutes ses permutations [59].

Dans notre cas, nous pouvons assurer que $\mathbf{K}$ est primitive en ajoutant la contrainte que les nœuds conservent toujours une portion non-nulle de leur estimée, c'est à dire $\forall i, \kappa_{ii} > 0$. En effet, lorsque $\mathbf{K}$ a une diagonale strictement positive, irréductibilité et primitivité sont équivalentes. Intuitivement, on peut en effet construire des cycles de longueurs arbitraires (notamment premières entre elles) pour démontrer que $\mathbf{K}$ est alors apériodique (voir *e.g.*, [178] pour une preuve complète) :

Propriété 3. *Si une matrice non-négative irréductible* $\mathbf{A}$ *a sa diagonale strictement positive, alors elle est primitive.*

Nous supposons donc par la suite que $\mathcal{K}$ est fortement connexe et que $\forall i, \kappa_{ii} > 0$. $\mathbf{K}$ est donc primitive et le théorème de Perron-Frobenius s'applique. Puisque $\mathbf{K}$ est stochastique, on a $r = 1$ et le vecteur propre droit associé est $\mathbf{1}$. Notons sa seconde valeur propre de plus grand module λ_2. L'analyse du comportement asymptotique de $\mathbf{K}^t$ est facilité par l'introduction de la forme normale de Jordan de $\mathbf{K}$:

$$\mathbf{K} = \mathbf{P}\mathbf{J}\mathbf{P}^{-1} \tag{1.27}$$

$$\mathbf{K}^t = \mathbf{P}\mathbf{J}^t\mathbf{P}^{-1} \tag{1.28}$$

D'après le théorème de Perron-Frobenius, 1 est une valeur propre simple et $|\lambda_2| < 1$. En posant

$$\mathbf{A} \stackrel{\text{def}}{=} \mathbf{K} - \mathbf{1}\mathbf{v}^\top \tag{1.29}$$

où $\mathbf{v}$ est le vecteur propre gauche dominant de $\mathbf{K}$, défini par $\mathbf{v}^\top\mathbf{K} = \mathbf{v}^\top$

et en introduisant le rayon spectral $\rho(\mathbf{A})$ défini comme la valeur propre dominante de $\mathbf{A}$, on a clairement $\rho(\mathbf{A}) = \lambda_2$. Or,

$$\rho(\mathbf{A}^t) \leq \rho(\mathbf{A})^t \tag{1.30}$$

Par conséquent $\lim_{t\to\infty} \mathbf{A}^t = \mathbf{0}$, d'où

$$\lim_{t\to\infty} \mathbf{K}^t = \mathbf{1}\mathbf{v}^\top \tag{1.31}$$

En rappelant (1.18), on obtient

$$\lim_{t\to\infty} \mathbf{x}(t)^\top = \lim_{t\to\infty} \mathbf{x}(0)^\top\mathbf{K}^t = \mathbf{x}(0)^\top\mathbf{1}\mathbf{v}^\top = \bar{x}N\mathbf{v}^\top \tag{1.32}$$

Notons que dans le contexte des chaînes de Markov, l'équation (1.31) signifie que $\mathbf{K}$ est *ergodique*, et $\mathbf{v}$ est appelé "distribution stationnaire" de la chaîne de Markov $\mathbf{K}$.

Considérons le cas où les colonnes de $\mathbf{K}$ somment à 1, qui assure que toute estimée $x_i(t+1)$ est toujours une moyennes pondérée des estimées des voisins $x_j(t)$. On a donc $\mathbf{1}^\top\mathbf{K} = \mathbf{1}^\top$ d'où $\mathbf{v} = \frac{1}{N}\mathbf{1}$. C'est notamment le cas lorsque l'on considère des échanges bidirectionnels, puisque $\mathbf{K}$ est alors symétrique. On obtient alors par (1.31) et (1.32) :

$$\lim_{t\to\infty} \mathbf{K}^t = \frac{1}{N}\mathbf{1}\mathbf{1}^\top \quad \text{et} \quad \lim_{t\to\infty} \mathbf{x}(t) = \bar{x}\mathbf{1} \tag{1.33}$$

Autrement dit, toutes les estimées x_i convergent vers leur moyenne sous réserve que les colonnes de $\mathbf{K}$ somment à 1. Rappelons qu'en tant que matrice stochastique, les lignes de $\mathbf{K}$ somment également à 1. On dit alors que $\mathbf{K}$ est *doublement stochastique*.

Pour toute norme matricielle $\|\cdot\|$, la formule de Gelfand [102] nous indique que

$$\lim_{t\to\infty} \|\mathbf{A}^t\| = \rho(\mathbf{A})^t = \lambda_2^t \tag{1.34}$$

Le taux de convergence asymptotique d'un protocole par consensus est donc exponentiel. Définissons le temps de convergence T, nécessaire pour obtenir une précision désirée $\varepsilon > 0$, également appelé "temps de ε-mélange" (ε-*mixing time*) de la chaîne de Markov $\mathbf{K}$, par

$$T = \min_i\{t : \forall t' \geq t, \Delta_i(t') \leq \varepsilon\} \quad \text{où} \quad \Delta_i(t') = \frac{1}{2}\sum_{j=1}^{N}\left|(\mathbf{K}^{t'})_{ij} - v_j\right| \tag{1.35}$$

Les bornes suivantes permettent alors de lier T à λ_2 [224] :

$$\frac{\lambda_2}{2(1-\lambda_2)}\ln\frac{1}{2\varepsilon} \leq T \leq \frac{\ln N + \ln(1/\varepsilon)}{1-\lambda_2} \tag{1.36}$$

La vitesse de convergence est ainsi directement liée à la différence $1-\lambda_2$ entre les deux valeurs propres dominantes de $\mathbf{K}$, appelée "gap spectral" (*spectral gap*).

1.4.2 Structure du réseau et vitesse de convergence

De nombreux travaux ont identifié des liens étroits entre les propriétés structurelle du graphe $\mathcal{K}$ et le spectre de $\mathbf{K}$, permettant ainsi de déterminer le taux de convergence du protocole de consensus associé [9, 248].

Par exemple, un gap spectral élevé correspond à une "bonne connectivité". Cette connectivité peut être exprimée par une mesure dite de "conductance", définie par

$$\Phi(\mathcal{G}) = \min_{S\subset V}\frac{|\partial(S)|}{d(S)d(V\setminus S)}, \quad \text{avec} \quad d(S) \stackrel{\text{def}}{=} \sum_{i\in S} d_i \tag{1.37}$$

où $V = \{1,\ldots,N\}$ est l'ensemble des nœuds de $\mathcal{G}$, d_i le degré du noeud i et $\partial(S)$ l'ensemble des arêtes (i,j) telles que $i \in S$ et $j \in V\setminus S$. La conductance mesure donc le nombre minimal d'arêtes qui relient toute partition de $\mathcal{G}$ en deux sous-graphes de taille significative. De manière équivalente, elle caractérise la probabilité qu'une marche aléatoire (*random walk*) "s'échappe" en une étape de tout sous-graphe de $\mathcal{G}$. Ainsi, une faible conductance implique une constriction de l'information dans des goulots d'étranglements empêchant la convergence vers une distribution stationnaire, tandis qu'une conductance élevée assure que l'information ne reste pas "bloquée" dans une zone du réseau relativement isolée. A partir de cette caractérisation topologique de conductance, l'inégalité de Jerrum-Sinclair offre une borne sur le gap spectral [142] :

$$\frac{\Phi(\mathcal{G})^2}{16} \leq 1-\lambda_2 \leq \Phi(\mathcal{G}) \tag{1.38}$$

Cette borne lie donc la "bonne connectivité" à la vitesse de convergence, dont la relation directe est démontrée dans [166] :

$$\forall i,j, \quad \left|\mathbf{K}_{ij}^t - v_j\right| \leq \sqrt{\frac{v_j}{v_i}}\left(1-\frac{\Phi^2}{8}\right)^t \tag{1.39}$$

La conductance, souvent peu commode (son calcul est NP-difficile dans la plupart des cas), peut elle-même être mise en relation avec la notion de flot maximal admissible [72, 224], en interprétant $\mathbf{K}$ comme la circulation d'un flot de $v_i v_j$ unités entre chaque couple de noeuds (i,j), avec $\mathbf{K}_{ji}$ représentant la capacité (quantité d'unités admissibles) de l'arête (i,j). Ce flot maximal est défini par

$$\rho_{\max} = \max_{(i,j)} \frac{v_i v_j}{\mathbf{K}_{ij}} \sum_{\gamma \in \Gamma(i,j)} |\gamma| \tag{1.40}$$

où $\Gamma(i,j)$ désigne l'ensemble des chemins de i vers j dans $\mathcal{G}$ et $|\gamma|$ est la longueur du chemin γ en nombre d'arêtes traversées. D'après [224], on a alors la borne suivante sur le gap spectral :

$$1 - \lambda_2 \geq \frac{1}{\rho_{\max}} \tag{1.41}$$

Une mesure très similaire à la définition (1.37) de la conductance est la constante de Cheeger $\Phi_C(\mathcal{G})$, analogue à celle utilisée en géométrie riemannienne pour identifier l'aire de l'hypersurface minimale qui sépare une variété riemannienne en deux morceaux disjoints (on voit ici clairement le lien géométrique avec la notion de goulot d'étranglement) :

$$\Phi_C(\mathcal{G}) = \min_{S \subset V} \frac{|\partial S|}{\min\{d(S), d(V \setminus S)\}} \tag{1.42}$$

L'inégalité de Cheeger [56] établit une relation entre cette constante et λ_2 (voir [58] pour une preuve récente sur les graphes orientés) :

$$\frac{\Phi_C(\mathcal{G})^2}{2} \leq 1 - \lambda_2 \leq 2\Phi_C(\mathcal{G}). \tag{1.43}$$

Le taux de convergence peut ainsi être caractérisé à partir de propriétés topologiques du réseau très diverses [8, 71, 166].

1.4.3 Exemples

1. **Diffusion** : A titre d'exemple, considérons tout d'abord $\mathbf{K} = \frac{1}{N}\mathbf{1}\mathbf{1}^\top$. En réexaminant (1.15), on retrouve clairement le cas du protocole par diffusion, qui converge en une seule étape. En effet, puisque $\frac{1}{N}\mathbf{1}\mathbf{1}^\top$ est de rang 1, on a $\lambda_2(\mathbf{K}) = 0$ et donc

$$\forall t \geq 1, \quad \mathbf{x}(t) = \bar{x}\mathbf{1} \tag{1.44}$$

   Cette stratégie est donc la plus rapide en terme de nombre d'itérations t, mais elle induit N calculs locaux en chaque noeud, d'où une complexité[1] en $\mathcal{O}(N^2)$.

2. **Cycle 1-régulier** : Considérons à présent

$$\mathbf{K}_{ij} = \begin{cases} 1/2 & \text{si } i \in \{j-1, j, j+1\} \quad \text{ou} \quad \{i,j\} = \{1,N\} \\ 0 & \text{sinon} \end{cases} \tag{1.45}$$

[1] Ici, nous qualifions de manière équivalente la complexité de communication et la complexité calculatoire, en

Le graphe $\mathcal{K}$ est un donc cycle 1-régulier, c'est à dire un réseau en anneau bidirectionnel ou chaque noeud communique avec deux voisins. Il a été démontré dans, *e.g.*, [37, 228] que

$$1 - \lambda_2(\mathbf{K}) = \Theta(1/N^2) \tag{1.46}$$

Il s'ensuit que le temps de convergence est en $\Omega(N^2)$, avec un coût par itération en $\mathcal{O}(N)$.

3. **Expandeur d-régulier** : On dit que $\mathcal{G}$ est un ε-expandeur si et seulement si pour tout sous-ensemble $S \in V$ contenant au plus la moitié des noeuds de $\mathcal{G}$, le nombre de nœuds qui ne sont pas dans S mais qui ont au moins un voisin dans S est supérieur à $\varepsilon|S|$. Un ε-expandeur d-régulier est un ε-expandeur dont tous les noeuds ont d voisins. Diverses méthodes permettent de construire un tel expandeur [36, 108]. D'après [166], tout ε-expandeur d-régulier satisfait

$$1 - \lambda_2 \geq \frac{\varepsilon^2}{5} \tag{1.47}$$

Ainsi, si on construit $\mathcal{G}$ de sorte à ce que $\varepsilon = \mathcal{O}(\log N)$, le temps de convergence devient $\mathcal{O}(\log^2 N)$ avec un coût par itération de dN où d ne dépend pas de N. Sur un expandeur de ce type, un protocole par consensus parvient ainsi à converger avec un coût en $\mathcal{O}(N \log^2 N)$, considérablement inférieur aux solutions par diffusion, tout en évitant les écueils d'une solution par arbre couvrant grâce à l'exploitation de la redondance des liens.

4. **Réseau "petit monde" (*small world*)** : Une classe importante de modèles de réseaux rassemblent ceux qui exhibent la propriété "petit monde" [249]. De manière informelle, un réseau petit monde est un réseau où tout couple de nœuds peut être relié par un chemin "très court" comparé au nombre de nœuds du réseau. Il existe plusieurs définitions formelles de ce phénomène, qui survient dans de nombreux réseaux réels (réseaux sociaux, réseaux de citations, réseaux de neurones biologiques, *etc*). Nous adoptons ici celle de [228] : Un graphe $\mathcal{G}$ a la propriété petit monde si et seulement si son diamètre $D(\mathcal{G})$, défini comme le maximum de la longueur du plus court chemin entre tout couple de nœuds, satisfait :

$$D(\mathcal{G}) = \mathcal{O}(\log N) \tag{1.48}$$

Un exemple de graphe ayant cette propriété est donné par le modèle de Newman [185], qui consiste à ajouter des arêtes aléatoires indépendantes et uniformes à un cycle d-régulier. Il peut être ainsi vu comme l'union d'un cycle d-régulier et d'un graphe Erdős-Rényi (ER, [82]), où l'arête (i, j) existe avec une probabilité donnée p indépendante de i et de j. Dans [228], il est prouvé que pour de tels réseaux le temps d'ε-mélange est

$$T = \mathcal{O}(\log^5 N) \tag{1.49}$$

Les réseaux petit monde partagent donc l'avantages de faible complexité des expandeurs tout en offrant des règles de constructions explicites. Il existe de nombreuses constructions

supposant que chaque message émis ou reçu implique au minimum une opération locale atomique de l'émetteur et

offrant la propriété petit monde. Par exemple le modèle de Barabasí-Albert (BA, [16]) intègre une règle d'attachement préférentiel par laquelle les nœuds se connectent avec une plus grande probabilité à des nœuds possédant déjà un grand nombre de voisins. Outre la propriété petit-monde, ces réseaux offrent des propriétés supplémentaires accélérant encore la convergence, parmi lesquelles l'invariance à échelle (*scale-free*) découlant de la distribution des degrés des nœuds en loi de puissance (le nombre de noeuds ayant k voisins est de l'ordre de $k^{-\gamma}$).

Notons enfin que les protocoles par consensus sont adaptés au calcul de moyenne pondérée, par le simple ajout de poids initiaux $\mathbf{w} = (w_1, \ldots, w_N)$. En effet, si l'on souhaite calculer $\bar{x} \overset{\text{def}}{=} \frac{\sum_{i=1}^{N} w_i x_i}{\sum_{i=1}^{N} w_i}$, il suffit de fixer $x_i(0) = w_i x_i$ et $w_i(0) = w_i$ et de faire évoluer $\mathbf{w}(t)$ selon le même processus que $\mathbf{x}(t)$:

$$\begin{cases} \mathbf{x}(t+1)^\top = \mathbf{x}(t)^\top \mathbf{K} \\ \mathbf{w}(t+1)^\top = \mathbf{w}(t)^\top \mathbf{K} \end{cases} \overset{(1.32)}{\Rightarrow} \begin{cases} \lim_{t\to\infty} \mathbf{x}(t) = \mathbf{x}(0)^\top \mathbf{1}\mathbf{v} \\ \lim_{t\to\infty} \mathbf{w}(t) = \mathbf{w}(0)^\top \mathbf{1}\mathbf{v} \end{cases} \tag{1.50}$$

$$\Rightarrow \forall i,\ \lim_{t\to\infty} \frac{x_i(t)}{w_i(t)} = \frac{\sum_{i=1}^{N} x_i(0)}{\sum_{i=1}^{N} w_i(0)} = \bar{x} \tag{1.51}$$

La moyenne $\bar{x}$ peut donc être estimée en tout noeud par le quotient $x_i(t)/w_i(t)$. Remarquons que dès lors, $\mathbf{K}$ n'a plus besoin d'être stochastique en colonne ($\mathbf{v} \neq \frac{1}{N}\mathbf{1}$), mais simplement stochastique, car $\mathbf{v}$ se trouve annulé dans le quotient (effet de normalisation des échanges).

1.5 Protocoles Gossip

Grâce à la prise en compte de la connectivité intrinsèque du réseau, les protocoles par consensus permettent d'exploiter les avantages de la décentralisation tout en maîtrisant la redondance des calculs locaux pour assurer une faible complexité en terme de calculs locaux et de nombre de messages échangés.

Cependant, une fois la matrice de communication $\mathbf{K}$ fixée, le protocole utilise les mêmes liens tout au long du processus, entraînant un certain nombre de limitations :

- Les liens (i, j) pour lesquels $\mathbf{K}_{ij} = 0$ sont totalement ignorés, alors qu'ils pourraient contribuer à la redondance donc à la décentralisation des calculs. A l'inverse, les integrer tous conduit à une sous-efficience considérable (*cf.* diffusion).

- Un lien rompu peut entraîner l'invalidation des hypothèses de stochasticité. Si un noeud i peut corriger la stochasticité en ligne de son côté en modifiant ses coefficients (chacun sur sa propre ligne i) sans coordination avec les autres nœuds, il n'en va pas de même de la stochasticité en colonne, dont la correction requiert une coordination. Notons que cette correction n'est pas nécessaire lorsque l'on calcule une moyenne pondérée par (1.51).

- Si un nouveau noeud rejoint le réseau en cours de processus (on parle de réseau dynamique), la matrice $\mathbf{K}$ doit être globalement adaptée pour l'intégrer aux échanges, opération qui ne peut se faire sans coordination globale.

du récepteur.

Les protocoles dits "Gossip", à l'origine conçus pour la dissémination et la réplication d'information dans les bases de données distribuées, ont été étendus avec succès aux problèmes de calculs d'agrégats dans le début des années 2000. Comme leur nom l'indique, les protocoles Gossip s'inspirent des mécanismes de propagation de rumeurs au sein d'une population d'individus hétérogène où chacun souhaite diffuser une information spécifique à ses voisins, mécanismes qui induisent des phénomènes émergents de consensus et d'attracteurs stables [202]. Dans leur fonction d'origine, à savoir la réplication de données, les protocoles Gossip s'inspirent des propriétés de propagation rapide des épidémies, en modélisant la diffusion de l'information comme un processus infectieux (*epidemics spreading*) qui se transmet des individus infectés aux individus sains avec une certaine distribution de probabilité [68]. Les protocoles Gossip s'appuient ainsi sur la propagation de données entre nœuds du réseau non plus de manière déterministe, mais aléatoire (la littérature anglophone explicite souvent cet aspect en les nommant *Randomized Gossip Algorithms* [39, 144, 219]). Ce processus de diffusion respecte généralement la propriété de Markov, puisque le taux d'infection à un instant donné dépend uniquement des situations immédiatement précédentes. Sa modélisation s'exprime donc sous la forme d'une chaîne de Markov inhomogène. Les modèles Gossip sont ainsi formellement semblables aux modèles de croissance de population (*population growth models*, [244]), aux modèles canoniques en épidémiologie (*e.g.*, Kermack-McKendrick [42, 147]), ou ceux utilisés en théorie de la percolation [113] ou en analyse de l'auto-organisation des systèmes multi-agents (*e.g.*, modèle de Vicsek [246]) ou des essaims d'insectes [46].

Dans le contexte de l'estimation d'agrégats statistiques les premières extensions de l'approche Gossip se sont intéressées au calcul d'extremums, de médianes et de moyennes [145]. Dans sa forme la plus générale, un algorithme d'estimation Gossip est donné par la spécification d'un opérateur d'agrégation $\oplus$ associatif et commutatif, et d'un processus de communication entre nœuds, itératif et aléatoire. L'estimée de chaque noeud i évolue ainsi selon la règle suivante :

$$x_i(t+1) = \bigoplus_{j=1}^{N} \mathbf{K}_{ji}(t) x_j(t), \tag{1.52}$$

où $\mathbf{K}_{ji}(t) = 1$ si j envoie un message à i à l'instant t et $\mathbf{K}_{ji}(t) = 0$ sinon.

A titre d'illustration, si $\oplus := \max$, il est clair que

$$\forall i,\ \lim_{t\to\infty} x_i(t) = \max_{1\leq j\leq N} x_j(0) \quad \text{ssi} \quad \exists T : \forall t \geq T, \prod_{t'=1}^{t} \mathbf{K}(t') > \mathbf{0} \tag{1.53}$$

Tous les nœuds du réseau convergent donc vers leur maximum après T itérations, la condition de convergence de (1.53) pouvant être vue comme une extension de la primitivité aux produits de matrices non-négatives inhomogènes [218] (lorsque $\mathbf{K}(t) \equiv \mathbf{K}$, on retrouve $\mathbf{K}^t > \mathbf{0}$). Cette propriété signifie qu'à partir d'un certain instant T, l'information portée par tout noeud i aura rejoint tout noeud j en transitant par au plus T voisins. Ainsi, on dit que l'ensemble $\mathcal{E} = \{\mathbf{K}(t)\}_t$ est *primitif* (ou *ergodique* [116]) si l'on peut construire un produit $\mathbf{K}_1 \dots \mathbf{K}_k > \mathbf{0}$ composé seulement d'éléments de $\mathcal{E}$ [59]. Par extension, on dit qu'une matrice aléatoire $\mathbf{K}$ à support dans $\mathcal{E}$ est elle-même *primitive* si $\mathcal{E}$ est primitif.

Par analogie avec un processus épidémique [84] défini par la contamination d'un individu sain i par un individu porteur j si et seulement si $\mathbf{K}_{ji}(t) > 0$, la primitivité de $\mathbf{K}(t)$ signifie que

l'infection se diffuse à l'ensemble de la population en au plus T étapes, quels que soient le ou les individus initialement affectés.

Considérons à présent $\oplus = +$. En fixant la contrainte que toutes les matrices $\mathbf{K}(t)$ sont stochastiques, on retrouve un processus similaire aux protocoles par consensus :

$$x_i(t+1) = \sum_{j=1}^{N} \mathbf{K}_{ji}(t+1)x_j(t), \quad \text{c'est à dire} \quad \mathbf{x}(t+1)^\top = \mathbf{x}(t)^\top \mathbf{K}(t+1) \tag{1.54}$$

Néanmoins, on constate que la chaîne de Markov correspondante n'est plus homogène dans le temps. Rappelons que dans le cas des chaînes de Markov homogènes (puissances d'une matrice $\mathbf{K}$ constante) l'ergodicité peut être facilement montrée par le théorème de Perron-Frobenius via le gap spectral de $\mathbf{K}$. Dans le cas des produits de matrices inhomogènes, puisque généralement $\lambda_2(\mathbf{AB}) \neq \lambda_2(\mathbf{A})\lambda_2(\mathbf{B})$, on ne peut plus utiliser directement ce théorème pour déterminer si il y a convergence vers un consensus.

1.5.1 Ergodicité forte, ergodicité faible

A la différence des chaînes de Markov homogènes pour lesquelles il n'existe qu'une forme d'ergodicité, dans le cas non-homogène, il existe une distinction importante entre *ergodicité forte* et *ergodicité faible* [218] :

Définition 3. *Une suite de matrices stochastiques $\{\mathbf{K}(t)\}_{t>0}$ est dite "fortement ergodique" si et seulement si*

$$\lim_{t\to\infty} \prod_{\tau=1}^{t} \mathbf{K}(\tau) = \mathbf{1}\mathbf{v}^\top, \quad \textit{où} \quad \mathbf{v}^\top \mathbf{1} = \mathbf{1} \tag{1.55}$$

L'ergodicité forte signifie donc que le produit des matrices $\mathbf{K}(t)$ tend vers une matrice constante dont les lignes sont identiques.

Définition 4. *Une suite de matrices stochastiques $\{\mathbf{K}(t)\}_{t>0}$ est dite "faiblement ergodique" si et seulement si*

$$\lim_{t\to\infty} \mu_S(\prod_{t'=1}^{t} \mathbf{K}(t)) = 0 \qquad \textit{où} \qquad \forall \mathbf{A},\ \mu_S(\mathbf{A}) \equiv \frac{1}{2} \max_{i,j} \sum_{k=1}^{N} |\mathbf{A}_{ik} - \mathbf{A}_{jk}| \tag{1.56}$$

$\mu_S(\mathbf{A})$ est appelé coefficient d'ergodicité de $\mathbf{A}$. Introduit par Dobrushin [74], ce coefficient a les propriété suivantes :

- *Borné* : Si $\mathbf{A}$ est stochastique, $0 \leq \mu_S(\mathbf{A}) \leq 1$.
- *Sous-multiplicatif* : $\mu_S(\mathbf{AB}) \leq \mu_S(\mathbf{A})\mu_S(\mathbf{B})$.
- *Propre* : $\mu_S(\mathbf{A}) = 0$ ssi $\text{rang}(\mathbf{A}) = 1$
- *Borne les valeurs propres sous-dominantes* : $\lambda_2(\mathbf{A}) \leq \mu_S(\mathbf{A})$.

Le coefficient d'ergodicité de Dobrushin indique ainsi la tendance d'une matrice à avoir des

lignes identiques. L'ergodicité faible signifie donc que le produit tend à avoir des lignes identiques, mais contrairement au cas fortement ergodique, ces lignes peuvent varier dans le temps. Les définitions équivalentes suivantes font apparaître clairement cette distinction :

Définition 5. $\{\mathbf{K}(t)\}_{t>0}$ *est fortement ergodique si et seulement si*

$$\exists \mathbf{v} \in \mathbb{R}^N : \lim_{t\to\infty} \left(\prod_{\tau=1}^{t} \mathbf{K}(\tau) - \mathbf{1}\mathbf{v}^\top \right) = 0 \tag{1.57}$$

Définition 6. $\{\mathbf{K}(t)\}_{t>0}$ *est faiblement ergodique si et seulement si*

$$\exists \{\mathbf{v}(t)\}_{t>0} : \lim_{t\to\infty} \left(\prod_{\tau=1}^{t} \mathbf{K}(\tau) - \mathbf{1}\mathbf{v}(t)^\top \right) = 0 \tag{1.58}$$

Toute suite fortement ergodique est aussi faiblement ergodique. Dans les deux cas le rang de la matrice produit tend vers 1, mais dans le cas fortement ergodique le vecteur propre dominant $\mathbf{v}$ est constant (comme dans le cas où $\mathbf{K}(t) \equiv \mathbf{K}$ est constante) tandis que dans le cas faiblement ergodique $\mathbf{v}$ dépend du temps.

Ces définitions sont formulées aussi bien pour les produits déterministes, où $\{\mathbf{K}(t)\}_{t>0}$ est une suite de matrices de communication fixée a priori, que pour les produits aléatoires, où cette séquence est générée par un processus stochastique. Les protocoles Gossip construisent la suite $\{\mathbf{K}(t)\}_{t>0}$ en tirant aléatoirement chaque $\mathbf{K}(t)$ de manière indépendante et identiquement distribuée parmi un ensemble fini de réalisations possibles.

1.5.2 Protocoles push-pull (doublement-stochastiques)

Il est clair que si toutes les matrices $\mathbf{K}(t)$ ont le même vecteur propre gauche dominant $\mathbf{v}$, *i.e*, $\forall t, \mathbf{v}^\top = \mathbf{v}^\top \mathbf{K}(t)$, ergodicité faible et ergodicité forte sont alors équivalentes. Pour cette raison, les protocoles dits "push-pull" adoptent des matrices $\mathbf{K}(t)$ doublement stochastiques, c'est à dire, $\mathbf{v} = \frac{1}{N}\mathbf{1}$.

Sous ces hypothèses, on peut alors montrer la convergence des estimées $\mathbf{x}(t) \equiv [x_1(t), \ldots, x_N(t)]^\top$ vers leur moyenne $\bar{x} = \frac{1}{N}\mathbf{x}^\top\mathbf{1}$ et déterminer la vitesse de cette convergence via le théorème suivant :

Theorème 2. *Soit un protocole Gossip push-pull définit par* $\mathbf{x}(t+1)^\top = \mathbf{x}(t)^\top \mathbf{K}(t+1)$ *où les* $\mathbf{K}(t)$ *sont iid et telles que*

$$\mathbf{1}^\top \mathbf{K}(t) = \mathbf{1}^\top, \qquad \mathbf{K}(t)\mathbf{1} = \mathbf{1} \qquad \textit{et} \qquad \mathbb{E}\mathbf{K} \textit{ est primitive} \tag{1.59}$$

L'erreur relative par rapport au consensus est alors bornée comme suit :

$$\forall 0 < \delta < 1,\ \forall \varepsilon > 0, \quad \mathbb{P}\left[\frac{\|\mathbf{x}(t) - \bar{x}\mathbf{1}\|_2^2}{\|\mathbf{x}(0)\|_2^2} \geq \varepsilon \right] \leq \delta \tag{1.60}$$

$$\textit{si} \quad t \geq T \equiv \frac{\ln\delta + \ln\varepsilon}{2\ln\lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top])} \tag{1.61}$$

La preuve de ce théorème est donnée en annexe A. Notons que la condition sur $\mathbb{E}[\mathbf{K}\mathbf{K}^\top]$ est

seulement suffisante. Dans [39, 130], les auteurs formulent une condition nécessaire impliquant $\mathbb{E}[\mathbf{K}\otimes\mathbf{K}]$ où $\otimes$ désigne le produit de Kronecker. Dans cette thèse, nous nous contentons toutefois du critère sur $\mathbb{E}[\mathbf{K}\mathbf{K}^\top]$ qui apparaît suffisamment précis pour démontrer la convergence des méthodes présentées dans les chapitres suivants. Le lecteur intéressé par une borne plus étroite pourra sans difficulté appliquer les résultats [39] à ces méthodes pour obtenir des critères plus favorables.

Observons que les contraintes de validité du théorème 2 sont bien plus générale que pour les protocoles par consensus : on peut choisir n'importe quelles matrices de communication doublement stochastiques tant que leur distribution est telle que $\mathbb{E}\mathbf{K}$ est primitive. Il n'est pas nécessaire que les $\mathbf{K}(t)$ soient elles-même doublement stochastiques. Il suffit seulement que l'ensemble des liens (i,j) sélectionnés avec une probabilité non-nulle forme un graphe fortement connexe, et que $\mathrm{diag}(\mathbf{K}(t)) > 0$. Lorsque tous les liens du réseau sont sélectionnés avec une probabilité non-nulle et que chaque noeud conserve une portion non-nulle de son estimée, $\mathbb{E}\mathbf{K}$ est ainsi logiquement primitive. Cela implique un résultat corollaire important :

Corollaire 1. *Tout protocole Gossip push-pull iid converge exponentiellement vers la moyenne dès lors que chaque lien du réseau est sélectionné avec une probabilité non-nulle.*

1.5.3 Exemple : Protocole Newscast

Newscast [140] est un protocole Gossip push-pull qui n'utilise qu'un seul lien bidirectionnel à chaque itération. Pour calculer une moyenne, chaque étape consiste ainsi à sélectionner uniformément une arête (i,j) du graphe non-orienté $\mathcal{G}$ et de calculer

$$x_i(t+1) = x_j(t+1) = \frac{x_i(t) + x_j(t)}{2} \tag{1.62}$$

Le fonctionnement pratique de Newscast est décentralisé : chaque noeud exécute une procédure identique rythmée par sa propre horloge interne. Il ne pose donc pas explicitement d'hypothèse de synchronisation globale. En un noeud quelconque i, chaque itération locale consiste à contacter un voisin aléatoire j et lui envoyer x_i. Le noeud j renvoie sa propre estimée x_j en réponse et calcule $x_j = (x_i + x_j)/2$. A réception de cette réponse, i calcule à son tour $x_i = (x_i + x_j)/2$. Cette procédure est synthétisée par l'algorithme 1 dans le cas général où les estimées $x_1, \ldots, x_N$ appartiennent à un espace vectoriel quelconque $\mathcal{V}$:

Algorithme 1 Newscast Averaging [141] *(en chaque noeud i)*

Entrées : $x_i \in \mathcal{V}$
Boucle
  $j \leftarrow$ noeud voisin aléatoire
  $x_i \leftarrow x_j \leftarrow \frac{1}{2}(x_i + x_j)$
Fin Boucle

Un avantage immédiat de Newscast est qu'à chaque étape, exactement deux messages sont échangés sur le réseau. On a ainsi une équivalence entre le nombre total d'itérations t (échelle de temps) et le nombre de messages échanges (coût de communication). On peut donc directement prédire les coûts de communication en appliquant le théorème 2.

Considérons un réseau totalement connecté $\mathcal{G} = (V, A)$ sans boucle, où tout couple de nœuds (i, j) peut communiquer de manière bidirectionnelle[2]. En supposant une sélection iid uniforme des liens, Newscast correspond à fixer :

$$\mathbf{K}(t) = \mathbf{I} - \frac{1}{2}(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\top, \quad \text{où} \quad (i, j) \sim \mathcal{U}_A \tag{1.63}$$

Le théorème suivant (prouvé en annexe A) établit alors le taux de convergence de Newscast sur un réseau complètement connecté :

Theorème 3. *Pour un réseau totalement connecté de N noeuds, Newscast atteint une erreur inférieure à ε avec une probabilité au moins δ en moins de M messages, où*

$$M \leq \frac{N}{2}(\ln 1/\delta + \ln 1/\varepsilon) \tag{1.64}$$

Autrement dit, il suffit que chaque noeud ait envoyé $(\ln 1/\delta + \ln 1/\varepsilon)/2$ messages. Par exemple, si chaque noeud envoie 10 messages, on obtient une précision de $\varepsilon = 10^{-6}$ avec une probabilité de 99% (ou une précision de 10^{-7} avec une probabilité de 90%), ceci *indépendamment de* N. Chaque message supplémentaire par noeud divise par $e \approx 2.5$ le seuil de précision ou d'incertitude.

Cette borne exhibe une propriété clé de Newscast : le nombre de messages par noeud nécessaires pour converger est invariant à la taille du réseau. En terme de passage à l'échelle et de déploiement pratique, ceci constitue un avantage important, puisqu'une étude empirique d'un petit système suffit à prédire celui d'un système analogue de très grande taille (dont l'étude est généralement difficile à mettre en œuvre en pratique).

Cet exemple illustre l'apport majeur des protocoles Gossip aléatoires vis-à-vis des protocoles de consensus déterministes. En effet, en présence d'un réseau totalement connecté, un protocole par consensus converge en une étape ($\lambda_2(\mathbf{K}) = 0$), mais cette étape requiert $N - 1$ messages par noeud. Au contraire, un algorithme Gossip tel que Newscast ne nécessite qu'un nombre d'étapes linéaire en N, où chaque étape n'implique que 2 échanges de messages, d'où un coût de communications largement inférieur en $\mathcal{O}(1)$ par noeud. Sur un graphe complet, les protocoles Gossip surpassent donc les protocoles déterministes d'un ordre de grandeur.

Lorsque que le graphe du réseau est un expandeur, le coût de communication des protocoles par consensus et des protocoles Gossip est équivalent, de l'ordre de $\mathcal{O}(\log N)$ messages par noeud. Similairement, pour un réseau en anneau (cycle), $\lambda_2 = \Theta(1 - \frac{1}{N^2})$, ce qui implique un coût par noeud en $\mathcal{O}(N)$. Sur un tel graphe, les protocoles Gossip ne sont alors pas plus efficaces qu'une approche par diffusion.

Les protocoles Gossip sont donc particulièrement adaptés aux réseaux complets ou de types petit monde, pour lesquels une grande précision peut être obtenue moyennant un nombre raisonnable de messages par noeud même lorsque le réseau comporte plusieurs milliards de nœuds.

Finalement, ces protocoles respectent notre contrainte de décentralisation. Notamment, on peut ajouter de nouveaux nœuds et/ou liens au réseau en cours d'exécution sans avoir à modifier le fonctionnement des nœuds existants.

[2]Notons que cette connectivité complète peut provenir soit directement de l'architecture matérielle du réseau elle-même, soit par un mécanisme de routage point-à-point sur un graphe fortement connecté (*e.g.*, Internet, *P2P overlay networks*, etc).

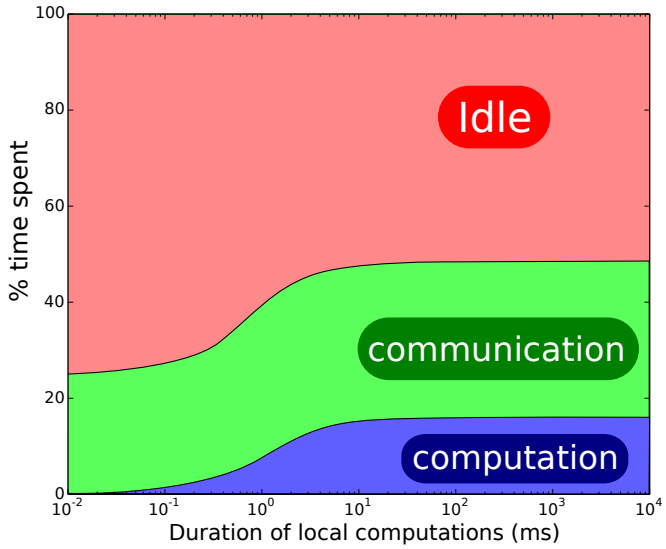

FIGURE 1.1 – Toute synchronisation locale des communications (ici entre paires de nœuds) engendre une synchronisation globale implicite et massive.

1.5.4 Problèmes de synchronisation implicite

Les protocoles Gossip décentralisés tels que Newscast n'impliquent aucune synchronisation globale du réseau, mais seulement une synchronisation locale, entre paires de nœuds. A première vue, cela semble offrir un grand avantage. Toutefois, comme nous l'avons suggéré en introduction de ce manuscrit, une synchronisation locale explicite induit systématiquement une synchronisation globale implicite.

Pour s'en convaincre, rappelons que les protocoles Gossip décrits jusqu'à présent nécessitent des matrices de communication $\mathbf{K}(t)$ doublement stochastiques pour garantir leur convergence : toutes les lignes et les colonnes des $\mathbf{K}(t)$ doivent sommer à 1. Supposons qu'à l'instant t, un noeud i envoie un message à un voisin j. On a donc $\mathbf{K}_{ij}(t) > 0$. Pour respecter la double stochasticité, il faut alors que $\mathbf{K}_{ii}(t) < 1$ et $\mathbf{K}_{jj}(t) < 1$. Il existe donc un noeud k tel que $\mathbf{K}_{jk}(t) > 0$, c'est à dire que j doit envoyer un message à k. Mais alors $\mathbf{K}_{kk}(t) < 1$ et donc $\exists l : \mathbf{K}_{kl}(t) > 0$ et ainsi de suite. En poursuivant ce raisonnement, on construit une chaîne de dépendance qui contient obligatoirement un cycle (ne serait-ce que par épuisement des N nœuds distincts).

Formellement, tout graphe de communication $\mathcal{K}(t)$ dont la matrice d'adjacence $\mathbf{K}(t)$ est doublement stochastique admet forcément un cycle, c'est à dire que $\mathbf{K}(t)$ ne peut être mise sous forme triangulaire par simple renumérotation des nœuds.

Tout protocole doublement stochastique nécessite donc qu'à chaque étape au moins un noeud soit à la fois émetteur et récepteur. C'est pourquoi on parle de protocole "push-pull", car une fois son estimée envoyée à l'étape t, un noeud doit attendre de recevoir celle d'un de ses voisins avant de procéder à l'étape $t + 1$.

Cette attente est obligatoire pour garantir la validité du consensus. En effet, supposons qu'un certain noeud i n'attende pas la réception de l'estimée $x_j(t)$ avant d'envoyer sa propre estimée à l'instant $t + 1$. Puisqu'en général $x_i(t) \neq x_i(t + 1) = x_i(t) + x_j(t)$, envoyer $x_i(t)$ à la place

de $x_i(t+1)$ résulte en une modification du consensus $\bar{x}$ qui ne correspond plus à la moyenne recherchée. La synchronisation locale est donc une caractéristique indissociable des protocoles doublement stochastiques. Dans sa forme la plus restreinte, cette synchronisation n'implique qu'une attente mutuelle entre un couple (i, j) à chaque instant t (*e.g.*, Newscast).

Même lorsque cette attente mutuelle est réduite à une paire de nœuds à chaque instant, les conséquences de cette synchronisation locale sur le comportement global du réseau sont désastreuses.

En observant l'algorithme 1, remarquons en effet qu'à tout instant, chaque noeud est dans un des trois états 'émission', 'réception' ou 'attente' :

- En un noeud i, l'état d'émission correspond à la multiplication par $\mathbf{K}_{ij}(t)$ de son estimée locale, la construction (empaquetage) du message et son envoi à j sur le réseau. Supposons que cette phase dure un temps réel constant $\nu_E > 0$.
- L'état de réception correspond à la réception d'un message entrant, le dépaquetage de l'estimée qu'il contient et son addition à l'estimée locale. Supposons que cette phase dure un temps réel $\nu_R > 0$.
- La phase d'attente correspond quant à elle au temps d'inactivité durant lequel le noeud i attend la réception d'un message de la part d'un voisin j auquel il a précédemment adressé un message. Cette réception ne peut survenir qu'un fois que j a traité tous ses messages entrants et effectué son étape d'émission. Ce temps d'attente ν_I est donc toujours supérieur à $Q\nu_R+\nu_E+2\nu_T$ où Q est le nombre de messages en attente de traitement en j au moment où i rentre en état d'attente et $\nu_T > 0$ le temps de transit d'un message sur le lien (i, j).

Au regard de la définition de ces trois états, le temps d'attente croît linéairement avec le nombre de messages en circulation à travers le réseau. Il en résulte que le temps d'attente tend à surpasser largement les temps d'émission et de réception en tout noeud. Une approche formelle de cette question recourt à la théorie des files d'attente, mais nous nous contentons ici d'une caractérisation empirique, suffisante à notre argument.

Simulons le comportement de Newscast sur un tel système, déterminé par les paramètres $(\nu_E, \nu_R, \nu_T, \nu_I)$, et mesurons le temps moyen passé par les nœuds dans les trois états 'émission','réception' et 'attente'. Pour toutes valeurs non nulles des ν_*, la figure 1.1 montre que les nœuds passent la majeure partie de leur temps à attendre leurs voisins. On constate donc l'émergence d'une synchronisation implicite au niveau global, malgré la restriction de la synchronisation locale à une simple paire de nœuds à chaque étape. Cette synchronisation globale induit ainsi un temps d'attente majoritaire qui nuit grandement aux performances du système.

Pour éviter cette synchronisation globale implicite et supprimer les temps d'attente, il faut se passer de toute synchronisation locale dans le fonctionnement du protocole, c'est à dire retirer la dépendance cyclique des échanges impliquée par la double stochasticité.

1.6 Protocoles Gossip asynchrones

Pour garantir un fonctionnement asynchrone, une alternative est de relâcher totalement une des contraintes de stochasticité. Si l'on relâche par exemple l'hypothèse de stochasticité en colonnes (on a donc plus $\mathbf{1}^\top\mathbf{K}(t) = \mathbf{1}^\top$), on peut alors sans difficulté mettre les matrices de communication $\mathbf{K}(t)$ sous forme triangulaire supérieure par renumérotation des nœuds. Un tel protocole est

alors qualifié de "*push only*" car les nœuds émetteurs ne sont jamais simultanément récepteurs. Formellement, le graphe de communication $\mathcal{K}(t)$ est toujours acyclique (*i.e.*, c'est un arbre).

Dans le cas des protocoles *push-pull*, nous avons vu que l'hypothèse de double-stochasticité des $\mathbf{K}(t)$, qui implique que $\mathbf{1}^\top$ est leur vecteur propre gauche dominant, est nécessaire pour garantir l'ergodicité forte de la chaîne de Markov inhomogène correspondante. Dans le cas des protocoles *push only*, l'ergodicité forte est perdue. D'une part, le vecteur propre gauche dominant de $\prod_{\tau=1}^{t} \mathbf{K}(t)$ n'est plus égal à $\mathbf{1}^\top$, mais plus important, il est susceptible d'évoluer librement dans le temps. Dans le cas général, le produit des $\mathbf{K}(t)$ ne converge alors jamais vers une matrice constante. La chaîne de Markov n'a donc pas de distribution stationnaire. Il nous faut donc integrer des mécanismes supplémentaires pour assurer la convergence d'un protocole *push-only* vers la moyenne recherchée.

1.6.1 Stochasticité seulement en espérance

Une première possibilité est de considérer des matrices de communication dont seule l'espérance est doublement stochastique (chaque $\mathbf{K}(t)$ n'est pas forcément stochastique, mais $\mathbb{E}\mathbf{K}$ est doublement stochastique). Dans ce cas, il est clair que $\mathbf{x}(t)$ ne converge pas vers une valeur stable. Néanmoins, si $\mathbb{E}\mathbf{K}$ est primitive, puisqu'elle est doublement stochastique, le théorème de Perron-Frobenius nous assure que

$$\lim_{t\to\infty} \mathbb{E}[\mathbf{K}^t] = \frac{\mathbf{1}\mathbf{1}^\top}{N} \tag{1.65}$$

L'astuce est alors de considérer en chaque noeud i la moyenne temporelle $y_i(t) = \frac{1}{t}\sum_{\tau=1}^{t} x_i(\tau)$. Contrairement aux estimées instantanées $x_i(t)$, ces moyennes temporelles $y_i(t)$, localement calculable en chaque noeud de manière incrémentale, convergent vers le consensus, comme le prouve le théorème suivant :

Theorème 4. *Soit $(\mathbf{K}(t))_{t\geq 1}$ une suite de matrices aléatoires iid telles que*

$$\forall t,\ \mathbf{K}(t) \textit{ non-négative}, \quad \mathbb{E}\mathbf{K} \textit{ primitive} \quad \textit{et} \quad \mathbb{E}\mathbf{K} \textit{ doublement stochastique} \tag{1.66}$$

Pour chaque noeud i, définissons les estimées locales suivantes

$$x_i(t+1) = \sum_{j=1}^{N} \mathbf{K}_{ji}(t) x_j(t) \quad \textit{et} \quad y_i(t+1) = \frac{t y_i(t) + x_i(t+1)}{t+1} \tag{1.67}$$

Alors les moyennes temporelles $y_i(t)$ convergent quasi-sûrement vers la moyenne $\bar{x} \equiv \frac{1}{N}\sum_{i=1}^{N} x_i(0)$:

$$\forall i,\ \forall \varepsilon > 0, \quad \lim_{t\to\infty} \mathbb{P}\Big[|y_i(t) - \bar{x}| < \varepsilon\Big] = 1 \tag{1.68}$$

Ce théorème est prouvé en annexe A. Cette solution permet ainsi de relâcher la contrainte de double-stochasticité des $\mathbf{K}(t)$, qui peuvent être des matrices non-négatives quelconques tant que $\mathbb{E}\mathbf{K}$ est doublement stochastique.

Par exemple, considérons le protocole suivant : à chaque instant t, un noeud aléatoire s envoie une portion de son estimée $x_s(t)$ à un voisin aléatoire r et conserve la portion restante. r

ajoute alors l'estimée reçue à sa propre estimée locale $x_r(t)$. Ce protocole a pour forme matricielle

$$\mathbf{x}(t+1)^\top = \mathbf{x}(t)^\top \mathbf{K}(t) \quad \text{avec} \quad \mathbf{K} = \mathbf{I} - \alpha_{sr}\mathbf{e}_s(\mathbf{e}_s - \mathbf{e}_r)^\top, \tag{1.69}$$

où $(s, r) \sim \mathcal{U}_A$ et α_{ij} des coefficients déterministes spécifiant la portion d'estimée envoyée par le noeud i au noeud j lorsque le lien (i, j) est sélectionné. La sélection des liens est aléatoire, mais les coefficients qui leur sont appliqués sont fixés. $\mathbb{E}\mathbf{K}\mathbf{1} = \mathbf{1}$ est trivialement vérifiée. Néanmoins, $\mathbf{1}^\top\mathbb{E}\mathbf{K} = \mathbf{1}^\top$ ne l'est pas nécessairement.

Protocoles symétriques en espérance

Pour respecter la double stochasticité de $\mathbb{E}\mathbf{K}$, une possibilité est de faire en sorte qu'elle soit symétrique. Cela implique que

$$\forall i, j, \quad \frac{\alpha_{ji}}{\alpha_{ij}} = \frac{\mathbb{P}[s = i, r = j]}{\mathbb{P}[s = j, r = i]}, \quad 0 \leq \alpha_{ij} \leq 1 \tag{1.70}$$

Autrement dit, le rapport des coefficients associés à un lien dans ses deux sens doit compenser le déséquilibre des probabilités de sélectionner une direction par rapport à l'autre. Il s'ensuit que les liens unidirectionnels ne sont jamais empruntés. En supposant que les émetteurs fonctionnent à la même fréquence et sélectionnent leurs récepteurs uniformément, on a

$$\mathbb{P}[s = i, r = j] = \mathbb{P}[r = j | s = i]\mathbb{P}[s = i] = \frac{1}{d(i)N}, \tag{1.71}$$

où $d(i)$ est le degré sortant du noeud i (*i.e.*, le nombre de voisins vers lesquels il peut émettre). La combinaison de (1.70) et (1.71) nous donne

$$\forall i, j, \quad \frac{\alpha_{ji}}{\alpha_{ij}} = \frac{d(j)}{d(i)} \tag{1.72}$$

Dans le cas d'un réseau d-régulier, où tous les nœuds ont le même nombre d de voisins, on peut sans problème fixer $\forall i, j, \alpha_{ij} \equiv \alpha$, avec $0 \leq \alpha \leq 1$. Lorsque le réseau n'est pas régulier, on peut prendre $\alpha_{ij} = d(i)/N^2$. Cependant, cela requiert que chaque noeud émetteur ait accès au nombre de nœuds du réseau, hypothèse qui nuit au caractère décentralisé de notre approche. Une autre possibilité est $\alpha_{ij} = \min\{1, \frac{d(i)}{d(j)}\}$, mais cette fois ci, les nœuds doivent connaître les degrés de leur voisins, avec des conséquences néfastes sur la décentralisation.

Correction d'asymétrie

$\mathbb{E}\mathbf{K}$ peut être doublement stochastique sans pour autant être symétrique. A partir de toute matrice stochastique irréductible, on peut en effet construire une matrice doublement stochastique en renormalisant ses colonnes. Supposons que $\mathbb{E}\mathbf{K}$ ait pour vecteur propre gauche dominant $\mathbf{v}$, c'est à dire, $\mathbf{v}^\top\mathbb{E}\mathbf{K} = \mathbf{v}^\top$. Il nous suffit alors de remplacer $\mathbf{K}(t)$ par $\mathbf{K}'(t)$ où $\mathbf{K}'_{ij}(t) = \frac{v_i}{v_j}\mathbf{K}_{ij}(t)$. En effet,

$$(\mathbf{1}^\top\mathbb{E}\mathbf{K}'(t))_i = \sum_{k=1}^{N} \frac{v_k}{v_i}\mathbb{E}\mathbf{K} = \frac{v_i}{v_i} = 1, \tag{1.73}$$

donc $\mathbb{E}\mathbf{K}'$ est doublement-stochastique. Pour que cette stratégie soit praticable, remarquons cependant que les nœuds doivent avoir accès à la distribution stationnaire $\mathbf{v}$, ce qui est non-trivial à obtenir de manière totalement décentralisée, puisque c'est une caractéristique globale du réseau.

Par conséquent, l'utilisation de la moyenne temporelle des estimées ne permet d'exploiter des matrices de communication doublement-stochastiques seulement en espérance que lorsque le graphe du réseau est régulier. Autrement, la décentralisation est non-triviale à assurer.

1.6.2 Protocoles Sum-Weight

Une autre approche considère des matrices $\mathbf{K}(t)$ simplement stochastiques. $(\mathbf{K}(t))_{t\geq 1}$ reste donc une chaîne de Markov (inhomogène). Définissons $\mathbf{P}(t) \equiv \prod_{\tau=1}^{t} \mathbf{K}(t)$. Sous l'hypothèse que $\mathbb{E}\mathbf{K}$ est primitive, on peut alors montrer que $(\mathbf{K}(t))_{t\geq 1}$ est faiblement ergodique :

Theorème 5. *Soit $(\mathbf{K}(t))_{t\geq 1}$ une suite de matrices aléatoires iid telles que*

$$\forall t, \mathbf{K}(t)\mathbf{1} = \mathbf{1}, \quad \mathbb{E}\mathbf{K}(t) \text{ est primitive} \tag{1.74}$$

alors $(\mathbf{K}(t))_{t\geq 1}$ est faiblement ergodique, c'est à dire que $\mu_S(\mathbf{P}(t)) \underset{a.s.}{\to} 0$. En particulier,

$$\forall \varepsilon > 0, \forall t, \quad \mathbb{P}[\mu_S(\mathbf{P}(t)) > \varepsilon] \leq \varepsilon^{-1} N \lambda_2^{t/2} (\mathbb{E}[\mathbf{K} \otimes \mathbf{K}]) \tag{1.75}$$

Ce théorème est prouvé en annexe A. D'après la définition 6, cela signifie qu'il existe une suite de vecteurs $(\mathbf{v}(t))_{t>0}$ telle que

$$\lim_{t\to\infty} \prod_{\tau=1}^{t} \mathbf{K}(\tau) = \mathbf{1}\mathbf{v}(t)^\top \tag{1.76}$$

En 2003, Kempe *et al* [145] remarquent alors que si l'on dote chaque noeud d'une seconde estimée w_i (appelée "poids"), évoluant selon le même processus $\mathbf{w}(t+1)^\top = \mathbf{w}(t)^\top \mathbf{K}(t+1)$ où $\mathbf{w}(t) \equiv (w_1, \ldots, w_N)$, leur quotient converge vers une valeur stable $\bar{x}$ qui correspond à une moyenne des x_i pondérés par les w_i. En effet,

$$\lim_{t\to\infty} \frac{\mathbf{x}(t)^\top}{\mathbf{w}(t)^\top} = \lim_{t\to\infty} \frac{\mathbf{x}(0)^\top \mathbf{P}(t)}{\mathbf{w}(0)^\top \mathbf{P}(t)} \overset{(1.76)}{=} \frac{\mathbf{x}(0)^\top \mathbf{1}\mathbf{v}(t)^\top}{\mathbf{w}(0)^\top \mathbf{1}\mathbf{v}(t)^\top} = \frac{\sum_{i=1}^{N} x_i(0)}{\sum_{i=1}^{N} w_i(0)} = \bar{x}, \tag{1.77}$$

où $\frac{\mathbf{a}}{\mathbf{b}}$ désigne la division composante par composante des vecteurs $\mathbf{a}$ et $\mathbf{b}$. Notons que par le théorème de Perron-Frobenius $\mathbf{v}(t) > \mathbf{0}$, donc la limite est bien définie dès lors que $\sum_i^N w_i(0) \neq 0$. On constate ainsi que même si aucune des deux estimées x_i et w_i ne se stabilise car $\mathbf{v}(t)$ ne tend pas vers un vecteur constant, ce dernier disparaît dans le quotient. Le protocole obtenu converge ainsi vers la moyenne $\bar{x}$ en utilisant uniquement des matrices stochastiques (et non doublement stochastiques). Remarquons que de telles matrices peuvent se mettre sans difficulté sous forme triangulaire par renumérotation, c'est à dire qu'à chaque étape un noeud est soit émetteur, soit récepteur, soit neutre. On a donc un protocole *push only* qui peut fonctionner de manière décentralisée et asynchrone via l'utilisation de deux estimées en chaque noeud : une somme $x_i(t)$ et un poids $w_i(t)$. C'est pourquoi on qualifie ce type de protocole de "*Sum-Weight*" [130].

Vitesse de convergence

Caractériser le taux de convergence d'un protocole Sum-Weight apparaît plus délicat que dans le cas de protocoles doublement stochastiques. On ne peut notamment pas s'appuyer sur la convergence de $\mathbf{P}(t)$ vers une matrice constante, puisque son premier vecteur propre gauche $\mathbf{v}(t)$ varie librement sans se stabiliser. On peut toutefois reformuler l'erreur d'estimation de la moyenne grâce au lemme suivant (prouvé en annexe A) :

Lemme 1. *Soit une suite $(\mathbf{K}(t))_{t\geq 1}$ de matrices aléatoires stochastiques et iid telles que $\mathbb{E}\mathbf{K}$ est primitive. Définissons les estimées $\mathbf{x}(t) = (x_1(t), \ldots, x_N(t))$ et $\mathbf{w}(t) = (w_1(t), \ldots, w_N(t))$ par*

$$\mathbf{x}(t+1)^\top = \mathbf{x}(t)^\top \mathbf{K}(t+1) \quad \textit{et} \quad \mathbf{w}(t+1)^\top = \mathbf{w}(t)^\top \mathbf{K}(t+1)$$

et l'erreur d'estimation $E(t)$ par

$$E(t) = \left\| \frac{\mathbf{x}(t)}{\mathbf{w}(t)} - \bar{x}\mathbf{1} \right\|_2 \qquad \textit{avec} \qquad \bar{x} = \frac{\sum_{i=1}^N x_i(0)}{\sum_{i=1}^N w_i(0)} \tag{1.78}$$

On a alors

$$E(t) \leq \frac{2\mu_S(\mathbf{P}(t))}{\min_{ij} \mathbf{P}_{ij}(t)} \|\bar{x}\mathbf{1}\|_2 \tag{1.79}$$

où $\mu_S(\mathbf{A})$ est le coefficient d'ergodicité de Dobrushin de la matrice $\mathbf{A}$.

L'ergodicité faible de $(\mathbf{K}(t))_{t\geq 1}$ entraîne que le numérateur tend exponentiellement vers 0. On voit apparaître au dénominateur l'entrée minimale de $\mathbf{P}(t)$. Observons que l'on peut tout à fait construire une matrice $\mathbf{P}(t)$ de rang 1 possédant des entrées nulles. Pour une telle matrice, on a alors $\mu_S(\mathbf{P}(t)) = 0$ mais aussi $\min_{ij} \mathbf{P}_{ij}(t) = 0$. Dans ce cas, puisque le rang de $\mathbf{P}(t)$ impose que toutes les lignes sont égales, il existe un ou plusieurs i tels que

$$\forall j, \mathbf{P}_{ji}(t) = 0 \quad \Leftrightarrow \quad w_i(t) = 0 \tag{1.80}$$

Les estimées x_i/w_i correspondantes sont alors indéterminées et l'erreur (1.78) est donc indéfinie. On peut se convaincre que ce cas dégénéré est susceptible de survenir asymptotiquement en considérant un protocole où un certain noeud i n'est jamais choisi comme récepteur. $\mu_S(\mathbf{P}(t))$ décroit alors exponentiellement mais $\min_j \mathbf{P}_{ij}(t)$ également. La simple ergodicité faible n'est donc pas suffisante pour garantir la convergence vers le consensus.

Cependant, nous avons formulé l'hypothèse additionnelle que $\mathbb{E}\mathbf{K}$ est primitive. Il existe donc un entier m tel que $\mathbb{E}\mathbf{K}^m > \mathbf{0}$. Les $\mathbf{K}(t)$ étant indépendantes, ceci implique qu'il existe une réalisation de $\mathbf{P}(m)$ strictement positive qui survient avec une probabilité non nulle. En rappelant que les $\mathbf{K}(t)$ sont iid, appliquons le lemme de Borel-Cantelli :

Lemme 2. *(Borel-Cantelli) Soit $(E_i)_{i=1}^\infty$ une suite d'événements aléatoires indépendants.*

$$\textit{Si} \quad \sum_{i=1}^{\infty} \mathbb{P}[E_i] = \infty \quad \textit{alors} \quad \mathbb{P}[\limsup_{i\to\infty} E_i] = 1 \tag{1.81}$$

Autrement dit, si la somme des probabilités des E_i diverge vers l'infini, alors cet événement survient infiniment souvent avec une probabilité 1. Dans notre cas, on a $\forall t, \mathbb{P}[\mathbf{P}(t+m) > \mathbf{0}] \equiv p > 0$, où p ne dépend pas de t. Donc $\sum_{t=1}^{\infty} \mathbb{P}[\mathbf{P}(t) > \mathbf{0}] = \infty$, ce qui implique que

$$\exists m > 0 : \quad \lim_{t \to \infty} \mathbb{P}[\min \mathbf{P}_{ij}(t) \geq m] = 1 \tag{1.82}$$

La probabilité que (1.78) soit indéterminée est donc asymptotiquement égale à 0 et la convergence est garantie.

Temps de convergence non-asymptotique

Cette analyse de convergence est seulement asymptotique. Elle n'est exploitable que lorsque l'on souhaite analyser le comportement du protocole sur une longue période dans le but d'obtenir une précision d'estimation infinitésimale. Une analyse asymptotique ne nous donne pas d'indication précise du temps nécessaire pour obtenir une erreur donnée (temps d'ε-convergence). Or, comme nous le verrons dans le chapitre suivant, il peut s'avérer nécessaire de disposer d'un nombre explicite de messages à échanger (*i.e.*, d'itérations) pour atteindre une erreur fixée ε. Une telle prédiction ne peut être issue que d'une analyse non-asymptotique.

La plupart des travaux théoriques autour des protocoles Gossip asynchrones se contentent pourtant d'une analyse asymptotique. Dans [18], la convergence est prouvée mais son taux n'est pas explicité. Dans [130], les auteurs formulent une borne sur le taux de convergence asymptotique, mais celle-ci ne permet pas de borner explicitement le temps (fini) à partir duquel l'erreur d'estimation passe durablement sous un seuil donné ε. Dans [145], une borne probabiliste explicite est donnée sur le temps d'ε-convergence, mais ce résultat est formulé en considérant un modèle de temps globalement synchrone où tous les nœuds émettent un message à chaque itération. Dans [87], les auteurs caractérisent asymptotiquement l'espérance de l'erreur quadratique entre les estimées et la moyenne, mais ces résultats ne s'appliquent pas aux protocoles *Sum-Weight*.

Par ailleurs, l'interêt pratique d'une analyse sur la variable t est souvent limité, car deux matrices de communication $\mathbf{K}(t)$ peuvent cacher des complexités de calcul (et donc des temps d'execution) très différentes. Une matrice peut en effet représenter plusieurs échanges de messages simultanés. Dans notre cas, il est plus pertinent de considerer le nombre de messages total échangé sur le réseau plutôt que le temps t. Un algorithme donné sera donc d'autant plus efficace qu'il envoie peu de messages pour un même resultat, ceci indépendament de l'évolution de t. De plus, nous considérons dans cette thèse des protocoles asynchrones, que nous pouvons interpreter formellement par une probabilité nulle d'avoir deux messages échangés simultanément. Notons d'ailleurs que toute matrice semblable à une matrice triangulaire à une permutation près (c'est à dire correspondant à un graphe de communication sans boucle) peut s'écrire comme un produit de n matrices n'ayant qu'un seul élément hors-diagonal non-nul, où n est inférieur au nombre de ses élements hors-diagonaux non-nuls. Tout protocole Gossip asynchrone peut donc être réécrit de sorte à ce que t corresponde exactement au nombre de messages échangés, sans en affecter l'analyse de coût. Dans la suite de cette thèse, nous adoptons donc cette démarche en considérant des matrices de communication aléatoires possédant un seul élément hors-diagonal non nul.

Pour obtenir une borne non-asymptotique à partir de (1.79), il nous faut trouver une borne supérieure sur $\mu_S(\mathbf{P}(t))$ et une borne inférieure sur $\min_j \mathbf{P}_{ji}(t)$ qui soient explicites. Pour sim-

plifier la formulation de cette borne, nous faisons ici l'hypothèse que les $\mathbf{K}(t)$ dont i.i.d., hypothèse qui couvre l'ensemble des cas pratiques abordés dans cette thèse. Le réseau est ainsi supposé stationnaire (sa topologie n'evolue pas dans le temps).

Le théorème suivant expose un des résultats principaux de cette thèse, à savoir une borne non-asymptotique sur le temps d'ε-convergence des protocoles Gossip Sum-Weight :

Theorème 6. ***Convergence non-asymptotique des protocoles Gossip Sum-Weight.***

Étant données une probabilité δ et une erreur $\varepsilon > 0$, pour tout t tel que

$$t \geq \max\left\{ \frac{3\ln N - \ln\varepsilon - 2C\ln\chi - \ln\frac{\delta+1}{2}}{-\ln\lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top])}, C \right\} \quad \textit{avec} \quad C = \frac{16N(\ln N+1)}{(1-\delta)(1-\lambda_2(\mathcal{G}))}$$

on a

$$\mathbb{P}\left[\frac{\left\| \frac{\mathbf{x}(t)}{\mathbf{w}(t)} - \bar{x}\mathbf{1} \right\|_2^2}{\|\bar{x}\mathbf{1}\|_2^2} \leq \varepsilon \right] \geq \delta,$$

où χ est la plus petite entrée non-nulle de toute réalisation de $\mathbf{K}(t)$ et $\lambda_2(\mathcal{G})$ est la 2ème plus petite valeur propre du Laplacien du graphe $\mathcal{G}$.

1.6.3 AGAVG : Asynchronous Gossip Averaging

Les algorithmes décentralisés et asynchrones proposés dans cette thèse s'appuient sur un cas particulier de protocole Gossip asynchrone de type Sum-Weight, que nous nommons Asynchronous Gossip Averaging (AGAVG). AGAVG utilise un modèle de temps Poissonnien totalement asynchrone, c'est à dire :

- Chaque noeud obéit à sa propre horloge sans coordination globale ou locale.
- Les liens sont activés indépendamment les uns des autres
- La probabilité que deux liens soient actifs simultanément est nulle.

AGAVG est de type *Sum-Weight* : pour assurer la convergence vers un consensus, nous utilisons deux estimées s_i (somme) et w_i (poids) en chaque noeud i. AGAVG s'exprime simplement par la mise à jour à chaque instant d'un unique émetteur i et d'un unique récepteur j :

$$x_i(t+1) = \frac{1}{2}x_i(t) \qquad w_i(t+1) = \frac{1}{2}w_i(t) \tag{1.83}$$

$$x_j(t+1) = x_j(t) + \frac{1}{2}x_i(t) \qquad w_j(t+1) = w_j(t) + \frac{1}{2}w_i(t) \tag{1.84}$$

où (i,j) est sélectionné aléatoirement parmi les arêtes du réseau. Les matrices de communication $\mathbf{K}(t)$ appliquées aux x_i comme aux w_i ont ainsi la forme suivante :

$$\mathbf{K}(t) = \mathbf{I} - \frac{1}{2}\mathbf{e}_i(\mathbf{e}_i - \mathbf{e}_j)^\top, \tag{1.85}$$

Algorithme 2 AGAVG : Asynchronous Gossip Averaging *(en chaque noeud i)*

Entrées : $x_i \in \mathcal{V}, \quad w_i \in \mathbb{R}$

∘ Procédure émission

$s_i \leftarrow w_i x_i$
Boucle
  $j \leftarrow$ noeud voisin aléatoire
  $s_i \leftarrow s_i/2$
  $w_i \leftarrow w_i/2$
  Envoyer (s_i, w_i) à j
Fin Boucle

∘ Procédure réception

Boucle
  Attendre réception d'un message (s, w)
  $s_i \leftarrow s_i + s$
  $w_i \leftarrow w_i + w$
Fin Boucle

Sous cette forme, remarquons que puisqu'une seule entrée hors diagonale (i, j) est non-nulle à chaque instant t, le protocole n'est pas doublement stochastique. AGAVG est donc bien de type *push-only*, garantissant ainsi que les nœuds ne s'attendent jamais les uns les autres. Remarquons que t correspond alors exactement au nombre de message envoyés à travers le réseau. Dans cette thèse nous confondrons donc ces deux notions.

Ainsi défini, AGAVG consiste en deux procédures exécutées concurremment en chaque noeud i (*cf.* Algorithme 2). La procédure d'émission sélectionne régulièrement un noeud voisin aléatoire et lui envoie la moitié de sa somme s_i et de son poids w_i tout en gardant les moitiés restantes. En parallèle, la procédure de réception traite les messages provenant des nœuds voisins en ajoutant chaque couple somme-poids reçu à ses propres estimées locales.

Fonctionnement asynchrone

Remarquons que la procédure de réception se contente d'additionner les estimées entrantes aux estimées locales. La commutativité de la somme apporte ici un avantage important : les messages entrants peuvent être traités dans un ordre quelconque. Cela signifie que

1. La convergence vers la moyenne n'est pas impactée lorsque les messages sont acheminés avec des délais imprévisibles. Les modalités de routage de l'information sur le réseau n'ont ainsi aucune influence sur la convergence.

2. Tant qu'un noeud i n'envoie pas de message, il peut se contenter de stocker les messages entrants en mémoire. L'accumulation effective des estimées entrantes peut se faire immédiatement avant de diviser ses estimées locales par 2.

3. Cette mémoire n'a pas besoin d'être ordonnée : les estimées entrantes peuvent être stockées comme un ensemble $\{(s_j, w_j)\}$ plutôt que comme une séquence $(s_j(t), w_j(t))_t$.

4. Cette mémoire ne réside pas obligatoirement dans le noeud récepteur, mais peut tout à fait être stockée par l'émetteur lui-même, ou une quelconque entité tierce capable de délivrer le message au récepteur. Ceci permet notamment de prendre en compte l'éventuelle indisponibilité du noeud récepteur due à une déconnexion temporaire. Notons que la stratégie d'acheminement de messages choisie n'a aucune influence sur le comportement de l'émetteur, qui poursuit son exécution quelle que soit la manière dont le message est acheminé au récepteur et sa durée d'acheminement.

Synchronisation interne

Observons toutefois qu'il est crucial qu'avant d'envoyer un message, un noeud accumule préalablement toutes les estimées mises en attente depuis sa dernière émission.

Cette hypothèse induit ainsi une contrainte de synchronisation entre les deux procédures concurrentes : à chaque itération de la boucle d'émission, la boucle de réception doit avoir traité l'ensemble des messages éventuellement stockés en mémoire. Par ailleurs, les itérations des deux boucles doivent être atomiques (mutuellement exclusives). En effet, les lignes 4-6 de la procédure d'émission et les lignes 3-4 de la procédure de réception ne doivent jamais être exécutées simultanément. Autrement, le comportement de la mise à jour concurrente des estimées locales (s_i, w_i) serait mal défini.

En dépit de ces contraintes de synchronisation internes aux nœuds, le réseau reste globalement asynchrone puisque aucune opération ne fait intervenir d'attente entre nœuds. Cette synchronisation entre les deux procédures locales n'a donc pas d'influence implicite sur la synchronisation globale. AGAVG respecte donc les contraintes de décentralisation et d'asynchronisme définies en introduction du présent manuscrit.

Tolérance aux fautes

Grâce à l'asynchronisme, les nœuds émetteurs n'ont pas à se soucier de la disponibilité des nœuds récepteurs. Ainsi, si un noeud quitte temporairement le réseau alors qu'il est destinataire d'un message, l'émetteur n'est pas bloqué, à la différence des protocoles *push-pull* où l'émetteur ne peut pas poursuivre au risque de fausser le consensus. Notons que si un message n'est pas acheminé du fait d'une rupture de connexion ou du départ brutal d'un noeud, deux cas peuvent se présenter. Dans le cas défavorable, le message est définitivement perdu, auquel cas le consensus est endommagé (la somme des estimées n'est plus conservée). Dans le cas favorable, une des entités du réseau (noeud, éléments de routage réseau, *etc*) peut détecter l'indisponibilité du lien ou du destinataire concernés et rediriger le message vers un autre noeud quelconque. Le consensus est alors conservé comme si l'événement n'était pas survenu.

Considérons par exemple une implémentation basée sur un protocole réseau connecté (type TCP). Dans ce protocole, les nœuds récepteurs notifient systématiquement les émetteurs de la bonne réception de l'information. Ainsi, il suffit à l'émetteur de conserver le message en mémoire tant qu'il n'a pas été notifié comme reçu et de le réémettre dans le cas où il reçoit une notification d'erreur en cas de dépassement du délai de réponse. Considérons maintenant un protocole réseau non-connecté (type UDP). Les nœuds émetteurs ne reçoivent dans ce cas aucune notification de bonne réception des messages au niveau de la couche transport (UDP). Cependant, on peut ajouter à la procédure de réception l'envoi d'un accusé de réception du récepteur à l'émetteur pour reproduire le comportement de TCP.

Ici, l'avantage clé des protocoles *push-only* est qu'il n'est pas nécessaire de coordonner les échanges. En effet, l'accusé de réception associé à un message peut revenir à l'émetteur avec un délai quelconque sans ralentir le processus, l'émetteur n'ayant pas à attendre cette réponse pour poursuivre ses traitements. Il lui suffit en effet de conserver en mémoire les messages dont l'accusé de réception n'a pas encore été reçu et de les supprimer de cette mémoire temporaire à réception de la notification. Une telle désynchronisation des échanges n'est pas autorisée dans les protocoles *push-pull*, qui requièrent l'atomicité des allers-retours entre l'émetteur et le récepteur.

Par ailleurs, de nombreux travaux ont montré la robustesse des protocoles Gossip aux fautes

non-recouvrées, aux erreurs d'arrondis liées à la quantification numérique, ainsi qu'aux ralentissements et aux congestions temporaires localisés [19, 38, 73, 211, 219, 231].

1.6.4 Temps de convergence non-asymptotique d'AGAVG

En pratique, l'implémentation que nous faisons dans cette thèse des protocoles Gossip s'appuie sur des protocoles réseaux de la couche "application" (au sens de la définition OSI). Autrement dit, nous supposons généralement qu'il existe un protocole réseau plus bas niveau (couche transport) tel que TCP/IP qui assure que chaque noeud du réseau est capable de communiquer avec tout autre noeud (en passant éventuellement par des noeuds intermediaires de manière transparente) même si le graphe physique n'est pas complet. Cela se traduit par l'existence implicite d'un graphe totalement connecté qui vient completer le réseau physique par des connexions *multi-hop* (transitives). D'un point de vue théorique, nous pouvons donc nous contenter de modéliser de tels réseaux comme des graphes complets dotés de pondérations pénalisantes sur les liens virtuels, correspondant au transit par des noeuds intermédiaires. Nous présentons donc dans ce paragraphe une borne non-asymptotique sur le temps de converge d'AGAVG spécifique aux graphes complets lorsque les $\mathbf{K}(t)$ sont i.i.d. (prouvée en annexe A)

Theorème 7. ***Temps d'ε-convergence d'AGAVG sur graphe complet.***
Étant données une probabilité δ et une erreur $\varepsilon > 0$, pout tout t tel que

$$t \geq 2N\left(3\ln N + \frac{6N(\ln N + 1)}{1-\delta} - \ln\varepsilon + \ln\frac{2}{1-\delta}\right)$$

on a

$$\mathbb{P}\left[\frac{\left\|\frac{\mathbf{x}(t)}{\mathbf{w}(t)} - \bar{x}\mathbf{1}\right\|_2^2}{\|\bar{x}\mathbf{1}\|_2^2} \leq \varepsilon\right] \geq \delta$$

1.7 Protocoles Gossip non-linéaires

Les protocoles présentés jusqu'ici se sont focalisés sur l'estimation de moyenne distribuée. Dans cette section, nous montrons qu'en se basant sur le comportement de moyennage spontané des protocoles Gossip, il est possible d'étendre ce paradigme décentralisée et asynchrone à une large variété d'opérations non-linéaires.

1.7.1 Stratégies par pré-encodage

Tout d'abord, il est possible de calculer certaines opérations non-linéaires au moyen d'un protocole de moyennage Gossip en pré-encodant les données d'entrées de sorte à ce que le problème devienne linéaire dans l'espace induit, puis de décoder le résultat obtenu à convergence, pour peu que l'encodage soit réversible. L'exemple le plus simple est celui du calcul décentralisé de produits ou de moyennes géométriques. Supposons que l'on cherche à calculer la moyenne géométrique de N réels $\mathbf{x} = (x_i)_{i=1}^N$, chacun étant hébergé par un noeud du réseau. Rappelons que

celle-ci est définie par :

$$\nu = \left(\prod_{i=1}^{N} x_i \right)^{\frac{1}{N}} \tag{1.86}$$

Cette opération est non-linéaire dans l'espace de départ, mais en écrivant $y_i = \ln x_i$, on a

$$\ln \nu = \ln \left(\prod_{i=1}^{N} x_i \right)^{\frac{1}{N}} = \frac{1}{N} \sum_{i=1}^{N} \ln x_i = \frac{1}{N} \sum_{i=1}^{N} y_i \equiv \bar{y} \tag{1.87}$$

Ainsi, il nous suffit de calculer la moyenne (arithmétique) $\bar{y}$ des y_i pour obtenir en chaque noeud le logarithme du résultat recherché. Chaque noeud peut alors calculer localement $\nu = \exp(\bar{y})$ pour obtenir le résultat recherché.

Cet exemple simple peut être généralisé à l'estimation de moments statistiques d'ordres arbitraires, l'évaluation de noyaux non-linéaires, jusqu'à la résolution de systèmes linéaires et l'analyse en composante principale (*cf* chapitre 3). Tant que les données peuvent être ré-exprimées localement dans un espace où le résultat en devient un combinaison linéaire et que ce dernier peut être re-traduit dans l'espace de départ, le problème peut être directement résolu de manière décentralisée et asynchrone, avec notamment des garanties de convergence équivalentes au cas linéaire.

1.7.2 Orchestration centralisée

Lorsque le problème considéré ne peut pas directement être mis sous forme de moyenne dans un espace induit, la stratégie par encodage est inutilisable. C'est le cas par exemple lorsque la résolution du problème nécessite d'alterner une ou plusieurs opérations non-linéaires et une ou plusieurs opérations linéaires de manière répétée. La nature itérative du processus et la présence d'opérations non-linéaires au sein d'une itération empêche toute reformulation locale comme une simple moyenne dans un espace induit.

Dans une telle situation, la plupart de systèmes de calcul Gossip existants adoptent une stratégie globalement synchrone. Le processus de resolution itératif est alors orchestré de manière centralisée par une entité maîtresse selon le schéma algorithmique suivant :

1. Exécuter l'opération non-linéaire de manière centralisée (via un noeud maître)
2. Réaliser la combinaison linéaire de manière décentralisée (*e.g.*, via un protocole Gossip).
3. Retourner en 1.

De fait, l'avantage de la décentralisation est totalement détruit par cette organisation séquentielle des opérations au niveau global. On observera dans ce cas un gain de performance notable sur de petits réseaux, mais ce gain tend à se réduire à mesure qu'on considère de plus larges réseau, l'avantage computationnel offert par la décentralisation de l'étape 2 étant compensé par le ralentissement causé par la centralisation de l'opération non-linéaire et de l'orchestration elle-même.

Si l'on souhaite éviter les écueils liés à la centralisation et la synchronisation globale (même implicite), il nous faut adopter une approche totalement décentralisée et asynchrone, se passant de chef d'orchestre global. Le processus itératif doit alors être défini au niveau des nœuds eux-même (et non au niveau global du réseau) sans coordination entre nœuds.

1.7.3 Protocoles Gossip perturbés

Pour résoudre des problèmes non-linéaires de manière totalement décentralisée et asynchrone, nous nous appuyons dans cette thèse sur un formalisme que nous appelons "protocoles Gossip perturbés". Un protocole Gossip perturbé est défini par le système dynamique suivant :

$$\forall i, \quad x_i(t+1) = \sum_{j=1}^{N} \mathbf{K}_{ji}(t+1)x_j(t) + f_i(t, x_i(t), \dots, x_i(0)) \tag{1.88}$$

On reconnais dans le terme de gauche un protocole Gossip linéaire classique, avec

$$\mathbf{K}(t)\mathbf{1} = \mathbf{K}(t), \quad \mathbf{K}(t) \text{ iid} \quad \text{et} \quad \mathbb{E}\mathbf{K} \text{ primitive} \tag{1.89}$$

Le terme de droite est une fonction f_i spécifique au noeud i, chaque noeud i définissant sa propre fonction f_i indépendamment des autres nœuds. On constate que f_i est une fonction du temps t et des valeurs précédentes de x_i, mais pas des valeurs des autres nœuds x_j. On dit que les f_i sont localement calculable, ce qui signifie que leur évaluation n'implique aucun échange de données entre nœuds. Notons que f_i peut être non-linéaire, aléatoire, voire non stationnaire.

Comme nous l'avons développé plus haut, le terme de gauche induit donc un comportement de moyennage spontané entre nœuds via des échanges de messages, tandis que le terme de droite vient "perturber" cette convergence vers un état stable grâce aux fonctions localement calculables f_i. C'est pourquoi on qualifie ce type de processus de *procotole Gossip perturbé*. Aux instants t où les fonctions f_i sont nulles, un tel protocole est faiblement ergodique, les instants où une des f_i étant non-nulle permettant d'influencer l'état stable vers lequel le système est susceptible de converger. A ce titre, on peut légitimement interpréter les fonctions f_i comme un moyen de "programmer" le système dynamique (1.88). Comme nous le verrons dans cette thèse, cette capacité à orienter le système par la spécification de fonctions f_i particulières nous permet de résoudre de nombreuses tâches de calcul distribué non-linéaires, en particulier dans le domaine de l'apprentissage statistique et la reconnaissance de formes.

Une propriété importante du protocole (1.88) est que toute contribution additive d'un noeud à travers sa fonction f_i est répercutée à l'identique sur le consensus. En effet, si à l'instant $t+1$, x_i est impacté par $f_i(t+1) = z$, on a

$$\begin{aligned}
\bar{x}(t+1) &= \frac{1}{N}\sum_{i}^{N} x_i(t+1) && (1.90)\\
&= \frac{1}{N}\sum_{i}^{N}\left(\sum_{j}^{N} \mathbf{K}_{ji}(t+1)x_j(t) + f_i(t+1)\right) && (1.91)\\
&= \frac{\sum_i^N x_i(t) + z}{\sum_i^N w_i(t)} && (1.92)\\
&= \bar{x} + \frac{z}{N} && (1.93)
\end{aligned}$$

D'après (1.77), suite à cette contribution, le réseau converge alors vers le nouveau consensus $\bar{x}(t+1)$. L'avantage ici est que pour impacter le consensus de manière cohérente les nœuds contributeurs n'ont pas à coordonner leurs actions.

Fonctions séparables

Si l'on reprend le schéma algorithmique de la section 1.7.2, on peut le transformer en un processus totalement décentralisé et asynchrone si :

- On peut confier à chaque noeud i une partie du calcul de l'opération non-linéaire de l'étape 1 pour en déterminer une composante partielle z_i.
- On peut identifier explicitement la contribution additive de z_i dans la combinaison linéaire de l'étape 2, sans l'agréger auparavant avec les résultats partiels des autres nœuds z_j.

Lorsqu'une fonction f satisfait ces conditions, nous la qualifierons de *séparable* :

Définition 7. *(Fonction séparable). Une fonction $f : (\mathcal{V}_1, \dots, \mathcal{V}_n) \mapsto \mathcal{V}_0$, où les $\mathcal{V}_i$ sont des espaces vectoriels, est dite "séparable" si et seulement si il existe des fonctions $(g_i)_{i=1}^n$ et $(\omega_i)_{i=1}^n$ telles que $g_i : \mathcal{V}_i \mapsto \mathcal{V}_0$ et $\omega_i : \mathcal{V}_i \mapsto \mathbb{R}$ et*

$$\forall \mathbf{x} = (x_1, \dots, x_n), x_i \in \mathcal{V}_i, \quad f(\mathbf{x}) = \frac{\sum_{i=1}^n g_i(x_i)}{\sum_{i=1}^n \omega_i(x_i)} \tag{1.94}$$

Autrement dit, une fonction à plusieurs variables est séparable si et seulement si elle peut être traduite comme une moyenne pondérée dont chaque couple terme-poids n'est fonction que d'une seule des variables.

On s'aperçoit immédiatement que toute fonction séparable dont les variables sont hébergées sur des nœuds distincts peut être calculée de manière décentralisée et asynchrone par un protocole Gossip *Sum-Weight* tel qu'AGAVG, en utilisant les g_i et les ω_i comme pré-encodeurs. En effet, en adoptant les définitions de (1.94), supposons que $x_i(0) \in \mathcal{V}_i$ soit hébergée par le noeud i. Ce noeud peut alors calculer localement $s_i(0) \equiv g_i(x_i(0))$ et $w_i(0) \equiv \omega_i(x_i(0))$ sans coordination avec les autres nœuds. Si on exécute ensuite AGAVG en prenant les s_i comme sommes et les w_i comme poids, (1.77) entraîne

$$\forall i, \quad \lim_{t\to\infty} \frac{s_i(t)}{w_i(t)} = \frac{\sum_{i=1}^n g_i(x_i)}{\sum_{i=1}^n \omega_i(x_i)} = f(\mathbf{x}) \tag{1.95}$$

Remarquons que l'on peut désynchroniser les évaluations des g_i et ω_i en débutant simplement avec $\forall i, s_i(0) = 0$ et $w_i(0)$ et en considérant que les $g_i(x_i(0))$ et les $\omega_i(w_i(0))$ sont ajoutés à des instants aléatoires $(T_1, \dots, T_N)$. On obtient alors un protocole Gossip perturbé de la forme (1.88) tel que

$$s_i(t+1) = \sum_{j=1}^N \mathbf{K}_{ji}(t+1) s_j(t) + \delta(t - T_i) g_i(x_i(0)) \tag{1.96}$$

$$w_i(t+1) = \sum_{j=1}^N \mathbf{K}_{ji}(t+1) w_j(t) + \delta(t - T_i) \omega_i(w_i(0)) \tag{1.97}$$

où $\delta(x) = 1$ si $x = 0$ et $\delta(x) = 0$ sinon. Ici, chaque noeud i ne subit qu'une seule perturbation à l'instant T_i. Si tous les T_i sont finis quasiment sûrement, on a alors

$$\forall i, \quad \mathbb{P}\left[\lim_{t\to\infty} \frac{s_i(t)}{w_i(t)} = f(\mathbf{x})\right] = 1 \tag{1.98}$$

L'avantage d'une telle approche est que le calcul des fonctions locales et le moyennage entre nœuds sont réalisés *simultanément*, et non l'un après l'autre, ceci sans coordination entre nœuds. On peut donc se passer d'une orchestration centralisée et réaliser les étapes 1 et 2 de la section 1.7.2 *en même temps*. Trois propriétés peuvent alors être mises en évidence :

1. Le temps d'évaluation locale des fonctions g_i et ω_i, qui dépend entre autres des capacités calculatoires de chaque noeud et de la complexité des fonctions elles-mêmes, ne ralentit pas le calcul des autres nœuds.

2. En un noeud donné i, l'évaluation de g_i et ω_i et le moyennage avec les autres nœuds peuvent être réalisés en parallèle, leur résultat étant intégré au consensus dès qu'il est disponible.

3. Les nœuds qui apportent une contribution locale nulle ($f_i' = 0$ et $\omega_i = 0$) ne sont pas inutiles : ils offrent une contribution à la vitesse de convergence en participant au moyennage, notamment grâce à leur connectivité. Ainsi, un noeud peut ne posséder aucune donnée de départ mais jouer un rôle clé dans le processus de calcul.

Intégration à un processus itératif

La nature alternée de l'orchestration centralisée provient de l'étanchéité entre l'étape 1 et l'étape 2. Mais si la fonction non-linéaire calculée par l'étape 1 est séparable, on peut exploiter l'astuce (1.96-1.97) pour la mêler à l'étape 2. La procédure est ainsi remplacée par un système dynamique de la forme (1.88). Les deux étapes étant réalisées simultanément, le processus global devient alors totalement asynchrone et décentralisé.

On passe ainsi d'un schéma algorithmique globalement orchestré donc sous-efficient à un système dynamique dont les entités contribuent au calcul sans se coordonner entre elles. Précisément, les nœuds contribuent au calcul en permanence, d'où une progression bien plus rapide vers le résultat. Par ailleurs, la lenteur ou la défaillance de certains nœuds n'a plus qu'un impact négligeable sur la dynamique globale. Finalement, le passage à l'échelle sur de grands réseaux ne se heurte pas à la barrière de synchronisation globale.

1.8 Réseaux à topologie dynamique

La décentralisation et l'asynchronisme apportent également aux protocoles Sum-Weight des propriétés avantageuses lorsque la connectivité du réseau est amenée à changer au cours du temps.

1.8.1 Ajout/suppression de liens

L'analyse de convergence des protocoles Gossip que nous avons présenté dans ce chapitre fait systématiquement l'hypothèse que le processus de sélection des liens est stationnaire. Cette hypothèse facilite sensiblement l'analyse, mais n'est pas une condition nécessaire. Notamment, la convergence est conservée lorsque des liens sont ajoutés ou supprimés à tout moment, tant que la connexité forte du graphe est maintenue. La chaîne de Markov inhomogène associée au protocole n'est alors plus stationnaire, mais certains outils de caractérisation de l'ergodicité, tels que les coefficients de contraction de Birkhoff [27, 49, 124] ou les coefficients d'ergodicité de Seneta [217, 218] conservent leur propriétés fondamentales (en particulier la sous-multiplicativité).

Dans un contexte non-stationnaire, la notion de matrices primitives est étendue au concept de matrices de brouillages (*scrambling matrix*) dont on peut prouver qu'elles sont faiblement ergodiques.

Il est ainsi possible de relâcher les hypothèses de stationnarité, tant que le graphe du réseau reste fortement connexe à tout instant t et que ses arêtes sont toutes sélectionnées avec une probabilité non-nulle. La convergence vers le consensus est alors toujours garantie.

Notons que dans ce cas non-stationnaire, le taux de convergence varie à chaque instant, en fonction de la seconde valeur propre de la matrice d'adjacence du graphe. Par exemple, si l'on débute avec une connectivité complète que l'on élague petit à petit au cours du temps jusqu'à obtenir un cycle, la vitesse de convergence va progressivement diminuer. Notons que pour tout intervalle où le graphe est constant à court terme, la convergence reste exponentielle.

1.8.2 Rupture de connexité

Lorsque, suite à la déconnexion d'un noeud ou la suppression d'une arête, la connexité forte du réseau est perdue, la chaîne de Markov associée au protocole Gossip n'est plus irréductible. Il est alors évident que la convergence vers le consensus est impacté, puisque il existe au moins un couple de noeuds (i, j) qui ne peut plus échanger d'informations via un chemin dans $\mathcal{G}$. On peut distinguer deux cas :

1. Le réseau reste connexe (mais non fortement connexe). Pour toute paire de noeuds $\{i, j\}$, il existe un chemin $i \xrightarrow{*} j$ ou $j \xrightarrow{*} i$, mais pas nécessairement de circuit $i \xrightarrow{*} j \xrightarrow{*} i$.

2. Le réseau n'est plus connexe. Les composantes connexes résultantes peuvent alors être vues comme autant de réseaux séparés.

Cas 1 : Ensembles absorbants

Si le réseau reste connexe mais plus fortement connexe, il existe un couple de noeuds (i, j), pour lesquels l'information peut transiter de i à j, mais ne peut plus revenir à i une fois qu'elle a rejoint j. Formellement, il existe donc un ou plusieurs sous-graphes absorbants dans $\mathcal{G}$, définis comme suit :

Définition 8. *Un sous-graphe $\mathcal{G}_1$ du graphe connexe $\mathcal{G}$ est dit absorbant si et seulement si il n'existe pas d'arête (i, j) dans $\mathcal{G}$ telle que $i \in \mathcal{G}_1$ et $j \notin \mathcal{G}_1$.*

Autrement dit, il existe un ou plusieurs ensembles de nœuds $\mathcal{A}_i$ tels qu'une marche aléatoire sur $\mathcal{G}$ rejoint un noeud d'un des $\mathcal{A}_i$ en un temps fini avec une probabilité 1 et reste "bloquée" dans ce $\mathcal{A}_i$. Si l'on utilise un protocole *push-pull*, il est clair que la convergence vers le consensus est totalement stoppée dès lors qu'un tel ensemble absorbant apparaît.

A contrario, les protocoles *push-only* ne requièrent pas d'échanges bidirectionnels. Si il existe plusieurs ensembles absorbants distincts, il est clair qu'ils accumulent chacun une quantité asymptotiquement non-nulle d'information et qu'ils ne peuvent en échanger entre eux. Il est donc évident que le protocole ne converge pas vers un consensus.

Toutefois, si il n'existe qu'un seul ensemble absorbant $\mathcal{A}$, il apparaît que toute l'information est asymptotiquement regroupée dans $\mathcal{A}$. Tous les nœuds de $\mathcal{A}$ convergent alors vers le consensus, comme le montre le théorème suivant (prouvé en annexe A) :

Theorème 8. *Soit un graphe $\mathcal{G}$ possédant un unique sous-graphe absorbant fortement connexe $\mathcal{A}$. Tout noeud i de $\mathcal{G}$ satisfait*

$$\begin{cases} \lim_{t\to\infty} \frac{x_i(t)}{w_i(t)} = \frac{\sum_{i\in\mathcal{G}} x_i(0)}{\sum_{i\in\mathcal{G}} w_i(0)} \equiv \bar{x} & \textit{si } i \textit{ est dans } \mathcal{A} \\ \lim_{t\to\infty} x_i(t) = \lim_{t\to\infty} w_i(t) = 0 & \textit{sinon} \end{cases} \tag{1.99}$$

Cette propriété dite "de Perron-Frobenius étendue" généralise donc l'ergodicité faible aux matrices réductibles lorsque le graphe associé n'a qu'une composante fortement connexe absorbante. De nombreux travaux ont considérés des extensions plus générales (*e.g.*, matrices avec certaines valeurs négatives [81, 189], réductibles [204, 229], quasi-périodiques [253], tenseurs [54], etc), permettant d'exhiber certaines propriétés de convergence partielle.

Dans ces cas étendus, seul un certain sous-ensemble de nœuds obtient la solution du problème de consensus. Les autres nœuds sont indéfinis ou ont une estimée erronée. Malgré ce phénomène, le processus global reste décentralisé, mais cette décentralisation est affaiblie car tous les nœuds participant au calcul du résultat ne l'obtiennent pas nécessairement localement. Nous nous trouvons donc dans une situation intermédiaire entre un processus totalement décentralisé et un processus totalement centralisé, qui peut correspondre à certaines configurations où l'on ne peut assurer une connexité forte, mais où l'on souhaite qu'une partie des nœuds convergent vers le résultat.

Nous mentionnons ici trois exemples typiques de cette situation :

- **Noeud maître asynchrone**. Considérons un graphe tel que tous les nœuds i soient connectés à un seul et même noeud z, qualifié de maître, et ne communiquent que dans le sens (i, z). Ce noeud constitue donc un sous graphe absorbant et on a

$$\lim_{t\to\infty} \frac{x_z(t)}{w_z(t)} = \bar{x} \quad \text{et} \quad \forall i \neq z, \lim_{t\to\infty} x_i(t) = \lim_{t\to\infty} w_i(t) = 0 \tag{1.100}$$

  On a donc un comportement similaire à un protocole classique à noeud maître, à la différence que le fonctionnement est asynchrone, c'est à dire que les nœuds contributeurs envoient leurs estimées sans coordination. L'algorithme converge alors exactement en $\mathcal{O}(N)$ messages non-nuls. Notons que cette solution n'est pertinente que si l'on fixe les coefficients non-nuls des $\mathbf{K}(t)$ à 1. Autrement la convergence exacte n'est qu'asymptotique et les messages provenant d'un même noeud sont redondants.

- **Arbre couvrant asynchrone**. Soit un graphe acyclique (un arbre) tel que les connexions soient orientés des feuilles vers la racine. La racine est ainsi absorbante et converge donc vers la moyenne $\bar{x}$. Ce protocole est par conséquent un équivalent asynchrone du protocole classique à arbre couvrant. Les feuilles sont des contributeurs purs, au sens où ils n'agrègent aucune information et ne font que transmettre leur propre donnée locale. Observons également que chaque nœud (sauf la racine) n'envoie qu'à un seul noeud (son ascendant direct dans l'arbre). Tout comme le graphe à noeud maître asynchrone, cette configuration n'est donc pertinente que si l'on fixe les coefficients non-nuls des $\mathbf{K}(t)$ à 1. L'algorithme converge alors exactement en un temps fini.

- **Connexion unidirectionnelle de plusieurs réseaux** : Considérons un ensemble de de n graphes $\{\mathcal{G}_i\}_{i=1}^n$ que nous connectons les uns aux autres selon un flot orienté, *i.e.*, $\mathcal{G}_i$ ne peut être connecté à $\mathcal{G}_j$ que si $i < j$. Notons $\mathcal{G}$ le graphe global obtenu. Clairement, $\mathcal{G}_n$

est un sous-graphe absorbant de $\mathcal{G}$. Par conséquent, seuls les nœuds de $\mathcal{G}_n$ convergent vers la moyenne globale des nœuds de $\mathcal{G}$. Notons que cette convergence est garantie quels que soient les états de convergence locale des sous-graphes $\mathcal{G}_i$ avant leur interconnexion.

Pour conclure, il n'est jamais computationnellement avantageux d'exploiter un réseau non fortement connecté, la convergence ne pouvant être que plus lente. Ce type de cas de figure ne se rencontre donc que lorsqu'il est en pratique impossible d'assurer la connexité forte, mais que l'on souhaite prendre en compte toutes les estimées initiales du réseau.

Cas 2 : Éclatement en plusieurs réseaux

Lorsque la connexité est rompue, le réseau initial éclate en plusieurs réseaux séparés. Clairement, chaque réseau résultant évolue alors indépendamment des autres. Remarquons que la somme des estimées n'est pas conservée par cet événement. En effet, en supposant que le réseau $\mathcal{G}$ éclate à l'instant T en n réseaux $\{\mathcal{G}_i\}_{i=1}^n$ avec $\bigcup_{i=1}^n \mathcal{G}_i = \mathcal{G}$, la somme des x_i et la somme des w_i n'écrivent respectivement

$$\sum_{i\in\mathcal{G}} x_i(T) = \sum_{i=1}^{n}\sum_{j\in\mathcal{G}_i} x_j(T) \quad \text{et} \quad \sum_{i\in\mathcal{G}} w_i(T) = \sum_{i=1}^{n}\sum_{j\in\mathcal{G}_i} w_j(T) \tag{1.101}$$

Dans le cas général, la moyenne $\bar{x} \equiv \frac{\sum_{i\in\mathcal{G}} x_i(0)}{\sum_{i\in\mathcal{G}} w_i(0)}$ n'est donc pas conservée en chaque sous-réseau. Puisque chacun évolue indépendamment à partir de l'instant T, il n'y a donc pas de raison que les nœuds de ces sous-réseaux distincts convergent vers le même consensus $\bar{x}$. Chaque sous-réseau $\mathcal{G}_i$ converge en effet vers son propre consensus local $\bar{x}^{(i)}$ défini par

$$\bar{x}^{(i)} = \frac{\sum_{j\in\mathcal{G}_i} x_j(T)}{\sum_{j\in\mathcal{G}_i} w_j(T)} \tag{1.102}$$

On peut néanmoins caractériser l'erreur d'estimation du consensus par chaque sous-réseau en fonction de l'état de convergence de $\mathcal{G}$ avant son éclatement.

Supposons tout d'abord que le réseau ait convergé avant son éclatement :

$$\exists \mathbf{v} : \quad \mathbf{v}^\top \mathbf{1} = \mathbf{1}, \quad \mathbf{x}(T)^\top = \mathbf{x}(0)^\top \mathbf{1}\mathbf{v}^\top \quad \text{et} \quad \mathbf{w}(T)^\top = \mathbf{w}(0)^\top \mathbf{1}\mathbf{v}^\top \tag{1.103}$$

Pour tout sous-réseau $\mathcal{G}_i$, on a alors

$$\frac{\sum_{j\in\mathcal{G}_i} x_j(T)}{\sum_{j\in\mathcal{G}_i} w_j(T)} = \frac{\sum_{j\in\mathcal{G}_i} v_j \sum_{k=1}^{N} x_k(0)}{\sum_{j\in\mathcal{G}_i} v_j \sum_{k=1}^{N} w_k(0)} = \frac{\sum_{k=1}^{N} x_k(0)}{\sum_{k=1}^{N} w_k(0)} \equiv \bar{x} \tag{1.104}$$

Ainsi, bien que la somme des estimées soit perdue en chaque sous-réseau, le consensus $\bar{x}$ est quant à lui parfaitement conservé. Chaque sous-réseau converge donc vers le même consensus. On remarque d'ailleurs que chaque réseau est déjà dans un état de convergence optimale à T. L'éclatement en plusieurs sous-réseaux n'a donc dans ce cas aucun effet sur la convergence.

Supposons maintenant une convergence incomplète de $\mathcal{G}$ avant son éclatement, c'est à dire que l'erreur d'estimation du consensus en chaque noeud soit au maximum $\varepsilon > 0$:

$$\forall i \in \mathcal{G}, \quad \left| \frac{x_i(T)}{w_i(T)} - \bar{x} \right| \leq \varepsilon \tag{1.105}$$

Que pouvons-nous dire de l'erreur d'estimation du consensus à convergence des sous-réseaux résultants de l'éclatement ? En un sous-réseau $\mathcal{G}_i$ quelconque, cette erreur est donnée par

$$E_i \equiv \left|\bar{x}^{(i)} - \bar{x}\right| = \left|\frac{\sum_{j\in\mathcal{G}_i} x_j(T)}{\sum_{j\in\mathcal{G}_i} w_j(T)} - \bar{x}\right| = \frac{\left|\sum_{j\in\mathcal{G}_i} x_j(T) - \sum_{j\in\mathcal{G}_i} w_j(T)\bar{x}\right|}{\sum_{j\in\mathcal{G}_i} w_j(T)} \tag{1.106}$$

$$= \frac{\left|\sum_{j\in\mathcal{G}_i} \left(x_j(T) - w_j(T)\bar{x}\right)\right|}{\sum_{j\in\mathcal{G}_i} w_j(T)} = \frac{\left|\sum_{j\in\mathcal{G}_i} w_j(T)\left(\frac{x_j(T)}{w_j(T)} - \bar{x}\right)\right|}{\sum_{j\in\mathcal{G}_i} w_j(T)} \tag{1.107}$$

$$\leq \frac{\sum_{j\in\mathcal{G}_i} w_j(T)\varepsilon}{\sum_{j\in\mathcal{G}_i} w_j(T)} = \varepsilon \tag{1.108}$$

Par conséquent, l'erreur d'estimation en un noeud quelconque du réseau ne peut jamais s'accroître du fait de l'éclatement en plusieurs sous-réseaux séparés. Néanmoins, le consensus exact est perdu.

1.8.3 Ajout/suppression de nœuds

Un des avantages des protocoles Gossip tient dans leur nature incrémentale. En effet, si à l'instant T on ajoute un noeud $N+1$ contribuant les estimées $(x_{N+1}(T), w_{N+1}(T))$, le consensus $\bar{x}(T)$ devient

$$\bar{x}(T) = \frac{\sum_{i=1}^{N+1} x_i(T)}{\sum_{i=1}^{N+1} w_i(T)} = \frac{\sum_{i=1}^{N} x_i(0) + x_{N+1}(T)}{\sum_{i=1}^{N} w_i(0) + w_{N+1}(T)} \tag{1.109}$$

Autrement dit, à tout instant t, le consensus $\bar{x}(t)$ est égal à la moyenne des contributions initiales des nœuds investis dans le réseau à cet instant. Une conséquence intéressante est qu'il n'est pas nécessaire de disposer de la totalité du réseau au lancement du processus. On peut tout à fait débuter avec un réseau à deux nœuds et leur adjoindre des nœuds supplémentaires au fil du temps, selon les besoins. Le comportement asymptotique du processus n'est impacté ni par l'ordre ni par la répartition des instants auxquels les nœuds sont ajoutés, sous réserve que le dernier noeud soit ajouté à un instant fini.

De son côté, la suppression d'un nœud peut entraîner des conséquences très différentes selon les cas :

- Si le noeud disparaît brutalement (erreur fatale, extinction soudaine, etc), ses estimées ne contribuent plus au consensus. Supposons (sans perte de généralité) que le noeud N quitte ainsi le réseau à l'instant T. Dans le même esprit que (1.108), supposons qu'à cet instant

l'erreur d'estimation de tous les autres nœuds soit bornée par ε. On a alors

$$|\bar{x}(T) - \bar{x}(T-1)| = \left| \frac{\sum_{i=1}^{N-1} x_i(T)}{\sum_{i=1}^{N-1} w_i(T)} - \bar{x}(T-1) \right| \tag{1.110}$$

$$= \frac{\left| \sum_{i=1}^{N-1} x_i(T) - \sum_{i=1}^{N-1} w_i(T)\bar{x}(T-1) \right|}{\sum_{i=1}^{N-1} w_i(T)} \tag{1.111}$$

$$= \frac{\left| \sum_{i=1}^{N-1} w_i(T)(\frac{x_i(T)}{w_i(T)} - \bar{x}(T-1)) \right|}{\sum_{i=1}^{N-1} w_i(T)} \tag{1.112}$$

$$\leq \frac{\sum_{i=1}^{N-1} w_i(T)\varepsilon}{\sum_{i=1}^{N-1} w_i(T)} = \varepsilon \tag{1.113}$$

Autrement dit, le consensus ne dévie pas de plus d'ε.

- Si le noeud quitte volontairement le réseau, il lui suffit d'envoyer la totalité de ses estimées (au lieu de la moitié) dans son dernier message. Ses estimées locales sont alors annulées, il peut donc quitter le réseau sans conséquence sur le consensus. Tant que la connexité forte est maintenue, le réseau converge ainsi vers la même moyenne, à un taux toutefois différent (celui-ci peut être plus rapide ou plus lent selon l'implication du noeud supprimé dans la connectivité du réseau initial).

- Si la connexité forte est rompue mais le réseau reste connexe, on tombe dans le cas des ensembles absorbants traité plus haut : le consensus est conservé mais seul un sous-ensemble des nœuds convergent vers la celui-ci.

- Si la connexité est perdue, le réseau éclate en plusieurs sous-réseaux, chacun convergeant vers son propre consensus. Comme mentionné plus haut, si le réseau initial avait déjà convergé, cela n'a pas de conséquence, tandis que si la convergence n'était que partielle, le consensus n'est qu'approximatif.

Pour conclure cette section, nous observons donc que les protocoles Gossip tolèrent remarquablement les changements de topologie de réseau. Ils s'adaptent en particulier de manière transparente à l'ajout de nœuds et à l'union de plusieurs réseaux. Toutefois les modifications soustractives (suppression de liens ou de nœuds, séparation du réseau en sous-réseaux) doivent être opérées avec précautions pour ne pas induire d'erreur significative dans l'estimation du consensus.

1.9 Conclusion

Dans ce chapitre, nous avons présenté un outil théorique de calcul de moyennes distribuées nommé "protocoles Gossip". Ces protocoles Gossip s'appuient sur les propriétés d'ergodicité d'une certaine chaîne de Markov (ou de manière équivalente d'un produit de matrices stochastiques) pour converger itérativement vers un état de consensus, où tous les nœuds du réseau de calcul obtiennent asymptotiquement la moyenne des estimées initialement contribuées par chacun d'eux.

Nous avons montré les limitations des protocoles synchrones qui entraînent une synchronisation globale implicite, même lorsque la coordination est seulement locale. Ces considérations nous ont amenés à considerer une extension asynchrone des protocoles Gossip. Les protocoles Gossip asynchrones ne peuvent s'appuyer sur la propriété d'ergodicité forte applicable aux matrices doublement stochastiques. Il ne sont ainsi que faiblement ergodiques, *i.e.*, ils ne convergent pas vers une distribution stationnaire. Nous avons présenté deux moyens de contourner ce problème. La première solution est de conserver une moyenne temporelle des estimées locales, la loi des grands nombre nous garantissant alors la convergence de cette moyenne vers son espérance, dont on peut assurer la convergence vers un état stationnaire. Une alternative est de considérer un couple d'estimées (une somme et un poids) qui évoluent selon la même chaîne de Markov. On parle de protocole Sum-Weight. Nous montrons alors que ce type de protocole converge vers un consensus avec un taux identique au cas synchrone. En particulier, nous proposons une borne non-asymptotique sur le temps d'ε-convergence qui fait intervenir les propriétés spectrales du graphe associé au réseau.

Nous avons formulé un algorithme totalement décentralisé et asynchrone nommé AGAVG, dont l'asynchronisme est caractérisé par un modèle de temps Poissonnien où chaque noeud est régi par sa propre horloge sans coordination avec les autres nœuds. Son avantage clé est de ne pas nécessiter de coordination locale entre les émetteurs et les récepteurs. Les messages peuvent ainsi être acheminés avec des délais quelconques sans entraver la convergence globale. Nous avons mis en évidence l'avantage de la décentralisation et de l'asynchronisme lorsque la connectivité du réseau change de manière dynamique (ajout/suppression de noeuds, union/séparation de réseaux, etc). Ceci confère à AGAVG une grande plasticité, dans le sens où le dimensionnement du réseau et de ses unités de calcul peut être accordé à la volée aux contraintes de la tâche à réaliser, tout en garantissant une continuité de la progression vers le résultat.

Dans une certaine mesure, l'approche Gossip asynchrone entre en contradiction avec l'organisation classique du calcul héritée du modèle Turing-Von Neumann, car elle adopte une conception essentiellement non-séquentielle des relations entre opérations. Dans la suite de ce manuscrit, nous nous attachons à montrer que cette approche non-conventionnelle est néanmoins pertinente, surtout lorsque le système est confronté à de grandes masses de données.

A ce stade, nous sommes en mesure d'établir une définition plus précise d'un système décentralisé et asynchrone :

Définition : Système décentralisé et asynchrone

- **C1 : Décentralisation.** Tous les nœuds et tous les liens jouent le même rôle. Tous les nœuds exécutent la même procédure, sans connaissance globale du réseau.
- **C2 : Asynchronisme.** Aucune opération n'implique d'attente entre nœuds.
- **C3 : Plasticité.** Le système tolère les changements de structure du réseau en cours de fonctionnement.
- **C4 : Confidentialité.** Les échantillons ne sont pas échangés entre nœuds. Seuls les paramètres peuvent être échangés.
- **C5 : Coûts de communication.** La taille et le nombre de messages échangés est minimale, dans l'idéal sous-linéaire en le nombre de nœuds.

Cette définition se traduit en un ensemble de contraintes formelles[3] sur les matrices de communications $\mathbf{K}(t)$ introduites dans ce chapitre :

Un système distribué défini par la suite de matrices $(\mathbf{K}(t))_{t\in\mathbb{R}}$ telles que $\mathbf{K}_{ij}(t) \neq 0$ ssi le noeud i envoie un message au noeud j à l'instant t est dit *décentralisé et asynchrone* si

1. Chaque $\mathbf{K}(t)$ est une matrice aléatoire telle que $(\mathbb{E}\mathbf{K}(t))_{ij} \neq 0$ ssi le réseau comporte un lien du noeud i au noeud j à l'instant t.
2. Les matrices $\mathbf{K}(t)$ sont indépendantes dans le temps.
3. Si l'on ajoute un noeud au réseau à l'instant t, l'ensemble des réalisations possibles de $\mathbf{K}(t)$ est exactement celui de $\mathbf{K}(t-dt)$ augmenté des réalisations associées au nouveau noeud.
4. Si l'on supprime un noeud quelconque du réseau à l'instant t, l'ensemble des réalisations possibles de $\mathbf{K}(t)$ est exactement celui de $\mathbf{K}(t-dt)$ diminué des réalisations dont la ligne ou la colonne correspondantes à ce noeud sont non-nulles.
5. A tout instant t, toutes les réalisations de $\mathbf{K}(t)$ ont au plus un coefficient hors diagonale non-nul.

Dans les chapitres suivants, nous étudions l'extension du paradigme d'optimisation décentralisée par protocoles Gossip asynchrones à des problèmes plus complexes que la simple estimation de moyennes, dans le domaine de l'apprentissage statistique. Pour ce faire, les algorithmes proposés dans les chapitres 2 à 4 s'appuient sur le protocole AGAVG défini par l'algorithme 2 et tirent profit des propriétés de séparabilité des opérations couramment utilisées en apprentissage statistique (catégorisation, estimation de densité, réduction de dimension, classification, etc).

[3]Nous adoptons ici le modèle de temps continu ($t \in \mathbb{R}$) défini en section 1.2.1, où les échanges de messages sont des événements ponctuels (Poissonniens) dont la probabilité de cooccurrence est nulle.

CHAPITRE 2

Catégorisation non supervisée et estimation de densité

La catégorisation non-supervisée est une tache d'apprentissage statistique très populaire massivement utilisée entre autres en traitement du signal, en reconnaissance de formes et en fouille de données. Elle consiste à regrouper en catégories un ensemble d'échantillons donnés en entrée. L'objectif est d'assurer une cohérence intra-catégorie maximale tout en garantissant une séparation inter-catégorie optimale. Un modèle de catégorisation est généralement associé à une fonction de coût qui mesure la fidélité avec laquelle il rend compte de la densité de distribution des échantillons dans l'espace d'entrée. Celle-ci peut se contenter d'un critère de cohésion intra-catégorie (*e.g.*, variance intra-catégorie) ou lui adjoindre un critère de séparation inter-catégories (catégorisation à vaste marge).

Dans une première phase dite d'apprentissage ou d'entraînement, les paramètres du modèle sont estimés de sorte à minimiser la fonction de coût sur un ensemble d'échantillons d'entraînement. Une fois les paramètres optimaux obtenus avec suffisamment de précision, le modèle peut être utilisé dans une deuxième phase dite d'inférence pour déterminer la ou les catégories correspondant à un nouvel échantillon jusqu'ici non-présenté.

Dans ce chapitre, nous nous intéressons au problème de *catégorisation distribuée*, qui étend la catégorisation classique centralisée au cas où l'ensemble d'échantillons d'entraînement est reparti sur de multiples nœuds de calcul connectés en réseau. On souhaite alors obtenir un modèle optimal de l'ensemble des échantillons du réseau, pris comme un tout. Notons que dans ce chapitre nous considérons uniquement le cas où ce sont les échantillons qui sont distribués. Nous ne considérons pas le cas où les composantes de ces échantillons sont distribuées, cas qui correspond typiquement au scénario où tous les nœuds observent en simultané des grandeurs diverses émanant d'un même phénomène physique ou un champ de valeurs reparties géographiquement (*e.g.*, météorologique, sismique, etc).

Les principales contributions de ce chapitre sont :

- L'algorithme AGKM (*Asynchronous Gossip K-means*), qui résout le problème K-means distribué de manière décentralisée et asynchrone.
- L'algorithme AGEM (*Asynchronous Gossip Expectation-Maximization*), qui optimise un modèle en mélange de Gaussiennes (*Gaussian Mixture Model*, GMM) de manière décentralisée et asynchrone.

Ces deux algorithmes sont bâtis sur le protocole Gossip asynchrone AGAVG introduit dans le chapitre 1. Ils s'inspirent pour une large part de leurs homologues centralisés, largement utilisés depuis des décennies : K-means [165] et Expectation-Maximization (EM) [70].

Dans la section 1, nous introduisons formellement les problèmes de catégorisation et présentons une sélection des algorithmes centralisés les plus utilisés. Dans la section 2, nous exposons le problème de catégorisation distribuée, ainsi qu'un éventail d'algorithmes dédiés à cette tache. Les section 3 et 4 sont respectivement consacrées à la présentation des deux algorithmes que nous proposons, AGKM et AGEM. Leur analyse théorique est détaillée en section 5. Les résultats de leur étude expérimentale sont rapportés en section 6, avant de conclure ce chapitre en section 7.

2.1 Catégorisation en environnement centralisé

Le problème de catégorisation non-supervisée comporte deux formes typiques :

- La quantification vectorielle, où tout échantillon appartient à une et une seule catégorie. La littérature anglophone utilise généralement le terme de *hard clustering*.
- La catégorisation probabiliste, qui associe à tout échantillon une probabilité d'appartenance à chacune des catégories. La littérature anglophone utilise généralement le terme de *soft clustering*.

Dans ces deux formes, la catégorisation non-supervisée est un problème non-convexe et NP-difficile [169]. La plupart des méthodes de résolution connues s'appuient donc sur une mise à jour itérative et heuristique des paramètres du modèle.

2.1.1 Quantification vectorielle : K-means

Étant donné un ensemble d'échantillons d'entraînement $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^D$, l'objectif de la quantification vectorielle est de trouver les K vecteurs $\mathcal{M} = \{\mu_1, \dots, \mu_K\} \subset \mathbb{R}^D$, appelés prototypes (*codeword* en littérature anglophone), qui minimisent la distance Euclidienne moyenne entre tout échantillon d'entraînement et son prototype le plus proche. Cette fonction coût est nommée Erreur Quadratique Moyenne (*Mean Squared Error*, MSE) et s'exprime comme suit :

$$\text{MSE}(\mathcal{M}) = \frac{1}{n} \sum_{\mathbf{x} \in \mathbf{X}} \min_{\mu \in \mathcal{M}} \|\mathbf{x} - \mu\|_2^2 \tag{2.1}$$

L'ensemble $\mathcal{M}$ des prototypes μ_k est appelé dictionnaire. Associer chaque échantillon au prototype le plus proche induit une catégorisation de $\mathbf{X}$ en K cellules suivant la partition de Voronoi de $\mathbf{X}$ associée à $\mathcal{M}$:

$$\forall k \in \{1, \dots, K\}, \quad \mathcal{C}_k = \{\mathbf{x} \in \mathbf{X} \ : \ \|\mathbf{x} - \mu_k\|_2^2 = \min_l \|\mathbf{x} - \mu_l\|_2^2\} \tag{2.2}$$

Pour effectuer une quantification vectorielle, la méthode la plus usitée est l'algorithme K-means, également connu sous le nom de LBG [163]. K-means adopte une stratégie de descente de gradient pour produire un dictionnaire $\mathcal{M}$ qui minimise localement la MSE.

Pour ce faire, il améliore itérativement le dictionnaire $\mathcal{M}(t)$ en s'appuyant sur des partitionnements successifs $\mathcal{C}(t) = \{\mathcal{C}_1(t), \ldots, \mathcal{C}_K(t)\}$ de $\mathbf{X}$. Pour prendre en compte ces partitions sous-optimales, la MSE peut être reformulée ainsi :

$$\text{MSE}(\mathcal{M}, \mathcal{C}) = \frac{1}{n} \sum_k \sum_{\mathbf{x} \in \mathcal{C}_k} \|\mathbf{x} - \mu_k\|_2^2 \tag{2.3}$$

Dans le cas où $\mathcal{C}$ est la partition de Voronoi de $\mathbf{X}$ associée à $\mathcal{M}$, (2.3) est clairement équivalente à (2.1). En partant d'un dictionnaire initial quelconque $\mathcal{M}(0)$, K-means effectue des itérations successives composées de deux étapes :

Étape 1 : Partitionnement. $\mathcal{M}(t)$ étant fixé, calculer la partition de Voronoi $\mathcal{C}(t+1)$ de $\mathbf{X}$, grâce à (2.2).

Étape 2 : Mise à jour du dictionnaire. La partition $\mathcal{C}(t+1)$ étant fixée, calculer un nouveau dictionnaire $\mathcal{M}(t+1)$ en affectant à chaque prototype $\mu_k(t+1)$ le barycentre de la cellule correspondante $\mathcal{C}_k(t+1)$:

$$\mu_k(t+1) = \bar{\mathcal{C}_k}(t+1) \triangleq \frac{1}{|\mathcal{C}_k(t+1)|} \sum \{\mathbf{x} \in \mathcal{C}_k(t+1)\} \tag{2.4}$$

En alternant ces deux étapes, le dictionnaire $\mathcal{M}(t)$ converge vers un minimum local de la MSE, comme le prouve le théorème suivant :

Theorème 9. *L'algorithme K-means fait décroître la MSE de manière monotone.*

Démonstration. Par définition, le partitionnement de Voronoi garantit que la partition obtenue à l'étape 1 minimise toujours la MSE pour $\mathcal{M}(t)$ fixé :

$$\forall \mathcal{C}', \quad \text{MSE}(\mathcal{M}(t), \mathcal{C}(t+1)) \leq \text{MSE}(\mathcal{M}(t), \mathcal{C}') \tag{2.5}$$

A son tour, l'Étape 2 produit systématiquement le dictionnaire qui minimise la MSE pour $\mathcal{C}(t+1)$ fixé :

$$\forall \mathcal{M}', \quad \text{MSE}(\mathcal{M}(t+1), \mathcal{C}(t+1)) \leq \text{MSE}(\mathcal{M}', \mathcal{C}(t+1)) \tag{2.6}$$

En effet, pour toute cellule $\mathcal{C}_k$ et tout point $\mathbf{z} \in \mathbb{R}^D$, on a

$$\begin{aligned}
\sum_{\mathbf{x} \in \mathcal{C}_k} \|\mathbf{x} - \mathbf{z}\|_2^2 &= \sum_{\mathbf{x} \in \mathcal{C}_k} \left[\|\mathbf{x}\|_2^2 - 2\mathbf{x}^\top \mathbf{z} + \|\mathbf{z}\|_2^2 \right] \\
&= \sum_{\mathbf{x} \in \mathcal{C}_k} \|\mathbf{x}\|_2^2 + |\mathcal{C}_k| \left(-2\bar{\mathcal{C}_k}^\top \mathbf{z} + \|\mathbf{z}\|_2^2 \right) \\
&= \sum_{\mathbf{x} \in \mathcal{C}_k} \left[\|\mathbf{x}\|_2^2 - \|\bar{\mathcal{C}_k}\|_2^2 \right] + |\mathcal{C}_k| \|\bar{\mathcal{C}_k} - \mathbf{z}\|_2^2 \\
&= \sum_{\mathbf{x} \in \mathcal{C}_k} \|\mathbf{x} - \bar{\mathcal{C}_k}\|_2^2 + |\mathcal{C}_k| \|\bar{\mathcal{C}_k} - \mathbf{z}\|_2^2
\end{aligned} \tag{2.7}$$

Comme observé précédemment, le partitionnement de Voronoi entraine que (2.3) appliquée à $(\mathcal{M}(t+1), \mathcal{C}(t+1))$ est équivalente à (2.1) appliquée à $\mathcal{M}(t+1)$. En chaînant (2.5) et (2.6), on obtient finalement

$$\text{MSE}(\mathcal{M}(t+1)) < \text{MSE}(\mathcal{M}(t)) \quad \text{sauf si} \quad \mathcal{C}(t+1) = \mathcal{C}(t) \tag{2.8}$$

□

Cette preuve nous donne d'ailleurs un critère de convergence strict pour K-means : le dictionnaire localement optimal est atteint dès lors que la partition $\mathcal{C}(t)$ n'évolue plus. Néanmoins, on applique souvent en pratique un critère moins fort tel qu'un seuil minimum de décroissance de la MSE entre deux itérations successives.

Remarquons que l'étape de partitionnement investit l'ensemble des échantillons $\mathbf{X}$, tandis que l'étape de mise à jour du dictionnaire ne s'appuie que sur les barycentres des cellules obtenue à l'étape précédente. Une implémentation pratique de K-means ne nécessite donc pas le stockage explicite des cellules. On peut se contenter d'additionner les échantillons au fur et à mesure de leur assignation aux cellules.

Il existe de nombreuses variantes de K-means [23, 134]. Pour les problèmes de grande dimensions, la distance Euclidienne offre généralement de mauvaises performances. Ceci est du au phénomène de concentration des distances qui accompagne l'augmentation de la dimensionnalité (malédiction de la dimensionnalité). En effet, dans les espaces de grande dimension, les distances entre échantillons tendent à se concentrer autour d'une même valeur. Cela conduit à la formation de *hubs* et de *anti-hubs* qui désignent respectivement les points qui deviennent plus proches voisins de quasiment tous les échantillons (hubs) et les points qui ne sont les plus proches voisins d'aucun échantillon (anti-hubs) [205]. L'apparition de ces points particuliers a des conséquences désastreuses sur les performances de catégorisation. Les métriques non-euclidiennes (normes ℓ_p fractionnaires [214], distances géodésiques [107]) offrent des alternatives efficaces pour combattre ce problème.

Au lieu d'appliquer l'étape 1 à l'ensemble des échantillons, on peut se restreindre à un sous-échantillon (aléatoire ou non) et utiliser un sous-échantillon différent à chaque itération. Le cas extrême, nommé quantification vectorielle stochastique (*Stochastic Vector Quantization*, SVQ), consiste à ne prendre qu'un seule échantillon aléatoire à chaque itération. Dans ce cas, la convergence n'est garantie que si l'on ajoute un terme d'inertie à l'étape de mise à jour du dictionnaire, sous la forme d'un pas d'apprentissage décroissant. Autrement, l'algorithme ne converge pas vers un minimum stable mais oscille autour d'un optimum local. SVQ peut être étendu au scenario où les échantillons ne sont pas disponibles en début d'algorithme mais sont capturés un par un pendant la phase d'apprentissage elle même. On parle alors de quantification "en ligne" (*online*).

Enfin, de nombreux travaux ont investigué des approximations du calcul des plus proches voisins dans l'étape de partitionnement, afin de réduire son coût de calcul prohibitif lorsque l'ensemble d'apprentissage est très grand [6, 127].

2.1.2 Catégorisation probabiliste

La limitation principale de K-means et de la quantification vectorielle de manière générale est de ne pas rendre compte du degré d'appartenance d'un échantillon à chacune des catégories, puisque tout vecteur d'entrée est assigné à son prototype le plus proche indépendamment de

sa position réelle. Si, pour palier à cette importante perte d'information, des extensions floues de K-means (*Fuzzy K-means*) ont été proposées, une alternative est d'adopter un modèle de catégorisation probabiliste ou génératif.

A la différence d'un modèle de quantification (*hard clustering*), un modèle de catégorisation probabiliste rend compte d'une probabilité d'appartenance des vecteurs d'entrée à chacune des catégories. Chaque catégorie est modélisée comme une densité de probabilité paramétrique, le modèle global étant formé par une superposition pondérée des distributions de chaque catégorie. Un des modèles les plus populaires est le Modèle en Mélange de Gaussiennes (*Gaussian Mixture Model*, GMM). La distribution des échantillons y est représentée par une superposition de loi normales :

$$p(\mathbf{x}) = \sum_{k}^{K} \omega_k \mathcal{N}_{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k}(\mathbf{x}), \tag{2.9}$$

où $\sum_k \omega_k = 1$ et $\mathcal{N}_{\boldsymbol{\mu},\boldsymbol{\Sigma}}$ représente la loi normale de moyenne $\boldsymbol{\mu} \in \mathbb{R}^D$ et de matrice de covariance $\boldsymbol{\Sigma} \in \mathbb{R}^{D\times D}$ dont la densité de probabilité est donnée par :

$$\forall \mathbf{x} \in \mathbb{R}^D, \quad \mathcal{N}_{\boldsymbol{\mu},\Sigma}(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{2.10}$$

Les catégories peuvent être modélisées par tout autre type de distribution de probabilité (paramétriques ou non) en remplaçant dans l'équation (2.9) les lois normales par les densités correspondantes. Dans tous les cas, l'objectif de la catégorisation probabiliste est généralement d'estimer les paramètres optimaux de ces lois ainsi que leurs pondérations optimales, selon un critère de fidélité de la distribution engendrée par le modèle à la distribution inconnue des échantillons d'entraînement. Un tel critère est par exemple celui du maximum de vraisemblance (*Maximum Likelihood Estimate*, MLE). Si l'on note $\boldsymbol{\theta}$ l'ensemble des paramètres du modèle de catégorisation (pour une GMM, $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_k^K$, la MLE est définie comme suit :

$$\boldsymbol{\theta}^\star = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta} \mid \mathbf{X}) \quad \text{avec} \quad \mathcal{L}(\boldsymbol{\theta} \mid \mathbf{X}) = \prod_{\mathbf{x}\in\mathbf{X}} p_{\boldsymbol{\theta}}(\mathbf{x}) \tag{2.11}$$

$\mathcal{L}(\boldsymbol{\theta} \mid \mathbf{X})$ représente la vraisemblance du jeu de paramètres $\boldsymbol{\theta}$ étant donné un ensemble d'échantillons $\mathbf{X}$, c'est à dire la probabilité qu'un tel jeu de données ait été généré par le modèle correspondant.

Pour optimiser un modèle de catégorisation probabiliste, on estime généralement les paramètres $\boldsymbol{\theta}$ qui maximisent la log-vraisemblance $\ln \mathcal{L}$. Si pour une distribution uni-modale (loi normale, poisson, log-normale, etc) l'estimée du maximum de vraisemblance a une expression analytique directement calculable, lorsque la densité à optimiser est un mélange de telles lois, l'optimum ne peut pas être calculé par une expression directe. Ainsi, comme dans le cas de K-means, on utilise une procédure itérative de type Espérance-Maximisation (*Expectation-Maximization*, EM [70]).

Étant donné un ensemble d'échantillons d'entraînement $\mathbf{X} \subset \mathbb{R}^D$ et partant d'un jeu de paramètres quelconque $\boldsymbol{\theta}(0)$, l'optimisation EM pour les modèles mélanges consiste en deux étapes alternées :

Étape 1 : Espérance. On calcule pour chaque échantillon $\mathbf{x} \in \mathbf{X}$ et chaque catégorie k la probabilité que $\mathbf{x}$ ait été généré par la catégorie k sachant les paramètres courants $\boldsymbol{\theta}(t)$, éga-

lement appelée *responsabilité*. Cette responsabilité est donnée par le théorème de Bayes comme la probabilité que $\mathbf{x}$ soit généré par la catégorie k sachant qu'il a été produit par le modèle complet :

$$\forall \mathbf{x} \in \mathbf{X}, \forall k \in \{1, \ldots, K\}, \quad Q(\mathbf{x}, k) = \frac{\omega_k \mathcal{N}_{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k}(\mathbf{x})}{\sum_l^K \omega_l \mathcal{N}_{\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l}(\mathbf{x})} \tag{2.12}$$

Étape 2 : Maximisation. Les responsabilités étant fixées, on calcule les nouveaux paramètres $\boldsymbol{\theta}(t+1)$ qui maximisent la log-vraisemblance du modèle :

$$\boldsymbol{\theta}(t+1) = \arg\max_{\boldsymbol{\theta}'} \ln \mathcal{L}(\boldsymbol{\theta}' \mid \mathbf{X}, Q). \tag{2.13}$$

Pour une GMM, ces paramètres optimaux ont une forme analytique simple :

$$\forall k, \quad \omega_k(t+1) = \frac{1}{|\mathbf{X}|} \sum_{\mathbf{x} \in \mathbf{X}} Q(\mathbf{x}, k) \tag{2.14}$$

$$\boldsymbol{\mu}_k(t+1) = \frac{1}{\omega_k(t+1)|\mathbf{X}|} \sum_{\mathbf{x} \in \mathbf{X}} Q(\mathbf{x}, k)\mathbf{x} \tag{2.15}$$

$$\boldsymbol{\Sigma}_k(t+1) = \frac{1}{\omega_k(t+1)|\mathbf{X}|} \sum_{\mathbf{x} \in \mathbf{X}} Q(\mathbf{x}, k)\mathbf{x}\mathbf{x}^\top - \boldsymbol{\mu}_k(t+1)\boldsymbol{\mu}_k(t+1)^\top \tag{2.16}$$

Cette procédure est très similaire à celle de K-means, étendue aux modèles probabilistes. On peut notamment interpréter K-means comme un cas particulier d'EM, appliqué au modèle non-probabiliste de la quantification vectorielle. Comme pour K-means, on observe que l'on peut se passer du stockage explicite des responsabilités $Q(\mathbf{x}, k)$, en les calculant une par une et en les sommant au fur et à mesure dans les équations (2.14) à (2.16).

Par ailleurs, on utilise souvent K-means comme initialisation d'EM, du fait de son coût de calcul et de stockage du modèle moins élevé, avant d'effectuer un nombre réduit d'itérations d'EM pour affiner les paramètres du mélange probabiliste.

Les GMM peuvent être estimées avec des matrices de covariance quelconques (on parle de GMM complètes) ou uniquement diagonales (on parle de GMM diagonales), le deuxième cas étant plus rapide en phases entraînement et d'inférence mais offrant une estimation de densité moins précise que le premier.

Le nombre de paramètres total d'une GMM pour k catégories en dimension D est de $k(D^2 + D + 1) + 1)$. En grande dimension, le stockage de ces paramètres peut être problématique et leur mise à jour peut être très coûteuse. Pour palier à ces problèmes, on peut utiliser des modèles dont le rang des matrices de covariance est contraint, telles que les Mélanges d'Analyseurs en Composantes Principales (*Mixtures of Probabilistic Principal Components Analysers*, MPPCA [235]) ou les Mélanges d'Analyseurs de Facteurs (*Mixtures of Factor Analysers*, MoFA [103, 104]). Ces modèles sont d'une implémentation légèrement plus ardue (notamment en calcul distribué) car ils nécessitent l'inversion et/ou la factorisation des matrices de covariance à chaque étape de Maximisation.

Enfin, une limitation des modèles de type GMM et assimilés est que leur procédure d'optimisation EM fait l'hypothèse que les catégories sont exclusives. En effet, dans l'étape E (équation (2.12)), chaque responsabilité Q estime la probabilité d'appartenance d'un échantillon à

une seule des K catégories. Dans le cas où un échantillon peut appartenir à plusieurs catégories (par exemple, les catégories 'chat' et 'animal' ne sont pas exclusives), la solution produite par EM sera généralement peu fidèle à la réalité modélisée (si deux catégories 'chat' et 'chien' sont différenciées, la catégorie 'animal' ne sera pas capturée). Pour palier à cette limitation, des modèles dits à "appartenance mixte" (*Mixed Membership Clustering*, [83]) ont été proposés. Certaines extensions permettent notamment de catégoriser les données d'entrée en considérant un sous-espace distinct pour chaque catégorie [114, 192].

Dans cette thèse, nous considérons uniquement les modèles à appartenance exclusive (*e.g.* GMM, K-means, etc), bien que les algorithmes présentés dans ce chapitre puissent s'étendre sans difficulté particulière aux modèles à appartenance mixte.

2.1.3 Sensibilité aux conditions initiales et minima locaux

En tant que procédures d'optimisation itératives appliquées à des problèmes non-convexes, K-means et EM sont très sensibles aux conditions d'initialisation, en particulier à l'organisation spatiale des prototypes en début d'algorithme. Ainsi, ils convergent souvent vers de "mauvais" minima locaux, c'est à dire des configurations finales où une ou plusieurs catégories sont vides ou peuplées de manière inéquitable. Dans ces situations, les performances de catégorisation sont médiocres, comparées aux configurations pour lesquelles chaque catégorie a une taille (nombre d'éléments), une variance et/ou une pondération équitables.

Pour combattre ces minima locaux non désirés, de nombreux travaux se sont portés sur la construction du modèle initial. Une approche naïve consiste à lancer de multiples instances de l'algorithme avec différents paramètres initiaux, puis de ne retenir que ceux qui minimisent la fonction de coût sur l'ensemble d'entraînement. Le coût de calcul total est alors multiplié par le nombre d'instances exécutées. D'autres méthodes s'appuient sur une estimation préalable d'agrégats statistiques grossier sur les données (covariance, axes principaux, hachage) pour conditionner le modèle à moindre coût et accélérer par la même occasion les premières étapes d'optimisation [13].

Par ailleurs, de nombreux mécanismes additionnels à la descente de gradient ont été proposés afin de rechercher de "meilleurs" minima locaux. Une solution efficace est d'effectuer des "sauts" dans l'espace des paramètres, pour détourner la descente de gradient du bassin d'attraction courant lorsque celui-ci conduit à un minimum local non-pertinent. C'est l'objectif des méthodes de *codeword shifting* [193, 194], qui déplacent brusquement un prototype sous-utilisé à proximité d'un prototype sur-utilisé pour équilibrer les catégories correspondantes. Dans le cas de K-means, le critère d'utilité d'un prototype est généralement une erreur quadratique normalisée par le nombre d'échantillons affecté à la catégorie correspondante. En effectuant cette opération à chaque fois que K-means a convergé vers un minimum local où la variance intra-cellule est inéquilibrée, on peut espérer un équilibrage progressif des utilités des prototypes, conduisant ainsi à la découverte de meilleurs minima locaux en terme de MSE totale.

Pour encourager les prototypes à s'organiser selon une structure contrainte prédéfinie, on peut ajouter des relations topologiques *a priori* entre les prototypes. Cette intuition est notamment l'origine des cartes auto-organisatrices (*Self-Organizing Maps*, SOM) [149] et certains champs de neurones dynamiques (*Dynamic Neural Fields*, DNF [10]). Toutefois, les phénomènes de repliement topologique qui surviennent quasi-systématiquement nuisent grandement à la recherche de meilleurs minima locaux.

De nombreux algorithmes apparentés à K-means comportent également un mécanisme de

recrutement de nouvelles catégories (*e.g.*, *Growing Neural Gas* [173], *Adaptive Resonance Theory* [48]). Ces nouvelles catégories sont souvent recrutées suivant un critère de vigilance ou de tension entre prototypes, déclenché essentiellement selon un seuil d'erreur quadratique maximale ou de distance euclidienne entre prototypes. Si ces extensions régulent par définition l'équilibre des catégories, le problème est en réalité reporté sur la maîtrise de l'explosion du nombre de catégories via le méta-paramètre de vigilance.

2.2 Catégorisation distribuée

La catégorisation distribuée est la transcription de la tache de catégorisation classique au cas où l'ensemble d'entraînement $\mathbf{X}$ n'est pas disponible sur une seule unité de stockage. Ce cas de figure se présente par exemple lorsque l'ensemble d'entraînement est très grand (grand nombre d'exemples n et/ou grande dimension D). Le coût de calcul de l'étape 1 (partitionnement pour K-means et espérance pour EM) devient alors prohibitif car elle implique nK calculs de distances (responsabilités pour EM) en dimension D. Une solution classique est alors de découper $\mathbf{X}$ en sous-ensembles $\{\mathbf{X}_i\}_{i=1}^N$ repartis sur N machines (nœuds) connectées en réseau. Les nœuds de calcul doivent alors échanger des informations sur le réseau pour optimiser un modèle global de l'ensemble $\mathbf{X}$.

On considère donc que chaque nœud i du réseau héberge un ensemble d'échantillons d'apprentissage locaux $\mathbf{X}_i \subset \mathbb{R}^D$ et un jeu de paramètres $\boldsymbol{\theta}_i$. On souhaite alors résoudre le problème de consensus suivant :

$$\begin{aligned} \text{minimiser} \quad & J = \sum_{i=1}^{N} J(\boldsymbol{\theta}_i, \mathbf{X}_i) \\ \text{sous la contrainte} \quad & \forall i, j,\ \boldsymbol{\theta}_i = \boldsymbol{\theta}_j \end{aligned} \tag{2.17}$$

où J est la fonction de coût du modèle de catégorisation choisi. On cherche donc à obtenir en chaque nœud un jeu de paramètres optimaux tels que :

(i) Tous les nœuds convergent vers les mêmes paramètres *consensus*, en partant d'une initialisation quelconque et non-coordonnée entre nœuds.

(ii) Ce modèle consensus doit minimiser la fonction de coût évaluée sur l'ensemble des échantillons du réseau $\mathbf{X}$.

Chaque nœud obtenant un exemplaire du modèle consensus, aucun échange d'information n'est nécessaire en phase d'inférence.

Remarquons que dans la contrainte d'égalité, les indices associés aux catégories doivent être également identiques ($\boldsymbol{\theta}_i$ est un n-uplet ordonné et non un simple ensemble). Ceci garantit qu'en phase d'inférence, tout vecteur présenté en entrée recevra le même numéro de catégorie quel que soit le nœud utilisé.

Dans le cas de la quantification vectorielle distribuée, chaque noeud maintient un dictionnaire local $\mathcal{M}^{(i)} = (\boldsymbol{\mu}_k^{(i)})_{k=1}^K$ et une partition locale $\mathcal{C}^{(i)} = \{\mathcal{C}_k^{(i)}\}_{k=1}^K$ (*i.e.*, $\bigcup_k \mathbf{C}_k^{(i)} = \mathbf{X}^{(i)}$). La fonction de coût à optimiser est similaire à (2.3) à la différence que chaque échantillon y est traité via le dictionnaire local au nœud qui l'héberge. En notant $\boldsymbol{\mathcal{M}} = \{\mathcal{M}^{(i)}\}_{i=1}^N$ et $\boldsymbol{\mathcal{C}} = \{\mathbf{C}^{(i)}\}_{i=1}^N$,

notre objectif s'exprime comme suit :

$$\text{Minimiser} \qquad \text{MQE}(\boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{C}}) = \frac{1}{n} \sum_{i}^{N} \sum_{k}^{K} \sum_{\mathbf{x} \in \mathcal{C}_k^{(i)}} \|\mathbf{x} - \boldsymbol{\mu}_k^{(i)}\|_2^2 \tag{2.18}$$

$$\text{sous la contrainte} \quad \forall i, j \quad \mathcal{M}^{(i)} = \mathcal{M}^{(j)}$$

La catégorisation probabiliste (estimation de densité multimodale) distribuée se défini similairement par la maximisation de la somme des log-vraisemblances des modèles de densité locaux à chaque nœuds, évaluées sur leurs sous-ensemble d'entraînement respectifs $\mathbf{X}_i$. En notant $\boldsymbol{\theta}_i$ le jeu de paramètres maintenu par chaque nœud i et $\boldsymbol{\theta} = \{\boldsymbol{\theta}_i\}_{i=1}^{N}$

$$\text{Maximiser} \qquad \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i}^{N} \sum_{\mathbf{x} \in \mathbf{X}_i} \ln \mathcal{L}(\boldsymbol{\theta}_i \mid \mathbf{x}) \tag{2.19}$$

$$\text{sous la contrainte} \quad \forall i, j \quad \boldsymbol{\theta}_i = \boldsymbol{\theta}_j$$

Dans cette thèse, nous nous intéressons au problème de catégorisation décentralisée et asynchrone, qui spécialise la catégorisation distribuée en ajoutant les contraintes de décentralisation et d'asynchronisme établies dans le chapitre 1 :

C1 : Décentralisation. Tous les nœuds jouent le même rôle.

C2 : Asynchronisme. Pas d'attente entre nœuds.

C3 : Plasticité. Adaptation aux modifications du réseau.

C4 : Coûts de communication. Minimisation des échanges entre nœuds.

C5 : Catégories ordonnées. Un échantillon recoit le même numéro de catégorie en tout noeud.

2.3 État de l'art

Dans cette section, nous présentons un éventail d'algorithmes de catégorisation distribuée, que nous confrontons aux contraintes **C1** à **C5** définies précédemment. Le panel de méthodes que nous décrivons ici est réduit à quelques représentants des grandes familles d'approches existantes. Pour une étude plus exhaustive, le lecteur pourra se référer par exemple à [1, 64]. Les approches distribuées présentées dans cette section partagent deux composantes en commun :

- Une procédure d'optimisation locale des paramètres, qui ne requiert pas de communications.
- Une méthode d'agrégation de modèles locaux qui calcule une combinaison entre deux ou plusieurs modèles échangés sur le réseau.

Ainsi, seuls les paramètres des modèles sont échangés (et pas les échantillons). Néanmoins, elles se distinguent par l'ordonnancement de ces deux composantes et la manière dont les jeux de paramètres sont agrégés. Ces divergences ont des conséquences importantes sur le respect ou non des contraintes fixées, conséquences que nous analysons dans cette section.

Optimisation itérative *versus* agrégation de modèles

Une approche traditionnelle à la catégorisation distribuée est de reprendre les deux étapes alternées de K-means ou d'EM et de les adapter à l'environnement distribué. Cette adaptation est relativement triviale, puisque l'étape 1 est intrinsèquement parallèle et peut donc être réalisée par chaque nœud indépendamment, tandis que l'étape 2 est un simple calcul de moyennes.

Dans [78], ce calcul de moyenne est confiée un nœud maître. Chaque nœud calcule une somme partielle sur son propre jeu de vecteurs, puis la transmet à un nœud particulier, identifié a priori comme "maître". Ce nœud maître, une fois tous les agrégats partiels reçus, peut calculer la moyenne globale et renvoyer le résultat à tous les nœuds du réseau.

Le principal défaut d'une telle procédure itérative est qu'elle nécessite des échanges de messages entre nœuds à chaque itération, entraînant un coût de communication important. Une solution alternative est de diviser le processus en deux temps. Dans un premier temps, tous les nœuds optimisent un modèle local sans échanger d'information. Dans un deuxième temps, ils envoient les modèles obtenus à un nœud maître qui en calcule une agrégation, sans reconsidérer les jeux de données locaux. Cette approche est adoptée par [135], où chaque nœud entraîne localement un dictionnaire optimal sur son propre jeu de données, puis le transmet à un nœud maître qui calcule un dictionnaire agrégé en catégorisant l'ensemble des prototypes produits par tous les nœuds. Le résultat obtenu est une approximation, et n'est donc pas garanti d'être optimal sur l'ensemble $\mathbf{X}$. Néanmoins, l'absence de communications dans l'optimisation itérative implique que chaque nœud n'envoie qu'un seul dictionnaire sur le réseau, et le nœud maître ne doit calculer et envoyer à tous les nœuds qu'un seul dictionnaire global.

Que l'on utilise une procédure itérative telle que [78] ou une approche par agrégation de modèles comme [135], un noeud maître est nécessaire pour réaliser l'étape de combinaison des dictionnaires locaux. De tels algorithmes enfreignent donc les contraintes **C4** à **C6**, puisque le nœud maître doit :

(i) avoir une connaissance globale du réseau et être connecté à tous les autres nœuds, ceci en permanence

(ii) exécuter une procédure spécifique, différente des autres nœuds (distinction *master-workers* habituelle en calcul distribué classique)

(iii) attendre la réception d'un modèle de la part de tous les nœuds avant d'en calculer un modèle agrégé.

Pour contourner le besoin d'un nœud maître, certaines méthodes combinent les dictionnaires le long d'une route prédéfinie à travers le réseau. Par exemple, dans [80] les auteurs proposent un protocole de type Echo-Probe pour calculer la moyenne de l'étape 2 en parcourant un arbre couvrant du réseau. Malgré le coût de communication optimal et équilibré (1 message aller et 1 message retour par nœud), les contraintes **C4** et **C5** ne sont pas satisfaites. En effet, les nœuds doivent attendre la propagation complète du protocole pour obtenir le modèle agrégé et passer à l'étape suivante. De plus, toute erreur de communication sur un lien entraîne l'écroulement de l'algorithme entier puisque toutes les branches descendantes de ce lien sont exclues du calcul de moyenne. Enfin, la nature même de la structure arborescente implique qu'un des nœuds doit être distingué comme racine, ce qui contredit l'hypothèse que tous les nœuds doivent avoir le même rôle.

Approches décentralisées

Une solution pour satisfaire les contraintes **C4** à **C6** est de concevoir un opérateur de combinaison de modèles qui soit décentralisé, c'est à dire qui ne repose que sur des interactions locales entre nœuds voisins. Dans [15, 65], chaque nœud estime la moyenne de l'étape 2 en utilisant uniquement les moyennes partielles calculées par tout ou partie de leurs voisinage direct. Clairement, ceci ne garantit pas que tous les nœuds obtiennent le même modèle agrégé, contrainte imposée par (2.17), surtout lorsque la distribution spatiale des échantillons est hétérogène. Par ailleurs, bien que l'opération d'agrégation soit décentralisée, elle n'en reste pas moins synchrone, puisque chaque nœud doit attendre un message de chacun de ses voisins avant de produire son estimée, invalidant ainsi **C5**.

Dans [99], les auteurs expriment la catégorisation distribuée comme une tache d'optimisation locale sujette à une contrainte de consensus qui force les modèles à être égaux. Pour resoudre ce problème, les noeuds diffusent (*broadcast*) leurs paramètres à tous leurs voisins, convergeant ainsi vers un modèle global en un nombre de passes réduites. Malheureusement, cette méthode spécifiquement conçue pour les canaux de diffusion (*broadcast channels*) suppose des cycles de communication globalement synchrones (tous les nœuds effectuent une étape simultanément), ce qui invalide **C5** et potentiellement **C3** dans un contexte de communication point à point.

Des algorithmes utilisant l'approche par agrégation de modèles sont proposés dans [44, 188]. Dans un premier temps, chaque noeud optimise un modèle local sur son jeu de données via une méthode EM, puis les modèles sont agrégés itérativement entre nœuds voisins aléatoires. L'opération de combinaison consiste à concaténer(superposer) m modèles entrants (proposant chacun K catégories) puis à réduire le modèle obtenu de mK à K catégories par minimisation d'une divergence de Kullback-Leibler (KL) approximée entre le modèle concaténé et le modèle réduit. Cette méthode satisfait **C3** à **C6**, mais viole **C2** car l'opération de réduction détruit clairement l'ordre des catégories.

Newscast-EM [151] est une extension décentralisée de EM, qui utilise le protocole *Newscast* [140] pour calculer la moyenne de l'étape 2. Comme détaillé dans le chapitre 1, Newcast s'appuie sur des moyennages successifs entre paires de noeuds aléatoires. A la différence de [188], **C2** est satisfaite. Un mécanisme additionnel d'"auto-correction" permet aux noeuds d'appliquer un correctif additif d'une itération à l'autre précédente au lieu de simplement repartir de zéro pour estimer la moyenne à itération. Grâce à ce mécanisme, le processus est significativement accélérer. Comme nous le verrons dans la section suivante, ce type de mécanisme ne permet pas seulement une accélération mais constituera un élément essentiel lorsqu'on considérera des communications asynchrones. Toutefois, comme montré dans le chapitre 1, l'opération de moyennage entre paires de nœuds suppose des échanges de messages synchrones, ce qui contredit **C5**. Ce défaut n'est pas mis en évidence du fait de la nature globalement synchrone de l'analyse théorique proposée dans [151] (où chaque nœud est supposé contacter exactement un voisin à chaque passe globale).

En dépit de la décentralisation qui relâche la contrainte de coordination avec un noeud maître, la plupart des algorithmes présentés comme décentralisés déplacent en réalité la contrainte de coordination vers les nœuds voisins [15, 80, 99, 151]. Pourtant, un réseau décentralisé typique est caractérisé par des temps de réponse et des vitesses de calcul très variables selon les noeuds, ce qui signifie que les noeuds rapides peuvent être ralentis par la synchronisation avec des voisins plus lents. Par conséquent, la décentralisation est généralement indissociable de l'asynchronisme lorsque l'on souhaite mettre en pratique ce type algorithmes sur des réseau hétérogènes.

MÉTHODE	C1	C2	C3	C4	C5	C6
DURUT12 [78]	√	√				
JANUZAJ04 [135]	√	√				
EISENHARDT03 [80]	√	√	√			√
BANDYOPADHYAY06 [15]	√		√	√		√
DATTA06 [65]	√		√	√		√
FORERO11 [99]	√	√		√		√
NIKSERESHT08 [188]	√		√	√	√	√
KOWALCZYK04 [151]	√	√	√	√		√
FATTA11 [90]	√	√		√		√

TABLEAU 2.1 – Avantages et inconvénients de quelques méthodes de catégorisation de la littérature, vis-à-vis des contraintes **C1**-**C6**

Epidemic K-means [89, 90] est une autre solution basée sur un protocole Gossip. Contrairement à Newscast, la moyenne de l'étape 2 est calculée de manière totalement asynchrone grâce à un protocole Sum-Weight inspiré de [145]. Chaque nœud y gère ses rôles d'émetteur et de récepteur indépendamment. Cependant, a chaque itération locale à un noeud, son estimée de la moyenne est réinitialisée à zéro. Lorsque le réseau est synchronisé (*i.e.*, tous les nœuds exécutent leur itération simultanément) cela ne pose aucun problème. Mais lorsque les nœuds réalisent leurs itérations de manière asynchrone, la remise à zéro des estimées locales a deux conséquences dramatiques :

(i) Toute contribution éventuellement reçue des nœuds voisins est brutalement annulée. Le réseau converge alors systématiquement vers une moyenne erronée.

(ii) Le réseau ne converge jamais vers un consensus stable, puisque les estimées sont constamment perturbées par la remise à zéro.

On peut donc conclure que dans un contexte totalement asynchrone, Epidemic K-means ne converge pas. Par ailleurs, dans sa conception originale, tous les nœuds doivent démarrer avec le même dictionnaire, et un seul nœud doit initialiser son poids à 1 et les autres à 0, contredisant ainsi **C4**. Enfin, le nombre de messages émis par chaque nœud est fixé arbitrairement, notamment sans prendre en compte la taille et/ou la nature du réseau. Une analyse théorique plus poussée serait nécessaire pour ne pas gaspiller de messages superflus.

Les avantages et défauts des méthodes sus-citées sont résumés dans le Tableau 2.1.

2.4 AGKM : Asynchronous Gossip K-Means

Dans cette section, nous présentons un algorithme original nommé AGKM (Algorithme 3), qui optimise un quantificateur vectoriel de manière totalement décentralisée et asynchrone lorsque les échantillons sont répartis sur de multiples nœuds de calcul au sein d'un réseau.

AGKM reprend les deux étapes alternées de K-means. Il appartient donc à la famille des méthodes par optimisation itérative, au même titre que [15, 65, 78, 90, 151]. Mais contrairement à ces approches, AGKM s'appuie sur un protocole Gossip perturbé pour garantir une convergence systématique y compris dans des environnements réseau totalement asynchrones. Ce protocole

Gossip perturbé est une combinaison d'AGAVG et d'un mécanisme d'auto-correction apparenté à [78, 151]. Il respecte ainsi l'ensemble des contraintes fixées **C1** $-$ **C6**.

AGKM comporte deux procédures exécutées en parallèle en chaque nœud du réseau : une procédure d'émission qui réalise une optimisation locale et envoie des messages aux nœuds voisins, et une procédure de réception chargée de fusionner les informations reçues avec les estimées locales. Sa convergence est confirmée par une analyse théorique en section 6, puis par une étude expérimentale en section 7.

Le principal apport d'AGKM est de montrer qu'un protocole de type Sum-Weight associé à un mécanisme d'auto-correction sont les deux éléments indispensables à un fonctionnement asynchrone et décentralisé. Dans [92], nous avons traité la question de l'initialisation des dictionnaires locaux et du nombre de messages que chaque nœud doit émettre du double point de vue théorique et expérimental, en fournissant une borne supérieure sur le nombre de messages à émettre par nœud pour garantir la convergence globale d'AGKM en fonction de la taille du réseau. Dans cet article, la stabilité du consensus et la condition de terminaison en environnement asynchrone restent des questions ouvertes. Dans cette thèse, nous montrons qu'un mécanisme d'auto-correction peut solutionner ces problèmes. Enfin dans [91, 94], nous avons évalué AGKM sur des tâches d'indexation d'images par dictionnaire visuel, étude expérimentale que nous développons dans le chapitre 5 du présent manuscrit.

Algorithme 3 AGKM : Asynchronous Gossip K-means (en chaque noeud i)

Entrées : Jeu d'échantillons $\mathbf{X} \in \mathbb{R}^{n_i \times D}$
Paramètres : K : nombre de cellules désirées
M : nombre de messages envoyés à chaque itération locale

∘ **Procédure émission**

$\boldsymbol{\mathcal{C}} \leftarrow$ partition aléatoire de $\mathbf{X}$ en K cellules
$\forall k \in \{1, \ldots, K\}, \quad w_k \leftarrow w_k^{\text{old}} \leftarrow |\mathcal{C}_k|; \quad \mathbf{s}_k \leftarrow \mathbf{s}_k^{\text{old}} \leftarrow \sum\{\mathbf{x} \in \mathcal{C}_k\}$
Boucle
  Pour m **de** 1 **à** M **faire**
    $\forall k,\ w_k \leftarrow w_k/2; \quad \mathbf{s}_k \leftarrow \mathbf{s}_k/2$
    $j \leftarrow$ nœud voisin aléatoire
    Envoyer $(\mathbf{s}_k, w_k)_{k=1}^K$ à j
  Fin Pour
  $\forall k,\ \mathcal{C}_k \leftarrow \{\mathbf{x} \in \mathbf{X}^{(i)} : k = \arg\min_l \|\mathbf{x} - \mathbf{s}_l/w_l\|_2^2\}$
    $w_k \leftarrow w_k - w_k^{\text{old}} + |\mathcal{C}_k|; \quad \mathbf{s}_k \leftarrow \mathbf{s}_k - \mathbf{s}_k^{\text{old}} + \sum\{\mathbf{x} \in \mathcal{C}_k\}$
    $w_k^{\text{old}} \leftarrow |\mathcal{C}_k|; \quad \mathbf{s}_k^{\text{old}} \leftarrow \sum\{\mathbf{x} \in \mathcal{C}_k\}$
Fin Boucle

∘ **Procédure réception**

Boucle
  Attendre réception d'un message $(\mathbf{s}'_k, w'_k)_{k=1}^K$
  $\forall k,\ w_k = w_k + w'_k; \quad \mathbf{s}_k = \mathbf{s}_k + \mathbf{s}'_k$
Fin Boucle

Dictionnaire consensus

AGKM s'appuie sur la notion de *dictionnaire consensus*. Chaque nœud i maintient un dictionnaire local $\mathcal{M}^{(i)} = \{\boldsymbol{\mu}_k^{(i)}\}_{k=1}^K$ et une partition locale $\mathcal{C}^{(i)} = \{\mathcal{C}_k^{(i)}\}_{k=1}^K$ de son ensemble d'apprentissage $\mathbf{X}^{(i)}$, initialisés aléatoirement, sans coordination entre nœuds. Le dictionnaire consensus $\mathcal{M}^\star = \{\boldsymbol{\mu}_k^\star\}_{k=1}^K$ est alors défini par :

$$\forall k, \quad \boldsymbol{\mu}_k^\star = \frac{\sum_{i=1}^N \sum\{\mathbf{x} \in \mathcal{C}_k^{(i)}\}}{\sum_{i=1}^N |\mathcal{C}_k^{(i)}|} = \frac{1}{|\mathcal{C}_k^\star|} \sum\{\mathbf{x} \in \mathcal{C}_k^\star\}, \tag{2.20}$$

où $\mathcal{C}_k^\star \triangleq \bigcup_{i=1}^N \mathcal{C}_k^{(i)}$ est l'union de toutes les cellules de même indice k à travers le réseau. $\mathcal{C}^\star = \{\mathcal{C}_k^\star\}_{k=1}^K$ est ainsi une partition de l'ensemble des échantillons du réseau. De plus, rappelons que d'après (2.6), $\mathcal{M}^\star$ est par définition le dictionnaire optimal associé à $\mathcal{C}^\star$. Naturellement, $\mathcal{M}^\star$ et $\mathcal{C}^\star$ ne sont pas localement observables par les nœuds. Néanmoins, pour $\mathcal{C}^\star$ donnée, chaque prototype consensus $\boldsymbol{\mu}_k^\star$ n'est qu'une simple moyenne des $\sum\{\mathbf{x} \in \mathcal{C}_k^{(i)}\}$ pondérés par les $|\mathcal{C}_k^{(i)}|$, qui ne sont autres que la somme et le nombres d'échantillons assignés à la cellule locale $\mathcal{C}_k^{(i)}$. Ces quantités sont évidemment localement calculable par le nœud i.

Notons que toute modification apportée par un nœud à sa partition locale est directement reflétée dans le dictionnaire consensus, de la même façon que le dictionnaire est mis à jour après une étape de partitionnement dans l'algorithme K-means classique. AGKM s'appuie alors sur les deux intuitions suivantes :

(a) Si la règle de partitionnement local est formellement équivalente à celle de K-means, alors l'évolution du dictionnaire consensus suit exactement celle du dictionnaire optimisé par K-means.

(b) Si à tout instant les dictionnaires locaux tendent vers ce dictionnaire consensus, alors tous les nœuds termineront avec un dictionnaire égal et équivalent à celui produit par K-means.

Protocole Gossip asynchrone perturbé

Le processus formel réalisé par AGKM est un protocole Gossip perturbé de la forme (1.88) (*cf* chapitre 1), dont nous rappelons l'équation générale d'évolution

$$\forall t, \forall i, \quad x_i(t+1) = \sum_{j=1}^N \mathbf{K}_{ji}(t+1) x_j(t) + f_{t+1}(x_i(t), \ldots, x_i(0)) \tag{2.21}$$

Le premier terme de (2.21) est la transcription de l'intuition (b), puisqu'il s'agit d'une moyenne pondérée entre nœuds. Le deuxième terme f_{t+1} est défini de sorte à réaliser l'intuition (a), via l'addition au dictionnaire consensus d'auto-corrections calculées localement par les nœuds à intervalles quelconques et désynchronisés (*i.e.*, à tout moment f_{t+1} est nulle ou non en un nœud i indépendamment des décisions de ses voisins).

Ainsi, en dépit de son schéma d'optimisation itératif, AGKM se distingue formellement de [90, 151] par son absence de cycles globalement synchrones. En effet, si [90, 151] recourent à un modèle de temps synchrone où tous les nœuds exécutent une opération simultanément à intervalle régulier, AGKM ne suppose pas une telle horloge globale. Au contraire, chaque nœud

évolue selon sa propre horloge indépendamment de ses voisins. Il en résulte que les étapes locales de partitionnement et de mise à jour du dictionnaire ne sont plus globalement distinctes comme c'est le cas dans la plupart des implémentations distribuées de K-means. A tout moment, certains nœuds peuvent être en phase de partitionnement (étape 1 de K-means) quand d'autres sont en phase de mise à jour (étape 2 de K-means). Dans ce cas de figure, il apparaît que les algorithmes précédemment cités échouent à converger vers un consensus stable.

Pour garantir un fonctionnement parfaitement asynchrone, les seules contraintes fixées sur la matrice de communication $\mathbf{K}(t)$ sont :

- $\mathbf{K}(t)$ est aléatoire et statistiquement indépendante de $\mathbf{K}(t-1)$, afin d'assurer l'indépendance temporelle des échanges de messages nécessaire à un fonctionnement asynchrone.

- $\mathbb{E}\mathbf{K}(t)$ est irréductible pour tout instant t. Cette hypothèse naturelle stipule *(i)* que le graphe orienté associé au réseau doit être fortement connecté (dans le cas de canaux bi-directionnels, cela équivaut à supposer que le graphe est connexe) et *(ii)* que tout lien peut être emprunté à tout moment avec une probabilité non nulle (autrement la contrainte d'asynchronisme serait violée).

- Toute réalisation de $\mathbf{K}(t)$ a une diagonale strictement positive. Cela signifie que les nœuds conservent systématiquement une portion strictement positive de leur propre estimée.

- Toute réalisation de $\mathbf{K}(t)$ est stochastique (*i.e*, ses lignes somment à 1). Autrement dit, $\mathbf{K}(t)\mathbf{1} = \mathbf{1}$. Cette dernière hypothèse peut être relâchée en imposant seulement à $\mathbb{E}\mathbf{K}(t)$ d'être stochastique. Néanmoins, pour simplifier l'analyse théorique d'AGKM on supposera que toute réalisation l'est, hypothèse finalement peu contraignante dans le contexte considéré.

Mécanisme d'auto-correction

Dans [151] et [78], deux mécanismes d'auto-correction très similaires sont proposés, avec l'accélération des calculs pour principale justification. Leur principe commun est le suivant : lorsqu'à l'issue de l'étape 1 (partitionnement) un nœud propose une nouvelle moyenne partielle pour l'étape 2 (mise à jour du dictionnaire), il n'écrase pas directement son estimée avec sa nouvelle moyenne partielle (comme dans [90]). Au contraire, il met à jour cette estimée en se basant sur sa valeur à l'itération précédente, sans la remettre au préalable à zéro.

Dans un contexte synchrone, où tous les nœuds réalisent simultanément soit l'étape de partitionnement soit l'étape de mise à jour, cette astuce n'apporte qu'un gain en vitesse de convergence. Cependant, dans un contexte asynchrone, ce mécanisme se révèle crucial. En effet, en son absence, tout nœud peut remplacer à tout moment son estimée courante par une moyenne partielle fraîchement calculée, ce qui entraîne deux conséquences désastreuses :

1. Toutes les contributions accumulées depuis les nœuds voisins sont annulées et définitivement perdues. Dans un contexte synchrone les conséquences sont mineures, puisque tous les nœuds recommencent en même temps leur calcul de moyenne avec de nouvelles estimées. Mais dans un environnement asynchrone, l'annulation des contributions des voisins en pleine estimation de moyenne modifie irrémédiablement cette moyenne vers laquelle les nœuds sont sensés converger. Toute confiance vis-à-vis du dictionnaire commun est

par conséquent perdue lorsque les événements de partitionnement surviennent de manière désynchronisée.

2. Dans la mesure où les nœuds écrasent régulièrement leur estimée du dictionnaire, celle-ci ne se stabilise jamais. Une fois de plus, ce problème n'apparaît que lorsque ces remises à zéro survient de manière désynchronisée. Dans un tel contexte, les dictionnaires locaux oscillent en permanence entre une certaine valeur locale de départ et le dictionnaire consensus ciblé. L'algorithme ne converge donc jamais totalement sauf exception. De plus, il n'y a aucune garantie que stopper l'algorithme à l'instant $t_2 > t_1$ produira un modèle plus performant qu'en le stoppant à l'instant t_1, ce qui empêche notamment l'utilisation de toute condition de terminaison basée sur un seuil d'amélioration de la MSE.

Le mécanisme d'auto-correction d'AGKM est conçu de sorte à conserver en permanence les contributions des nœuds voisins à l'estimée locale du dictionnaire consensus. De ce fait, il ne fonctionne que par des corrections additives aux estimées locales. Lorsqu'un nœud i réalise une étape de partitionnement (ligne 9), il ajoute la somme des échantillons locaux assignés à chaque cellule $\mathcal{C}_k^{(i)}$ à l'estimée correspondante $\mathbf{s}_k^{(i)}$ et retranche celle qui avait été ajoutée à l'étape de partitionnement locale précédente $\mathbf{s}_k^{\text{old}(i)}$. Cette opération est également effectuée pour le nombre d'éléments $|\mathcal{C}_k^{(i)}|$ de chaque cellule :

$$\mathbf{s}_k^{(i)}(t+1) = \mathbf{s}_k^{(i)}(t) + \sum\{\mathbf{x} \in \mathcal{C}_k^{(i)}(t)\} - \mathbf{s}_k^{\text{old}(i)} \tag{2.22}$$

$$w_k^{(i)}(t+1) = w_k^{(i)}(t) + |\mathcal{C}_k^{(i)}(t)| - w_k^{\text{old}(i)} \tag{2.23}$$

Suite à cette mise à jour, le dictionnaire consensus $\mathcal{M}^\star(t)$ est ainsi implicitement modifié. En supposant que i possédait une estimée parfaite de $\mathcal{M}^\star(t)$ au moment du partitionnement, cette mise à jour est formellement équivalente à celle qu'aurait effectué un K-means classique en partitionnant le sous ensemble $\mathbf{X}^{(i)}$. Comme nous le verrons dans la section suivante il n'est pas nécessaire de disposer d'une estimée parfaite de $\mathcal{M}^\star(t)$, cette équivalence tolérant une faible erreur d'estimation.

On sait, d'après la preuve de convergence de K-means (Théorème 9) qu'à mesure que le dictionnaire consensus approche d'un minimum de la MSE (évaluée sur $\mathbf{X}$) sa mise à jour tend vers zéro. Il s'ensuit que, contrairement à [90], les estimées $(\mathbf{s}_k^{(i)}(t), w_k^{(i)}(t))$ se stabilise progressivement. AGKM converge alors vers notre objectif, à savoir un dictionnaire égal en tout nœud, qui minimise localement la MSE sur $\mathbf{X}$. Ce dictionnaire est simplement obtenu en calculant

$$\forall i, \forall k,\ \boldsymbol{\mu}_k^{(i)}(t) = \frac{\mathbf{s}_k^{(i)}(t)}{w_k^{(i)}(t)} \tag{2.24}$$

Paramètre critique M

Parce que dans un contexte asynchrone les événements de partitionnement peuvent survenir à tout instant en un nœud quelconque, le dictionnaire consensus est sujet à des variations imprévisibles durant son estimation. Plus les événements de partitionnement (*i.e.*, instants t où f_t est non-nulle) sont fréquents par rapport aux événements de moyennage (*i.e.*, instants t où f_t est nulle), plus ces variations perturbent le bon déroulement du processus de moyennage vers le dic-

tionnaire consensus. Lorsque ce rapport est élevé, les estimées locale deviennent très imprécises, et peuvent nuire à la convergence du processus global.

En effet, lorsqu'un nœud effectue une étape de partitionnement malgré une estimée du dictionnaire consensus imprécise, un échantillon peut être assigné à une cellule différente de celle à laquelle il aurait été assigné en utilisant le dictionnaire consensus. Nous nommons ce phénomène "désaccord d'assignation" (*Membership Mismatch*, MM). Intuitivement, plus le rapport entre la fréquence de partitionnement et celle de moyennage est élevé, plus le taux de désaccords d'assignation (*Percentage of Membership Mismatch*, PMM) sera grand.

Dans AGKM, cette fréquence relative est contrôlée par le paramètre M, qui fixe le nombre de messages envoyés par chaque nœud entre chaque étape de partitionnement locale. Si l'on attribue à tous les nœuds le même M, les événements de moyennage à travers le réseau sont clairement M fois plus fréquents que les événements de partitionnement. Remarquons que si augmenter M favorise une meilleure précision d'estimation du dictionnaire consensus, cela accroît également le coût de communication. Dans la mesure où seuls les événements de partitionnement font progresser le dictionnaire consensus, le processus global d'optimisation en est d'autant ralenti. A l'inverse, un M trop faible peut conduire à l'échec de la convergence par manque de précision sur les estimées du consensus. Le réglage de M relève par conséquent d'un compromis stabilité et convergence *versus* temps de calcul et coût de communication.

Déterminer une valeur optimale de M n'est pas une question triviale. En effet, celle-ci dépend en grande partie de la distribution de densité des données, qui est par définition inobservable. Néanmoins, l'analyse théorique présentée dans la section suivante montre que la convergence de l'algorithme peut être garantie malgré une légère erreur d'estimation, ce qui permet de déterminer un nombre de message suffisant à échanger entre deux étapes de partitionnement. L'étude expérimentale en section 6 révèle qu'en pratique le processus converge malgré une erreur d'estimation bien plus élevée, autorisant ainsi des valeurs de M relativement faibles.

Sensibilité aux conditions initiales et codeword-shifting

Comme K-means, AGKM est sensible à la configuration initiale des dictionnaires locaux et converge donc vers un minimum local différent selon cet état initial. Toutefois, la nature distribuée et stochastique d'AGKM lui confère un avantage significatif pour combattre les minima locaux les plus "simples". Ces minima locaux sont ceux pour lesquels une ou plusieurs cellules sont vides ou très peu peuplées. Ils résultent généralement de l'initialisation de certains prototypes dans des régions où se trouvent seulement quelques échantillons aberrants. Lorsque l'ont choisit aléatoirement K échantillons comme prototypes initiaux de K-means, la probabilité de sélectionner un échantillon isolé ou aberrant est non-négligeable. Dans AGKM, chaque nœud initialisant ses prototypes indépendamment, il est peu probable que tous les nœuds choisissent d'initialiser leur prototype d'indice k sur un point aberrant. Aussi, parmi les N prototypes d'indice k qui contribuent à la formation de la cellule globale $\mathcal{C}_k^\star$, il est probable qu'un de ces prototypes ait été initialisé dans une région dense en échantillons. Les prototypes du dictionnaire consensus étant une moyenne des prototypes locaux pondérés par la taille des cellules associées, la probabilité que les prototypes du dictionnaire consensus soient attirés dans des régions denses est plus élevée que dans le cas de K-means avec un unique dictionnaire initialisé aléatoirement. AGKM profite donc de la nature stochastique des multiples dictionnaires initiaux pour atteindre de meilleurs minima locaux dans le cas moyen.

Malgré cet avantage, AGKM tombe tout de même dans des minima locaux que la simple ini-

tialisation stochastique ne peut pas combattre a priori. Pour traiter ces minima, nous avons proposé dans [92, 94] d'intégrer une procédure locale de *codeword-shifting*, inspirée de l'approche centralisée de [193]. Elle permet la recherche de meilleurs dictionnaires lorsque l'algorithme a convergé vers une configuration stable mais sous-optimale. Pour ce faire, elle s'appuie sur une estimation locale de la distorsion de chaque cellule globale $\mathcal{C}_k^\star(t) = \bigcup_{i=1}^N \mathcal{C}_k^{(i)}(t)$. En tant que moyenne des distorsions des cellules locales, celle-ci peut se calculer selon le même schéma que le dictionnaire lui-même (*i.e.*, par moyennage d'estimées localement calculables) :

$$\forall t, \quad d_k(t) = \frac{\sum_{i=1}^N \sum_{\mathbf{x}\in\mathcal{C}_k^{(i)}} \|\mathbf{x} - \boldsymbol{\mu}_k^{(i)}\|_2^2}{\sum_{i=1}^N |\mathcal{C}_k^{(i)}|} \tag{2.25}$$

De manière similaire, les noeuds peuvent estimer la MSE globale comme une moyenne des MSE locales :

$$\forall t, \quad \text{MSE}(t) = \frac{\sum_{i=1}^N \sum_{\mathbf{x}\in\mathbf{X}^{(i)}} \min_k \|\mathbf{x} - \boldsymbol{\mu}_k^{(i)}\|_2^2}{\sum_{i=1}^N |\mathbf{X}^{(i)}|} \tag{2.26}$$

La procédure de *codeword-shifting* consiste à déplacer le prototype k_{min} associé à la cellule de plus faible distorsion à proximité du prototype k_{max} associé à la cellule de plus forte distorsion :

$$\boldsymbol{\mu}_{k_{\text{min}}}(t+1) = \boldsymbol{\mu}_{k_{\text{max}}}(t) + \eta \quad \text{et} \quad \boldsymbol{\mu}_{k_{\text{max}}}(t+1) = \boldsymbol{\mu}_{k_{\text{max}}}(t) - \eta, \tag{2.27}$$

où η est un petit vecteur arbitraire fixé. Bien que moins évoluée que [193], cette fonctionnalité relance l'optimisation dans une configuration propice à l'égalisation des distorsions des cellules. Cette procédure de *codeword-shifting* est déclenché spontanément par un nœud i lorsque les deux conditions suivantes sont satisfaites :

1. Son estimée $\text{MSE}^{(i)}$ de l'erreur quadratique moyenne globale a convergé vers un minimum local (*i.e.* sa variation est passée sous un seuil fixé).

2. L'erreur quadratique du modèle est mal repartie entre les cellules. Cette condition est vérifiée lorsque l'écart-type relatif de la distorsion des cellules $\text{RSD}^{(i)}$ est supérieur à un seuil fixé.

$$\text{RSD}^{(i)} = \frac{1}{\text{MSE}^{(i)}} \sqrt{\sum_{k=1}^K \left(d_k^{(i)} - \text{MSE}^{(i)}\right)^2} \tag{2.28}$$

Avec ce mécanisme additionnel, AGKM termine en un nœud i lorsque son estimée $\text{MSE}^{(i)}$ a convergé et que $\text{RSD}^{(i)}$ est passée sous le seuil σ_{shift}. Il retourne alors le dictionnaire défini par (2.24) dont la MSE est ainsi localement optimale avec une erreur quadratique équitablement repartie entre les cellules.

2.4.1 Analyse Théorique

Dans cette section, nous analysons la convergence d'AGKM d'un point de vue théorique. Nous montrons en particulier qu'il converge vers des modèles locaux qui minimisent l'erreur quadratique sur l'ensemble des échantillons du réseau tout en garantissant que tous les nœuds possèdent des modèles rigoureusement égaux.

On reconnaît dans (2.18) un problème de minimisation sous contraintes d'égalité. Il nous faut donc garantir à la fois la convergence de la MQE vers un minimum local et l'égalité des modèles locaux à convergence. Pour ce faire, nous définissons deux fonctions de coût $J_1(\boldsymbol{\mathcal{M}})$ et $J_2(\boldsymbol{\mathcal{M}})$ qui caractérisent ces deux composantes :

$$J_1(\boldsymbol{\mathcal{M}}) = \frac{1}{n}\sum_{i=1}^{N}\sum_{\mathbf{x}\in\mathbf{X}^{(i)}} \min_k \|\mathbf{x} - \boldsymbol{\mu}_k^{(i)}\|_2^2 \tag{2.29}$$

$$J_2(\boldsymbol{\mathcal{M}}) = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{K} \|\boldsymbol{\mu}_k^{(i)} - \boldsymbol{\mu}_k^{(j)}\|_2^2 \tag{2.30}$$

Rappelons que le dictionnaire consensus $\mathcal{M}^\star$ défini par (2.20) ne dépend que des partitions $\boldsymbol{\mathcal{C}}$. On peut donc réécrire J_1 et J_2 de sorte à obtenir un problème équivalent où $\boldsymbol{\mathcal{M}}$ n'intervient plus dans J_1 :

$$J_1(\boldsymbol{\mathcal{C}}) = \frac{1}{n}\sum_{\mathbf{x}\in\mathbf{X}} \min_k \|\mathbf{x} - \boldsymbol{\mu}_k^\star\|_2^2 \tag{2.31}$$

$$J_2(\boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{C}}) = \sum_{i=1}^{N}\sum_{k=1}^{K} \|\boldsymbol{\mu}_k^{(i)} - \boldsymbol{\mu}_k^\star\|_2^2 \tag{2.32}$$

J_1 mesure ainsi la MSE du dictionnaire consensus évaluée sur l'ensemble des données du réseau, et J_2 mesure la déviation des dictionnaires locaux par rapport au dictionnaire consensus. Clairement, lorsque $\forall i, j, \mathcal{M}^{(i)} = \mathcal{M}^{(j)}$, ce problème reformulé est équivalent à (2.18) et on observe que (2.31) est équivalente à (2.1) appliquée à $\mathcal{M}^\star$.

Pour prouver la convergence de AGKM, nous montrons *(i)* que le dictionnaire consensus se comporte exactement comme si il était optimisé avec un algorithme de K-means centralisé, et *(ii)* que la contrainte d'égalité des dictionnaires locaux est asymptotiquement vérifiée.

Pour ce faire, nous analysons l'effet sur J_1 et J_2 des deux types d'événements qui peuvent survenir de manière asynchrone en un nœud quelconque au cours du temps :

- **Mise à jour d'un dictionnaire local.** Un nœud i reçoit un message et met ainsi à jour son dictionnaire $\mathcal{M}^{(i)}$ par modification de ses estimées locales $(\mathbf{s}_k^{(i)}, w_k^{(i)})_k^K$. Les partitions $\boldsymbol{\mathcal{C}}$ n'étant pas modifiées, $\mathcal{M}^\star$ est invariant à ce type d'événement. Par conséquent, un événement de mise à jour n'influence que J_2.

- **Partitionnement local.** Un nœud i calcule une nouvelle partition $\mathcal{C}^{(i)}$. Ce type d'événement influence à la fois J_1 et J_2, puisque le dictionnaire consensus est modifié.

Remarquons que seuls ces deux types d'événements (qui sont par définition discrets) influencent J_1 et J_2. On peut donc adopter un modèle de temps discret $t \in \mathbb{N}$ qui compte simplement le nombre de ces événements écoulés depuis le lancement de l'algorithme pour rendre compte fidèle du comportement d'AGKM en temps réel (indépendamment de la nature discrète ou continue du cadre temporel d'exécution considéré). Pour une meilleure lisibilité, nous définissons par ailleurs deux valeurs entières p et m qui comptent respectivement le nombre d'événements de partitionnement et de mise à jour survenus sur le réseau depuis la lancement de l'algorithme.

Pour montrer la décroissance de J_1 et J_2, notre schéma de preuve est le suivant : Premièrement, nous montrons que J_2 décroit de manière monotone entre deux événements de partitionnement. Deuxièmement, nous prouvons qu'à chaque événement de partitionnement J_1 décroit, sous réserve qu'on nombre minimal d'événements de mise à jour aient lieu entre deux événements de partitionnement. Troisièmement, nous montrons que lorsque J_1 atteint un minimum local les événements de partitionnement n'ont plus d'impact et peuvent être ignorés. Finalement, puisque dès lors seuls les événements de mise à jour subsistent, J_2 converge vers 0.

Domination de J_2

Pour montrer la décroissance de J_2, nous prouvons tout d'abord que les prototypes consensus sont à tout instant la moyenne des estimées $\mathbf{s}_k^{(i)}$ à travers le réseau pondérées par les $w_k^{(i)}$ associés. Cette propriété est établie par le lemme suivant :

Lemme 1. *A tout instant t et pour toute catégorie k, l'égalité suivante est vérifiée :*

$$\frac{\sum_i^N \mathbf{s}_k^{(i)}(t)}{\sum_i^N w_k^{(i)}(t)} = \boldsymbol{\mu}_k^\star(t) \tag{2.33}$$

Ainsi, $\mathcal{M}^\star$ est à tout instant le dictionnaire moyen associé au sommes $\mathbf{s}_k^{(i)}$ et aux poids $w_k^{(i)}$. D'après les lignes 5-7 de la procédure d'émission et 2-3 de la procédure de réception d'AGKM, ces estimées évoluent suivant le protocole Gossip asynchrone présenté dans le chapitre 1. Les prototypes locaux étant définis par $\boldsymbol{\mu}_k^{(i)}(t) = \mathbf{s}_k^{(i)}(t)/w_k^{(i)}(t)$, le Théorème 7 nous garantit la convergence monotone et exponentielle de ces prototypes vers les prototypes consensus correspondants, et nous donne une borne sur le temps ε-convergence. Entre deux événements de partitionnement, la décroissance de J_2 est alors établie par le théorème suivant :

Theorème 10. *A tout instant t et pour tout $\epsilon > 0$ et toute probabilité $\delta \in]0,1[$, on a $J_2(t) \leq DK\varepsilon$ avec une probabilité au moins δ, si les T événements précédents sont tous des mises à jour de dictionnaire, où*

$$T = 2N\left(4\ln N + \frac{6N(\ln N + 1)}{1-\delta} - \ln\varepsilon + \ln\frac{2}{1-\delta}\right) \tag{2.34}$$

Condition suffisante pour la décroissance de J_1

Lors d'un événement de partitionnement à l'instant $t+1$ en un nœud i, la partition globale $\mathcal{C}^\star(t)$ est modifiée en réaffectant les échantillons de $\mathbf{X}^{(i)}$ à leur plus proche voisin dans $\mathcal{M}^{(i)}(t)$. Par définition de la partition de Voronoi, $\mathcal{C}^{(i)}(t+1)$ minimise donc la MSE sur $\mathbf{X}^{(i)}$ sachant $\mathcal{M}^{(i)}(t)$. $\boldsymbol{\mathcal{C}}(t+1)$ est donc logiquement l'ensemble optimal de partitions locales sachant les dictionnaires locaux $\boldsymbol{\mathcal{M}}(t)$. Cependant, il n'est pas garanti que $\boldsymbol{\mathcal{C}}(t+1)$ minimise J_1 sachant $\mathcal{M}^\star(t)$. Il est même possible que J_1 augmente significativement lors d'un événement de partitionnement et fasse diverger AGKM. Cette propriété distingue d'ailleurs formellement AGKM des méthodes synchrones pour lesquelles J_1 est toujours minimisée par le calcul d'une nouvelle partition.

Toutefois, rappelons que $\boldsymbol{\mathcal{C}}(t+1)$ minimiserait systématiquement J_1 sachant $\mathcal{M}^\star(t)$ si l'on utilisait $\mathcal{M}^\star(t)$ pour effectuer le partitionnement local. $\mathcal{M}^\star(t)$ étant localement inconnu, cette solution n'est pas praticable mais nous sert de référence, puisque dans ce cas AGKM

se comporterait exactement comme un K-means centralisé. Ainsi, l'éventuelle augmentation de J_1 est toujours liée à une estimation trop imprécise de $\mathcal{M}^\star(t)$ en i. Si i parvient à obtenir $\mathcal{M}^{(i)}(t) = \mathcal{M}^\star(t)$, il est donc assuré de minimiser J_1 sachant $\mathcal{M}^\star(t)$.

Remarquons maintenant que pour minimiser J_1 sachant $\mathcal{M}^\star(t)$, il n'est pas nécessaire de respecter $\mathcal{M}^{(i)}(t) = \mathcal{M}^\star(t)$. Il suffit de respecter

$$\forall k, \quad \mathcal{C}_k^{(i)}(t) = \{\mathbf{x} \in \mathbf{X}^{(i)} : \forall l, \|\mathbf{x} - \boldsymbol{\mu}_k^\star(t)\|_2^2 \leq \|\mathbf{x} - \boldsymbol{\mu}_l^\star(t)\|_2^2\}. \tag{2.35}$$

Autrement dit, il suffit d'obtenir la même partition des données locales que celle que l'on aurait obtenue en utilisant le dictionnaire consensus. Cette condition moins stricte est explicitée par le lemme suivant (prouvé en annexe A) :

Lemme 2. *A tout instant t et pour tout nœud i, la partition de Voronoi de $\mathbf{X}^{(i)}$ associée à $\mathcal{M}^{(i)}(t)$ est identique à celle associée à $\mathcal{M}^\star(t)$ si*

$$J_2(t) \leq \xi_i(t) \equiv \frac{1}{2} \min_{\mathbf{x} \in \mathbf{X}^{(i)}} \left(\|\mathbf{x} - 1_{\text{NN}}(\mathbf{x}, \mathcal{M}^{(i)}(t))\|_2^2 - \|\mathbf{x} - 2_{\text{NN}}(\mathbf{x}, \mathcal{M}^{(i)}(t))\|_2^2 \right),$$

où $k_{\text{NN}}(\mathbf{x}, \mathcal{A})$ désigne le k-plus proche voisin de $\mathbf{x}$ dans l'ensemble $\mathcal{A}$.

En combinant ce résultat avec celui du Théorème 10, on obtient directement un nombre suffisant de messages que chaque nœud doit échanger avant que l'un d'eux n'effectue un partitionnement pour assurer une décroissance monotone de J_1 :

Theorème 11. *A tout instant t, J_1 décroit de manière monotone si au moins T messages sont échangés sur le réseau entre deux événements de partitionnement, avec*

$$T = 2N\left(4 \ln N + \frac{6N(\ln N + 1)}{1 - \delta} - \ln \xi_i(t) + \ln DK + \ln \frac{2}{1 - \delta}\right) \tag{2.36}$$

En supposant une activation poissonnienne homogène des nœuds émetteurs et une sélection uniforme des destinataires des messages, on peut alors établir une borne probabiliste sur la valeur du paramètre M a fixer pour assurer la diminution de J_1. En effet, l'identité du nœud concerné par l'événement t suit alors une loi uniforme, ce qui implique que le nombre d'événements de mise à jour entre deux événements de partitionnement suit une loi exponentielle de taux $1/M$. On obtient alors une borne inférieure sur M suffisante pour garantir la diminution de J_1 avec une probabilité au moins δ.

$$M \geq 2\left(4 \ln N + \frac{6N(\ln N + 1)}{1 - \delta} - \ln \xi_i(t) + \ln DK + \ln \frac{2}{1 - \delta}\right) \tag{2.37}$$

Notons que la décroissance de J_1 est suffisante mais non-nécessaire pour en assurer la convergence vers un minimum. En pratique, on peut ainsi se permettre de fréquentes augmentations de J_1 dès lors qu'elles sont compensées par des chutes régulières significatives. Les conditions de convergence de ce type de perturbations apparaissent néanmoins non-triviales à identifier. En déterminer un ou plusieurs critères permettra notamment d'obtenir dans de futurs travaux une borne plus favorable pour le paramètre M. Nos expérimentations suggèrent notamment que ce critère est intimement lié à la densité intrinsèque de l'ensemble d'entraînement et revêt donc un caractère spécifique à chaque problème d'apprentissage. Dans la section 7, nous illustrons ce phénomène de manière expérimentale en montrant qu'une valeur de M beaucoup plus faible que cette borne suffisante permet malgré tout une bonne convergence d'AGKM sur des données synthétiques et réelles.

Convergence d'AGKM

Si l'on respecte les conditions de décroissance de J_1 établies par le Théorème 11, J_1 converge progressivement vers un minimum local, à l'instar de K-means. Lorsque ce minimum local est atteint, le dictionnaire consensus se stabilise et devient constant. Les étapes de partitionnement n'améliorent alors plus ni la MSE locale, ni J_1. N'ayant plus d'impact sur les partitions locales, elles n'en ont donc plus non plus sur les estimées $(\mathbf{s}_k, w_k)$. Dès lors, $\mathcal{M}^\star$ est constant et seuls les événements de mise à jour modifient les $(\mathbf{s}_k, w_k)$. Ces estimées sont ainsi libres de converger vers leur consensus sans être perturbées par les événements de partitionnement devenus sans effet. Le théorème 10 nous garantit alors que les prototypes locaux convergent exponentiellement vers les prototypes consensus.

Finalement, toute convergence de J_1 implique systématique la convergence de J_2 vers 0, ce qui achève la preuve de convergence d'AGKM. Insistons sur le fait qu'il existe un temps t à partir duquel J_1 et J_2 ne varient plus, ce qui différencie nettement AGKM de nombreuses méthodes de K-means distribuées où cette erreur ré-augmente cycliquement par sauts brutaux du fait de la réinitialisation des estimées à chaque itération. A la différence de ces méthodes, tout nœud peut décider, à partir d'un certain temps considéré localement comme suffisant, de quitter terminer sa procédure d'optimisation et quitter soudainement le réseau, sans coordination avec ses nœuds voisins.

2.5 AGEM

De part ses similitudes avec K-means, l'algorithme EM pour l'optimisation de mélanges de Gaussiennes (GMM) peut être facilement transposé dans le paradigme d'AGKM. En effet, l'étape d'Espérance (E) est localement calculable au même titre que l'étape de partitionnement de K-means, tandis que l'étape de Maximisation (M) n'est qu'une simple moyenne pondérée des échantillons locaux, comme l'étape 2 de K-means. En s'appuyant sur cette propriété générale, nous proposons l'algorithme AGEM (Algorithme 4) qui optimise une GMM de manière totalement décentralisée et asynchrone. AGEM est donc fortement inspiré d'AGKM et en hérite ainsi de nombreuses propriétés, parmi lesquelles

- Une convergence et une stabilité terminale garantie par le mécanisme d'autocorrection (lignes 12-13)
- Une meilleure résistance aux mauvaises conditions initiales, via l'initialisation stochastique des modèles locaux
- Une sensibilité au paramètre critique M, qui y joue le même rôle que dans AGKM.

Comme remarqué dans la section 1, il n'est pas nécessaire de stocker explicitement les responsabilités Q (ligne 11) car elles peuvent être calculées à la volée au moment de leur intégration aux estimées locales (ligne 12). Notons quelques itérations d'AGKM peuvent être utilisées en substitution aux premières itération d'AGEM, tout comme K-means est souvent utilisé pour amorcer EM. Cela évite l'échange et le stockage des matrices de covariance qui peut être lourd en grande dimension.

La généralisation d'AGKM à AGEM peut légitimement s'étendre à de toute méthode d'optimisation non-linéaire dès lors que celle-ci est composée des deux éléments suivants :

1. Une composante potentiellement non-linéaire mais localement calculable (*e.g.*, étape E).

Algorithme 4 AGEM : Asynchronous Gossip Expectation Maximization (en chaque noeud i)

Entrées : Jeu d'échantillons $\mathbf{X} \in \mathbb{R}^{n_i \times D}$
Paramètres : K : nombre de cellules désirées
M : nombre de messages envoyés à chaque itération locale

∘ **Procédure émission**

$\mathbf{Q} \leftarrow$ une matrice aléatoire $K \times n_i$ telle que $\mathbf{Q}^\top \mathbf{1} = \mathbf{1}$
$\forall k, \quad \pi_k \leftarrow \sum_{j=1}^{n_i} \mathbf{Q}_{(k,j)}$; $\quad \mathbf{m}_k \leftarrow \sum_{j=1}^{n_i} \mathbf{Q}_{(k,j)} \mathbf{x}_j$; $\quad \mathbf{C}_k \leftarrow \sum_{j=1}^{n_i} \mathbf{Q}_{(k,j)} \mathbf{x}_j \mathbf{x}_j^\top$
$\forall k, \quad \pi_k^{\text{old}} \leftarrow \pi_k$; $\quad \mathbf{m}_k^{\text{old}} \leftarrow \mathbf{m}_k$; $\quad \mathbf{C}_k^{\text{old}} \leftarrow \mathbf{C}_k$
Boucle
  Pour m **de** 1 **à** M **faire**
    $\forall k, \quad \pi_k \leftarrow \pi_k/2; \quad \mathbf{m}_k \leftarrow \mathbf{m}_k/2; \quad \mathbf{C}_k \leftarrow \mathbf{C}_k/2$
    $j \leftarrow$ nœud voisin aléatoire
    Envoyer $(\pi_k, \mathbf{m}_k, \mathbf{C}_k)_{k=1}^K$ à j
  Fin Pour
  $\forall k, \quad \boldsymbol{\mu}_k \leftarrow \mathbf{m}_k/\pi_k$; $\quad \boldsymbol{\Sigma}_k \leftarrow \mathbf{C}_k/\pi_k$
  $\forall j, \ \mathbf{Q}_{(k,j)} \leftarrow \frac{1}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}_k|}} \exp\left[-\frac{1}{2}(\mathbf{x}_j - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\mathbf{x}_j - \boldsymbol{\mu}_k)\right]$
  $\pi_k^{\text{new}} \leftarrow \sum_{j=1}^{n_i} \mathbf{Q}_{(k,j)}$; $\quad \mathbf{m}_k^{\text{new}} \leftarrow \sum_{j=1}^{n_i} \mathbf{Q}_{(k,j)} \mathbf{x}_j$; $\quad \mathbf{C}_k^{\text{new}} \leftarrow \sum_{j=1}^{n_i} \mathbf{Q}_{(k,j)} \mathbf{x}_j \mathbf{x}_j^\top$
  $\pi_k \leftarrow \pi_k + \pi_k^{\text{new}} - \pi_k^{\text{old}}$; $\quad \mathbf{m}_k \leftarrow \mathbf{m}_k + \mathbf{m}_k^{\text{new}} - \mathbf{m}_k^{\text{old}}$; $\quad \mathbf{C}_k \leftarrow \mathbf{C}_k + \mathbf{C}_k^{\text{new}} - \mathbf{C}_k^{\text{old}}$
Fin Boucle

∘ **Procédure réception**

Boucle
  Attendre réception d'un message $(\pi'_k, \mathbf{m}'_k, \mathbf{C}'_k)_{k=1}^K$
  $\forall k, \quad \pi_k \leftarrow \pi_k + \pi'_k; \quad \mathbf{m}_k \leftarrow \mathbf{m}_k + \mathbf{m}'_k; \quad \mathbf{C}_k \leftarrow \mathbf{C}_k + \mathbf{C}'_k$
Fin Boucle

2. Une composante potentiellement non-séparable mais linéaire (*e.g.*, étape M)

En effet, cette distinction entre composante non-linéaire séparable et non-séparable linéaire est constitutive de la plupart des fonctions de coût utilisées en Machine Learning, qui considèrent une contribution additive de chaque échantillon au risque empirique global :

$$J = \frac{1}{n} \sum_{\mathbf{x} \in \mathbf{X}} f(\mathbf{x}) = \mathbb{E}_{\mathbf{X}} f \tag{2.38}$$

En particulier, ces modèles de coût ne considèrent pas d'interaction complexe entre échantillon (causalité, dépendance statistique mutuelle, relation fonctionnelle déterministe entre échantillons, etc). Dans ces cas de figure, l'adaptation décentralisée et asynchrone des schémas d'optimisation des modèles associés peut s'avérer non-triviale. Néanmoins, la grande diversité des problèmes de Reconnaissance de Forme exprimés suivant cette distinction non-linéaire séparable/non-séparable linéaire et leurs nombreuses applications concrètes montrent la large expressivité du paradigme porté par AGKM et AGEM à travers les protocoles Gossip asynchrones perturbés.

Notre analyse théorique s'est concentrée sur le cas d'AGKM. Certains résultats établis pour AGKM se généralisent aisément à AGEM, en fixant $\boldsymbol{\theta} = \{\boldsymbol{\theta}_i\}_{i=1}^N$ avec $\boldsymbol{\theta}_i = \{(\pi_k^{(i)}, \mathbf{m}_k^{(i)}, \mathbf{C}_k^{(i)})\}_{k=1}^K$

et en définissant les fonctions des fonctions de coût semblables :

$$J_1(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{i=1}^{N}\sum_{\mathbf{x}\in\mathbf{X}^{(i)}} \ln \mathcal{L}(\boldsymbol{\theta}_i \mid \mathbf{x}) \tag{2.39}$$

$$J_2(\boldsymbol{\theta}) = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{K} \left\| (\pi_k^{(i)}, \mathbf{m}_k^{(i)}, \mathbf{C}_k^{(i)}) - (\pi_k^{(j)}, \mathbf{m}_k^{(j)}, \mathbf{C}_k^{(j)}) \right\|_2^2 \tag{2.40}$$

Toutefois nous laissons sa preuve de convergence en perspective.

2.6 Résultats Expérimentaux

Cette section est dédiée à l'analyse expérimentale d'AGKM. Cet algorithme est évalué sur des données synthétiques générées par des mélanges de lois normales et sur des jeux de données naturelles couramment utilisés dans la littérature, à savoir MNIST [156] et Inria Holidays [136]. Son exécution a été effectuée en conditions réelles jusqu'à 200 noeuds et en simulation jusqu'à 10^5 noeuds.

Les expériences en conditions réelles ont été réalisées sur un cluster de calcul comportant 200 cœurs de calcul répartis sur 7 machines sur un réseau local. Les expériences simulées ont été conduites en générant des horloges Poissonniennes indépendantes pour chaque nœud afin d'assurer un fonctionnement synchrone, et différentes topologies de réseaux ont été considérées. Ces horloges Poissonniennes permettent de simuler les temps de calcul théoriques des étapes de partitionnement et d'échange de message pour rendre compte de différents environnements matériels (fréquences des processeurs, débit et latence réseau, etc) plus ou moins hétérogènes, via deux paramètres λ_{part} et λ_{com} spécifiques à chaque nœud.

La convergence d'AGKM est caractérisée par la mesure des deux fonctions de coût J_1 et J_2 définies dans la section précédente, en faisant varier :

- La taille du jeu de données (dimension D et nombre d'échantillons n)
- Le paramètre critique M
- La taille du réseau (nombre de noeuds N)
- La connectivité du réseau
- Les vitesses de partitionnement et de communication λ_{part} et λ_{com}

Dans tous les cas, nous avons adopté une sélection uniforme des destinataires des messages parmi le voisins directs des émetteurs.

Rappelons que dans le cas spécifique d'AGKM, J_1 mesure la MSE globale du dictionnaire consensus (*i.e.* sur l'ensemble des données du réseau) tandis que J_2 mesure l'erreur euclidienne totale entre les dictionnaires locaux et le dictionnaire consensus. Lorsque les valeurs explicites des paramètres ne sont pas spécifiées, leurs valeurs par défaut sont définies selon le tableau 2.2.

La figure 2.1 montre graphiquement le comportement d'AGKM au cours du temps sur un problème jouet. 10^4 échantillons ont été générés sur 10 nœuds en proportions égales par échantillonnage de mélanges de lois normales bidimensionnelles, avec des paramètres distincts en

DONNÉES	RÉSEAU	ALGORITHME
$n = 10^5$	$N = 100$	$M = 10$
$D = 2$	Connectivité complète	$K = 32$
Répartition équitable	$\lambda_{\text{part}} = 1$	Initialisation : Partitions aléatoires
Assignation uniforme	$\lambda_{\text{com}} = 1$	(loi d'assignation uniforme)

TABLEAU 2.2 – Valeurs par défaut des paramètres pour AGKM

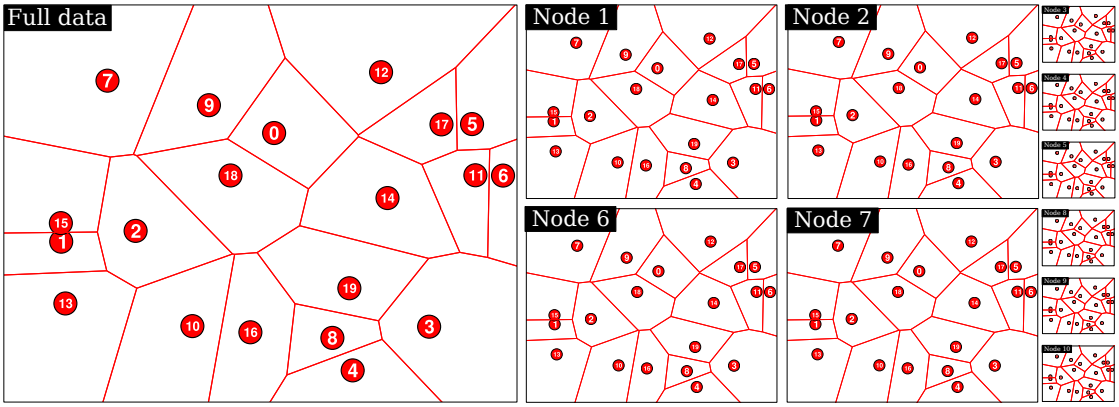

FIGURE 2.1 – Résultat produit par AGKM sur un jeu de données synthétiques. Chaque couleur représente une catégorie obtenue (ici $K = 20$). Tous les noeuds, bien qu'ils n'hébergent chacun qu'un petit sous-ensemble d'entraînement, converge vers un quantificateur de Voronoi identique (frontières en rouge). Ce quantificateur modélise correctement les catégories présentes dans les données du réseau (en haut à gauche).

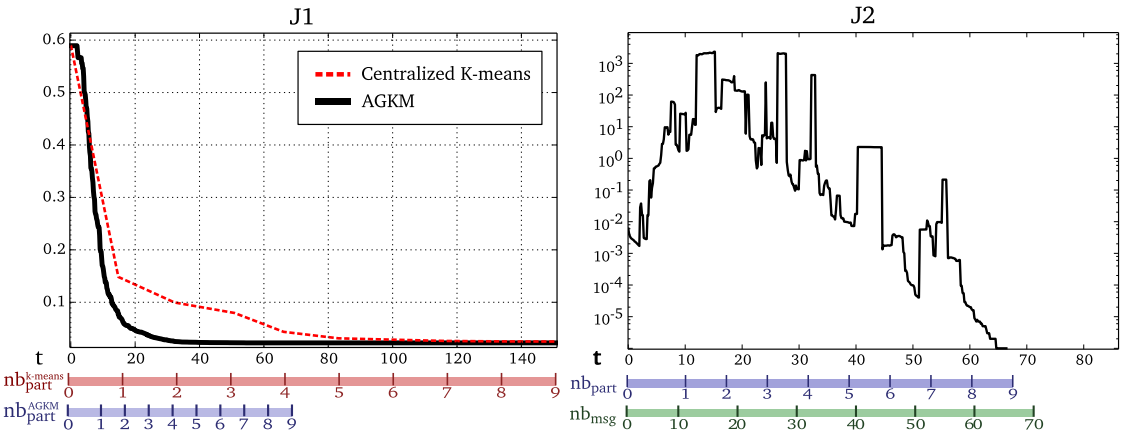

FIGURE 2.2 – J_1 converge vers le même minimum local de la MSE qu'un K-means centralisé, mais plus rapidement. J_2 tend vers zéro exponentiellement. Ici, nb_{part}^{AGKM},$nb_{part}^{\text{K-means}}$ désignent respectivement le nombre moyen de partitions calculées en chaque noeud et le nombre d'itérations de K-means. nb_{msg} désigne le nombre moyen de messages émis par chaque nœud.

chaque nœud. La distribution des échantillons est hétérogène (pas de recouvrement des densités). Les frontières en rouge délimitent les partitions de Voronoi calculées en chaque nœud. On observe que toutes les partitions locales sont identiques (minimisation de J_2), et qu'elles modélisent correctement des catégories intrinsèques aux données du réseau (minimisation de J_1). A la différence de la plupart des algorithmes de K-means distribués, on ne distingue pas d'étapes alternées de réduction de J_1 puis de J_2. Au contraire, les deux objectifs sont minimisés simultanément pour finalement se stabiliser sur un minimum conjoint.

La figure 2.2 montre l'évolution de J_1 et J_2 sur un problème synthétique de 10000 échantillons générés par 30 lois normales quelconques en dimension 16 et répartis sur $N = 100$ nœuds. On observe que J_1 décroit de manière lisse et monotone. En comparaison avec un K-means centralisé classique (courbe pointillée en rouge), cette décroissance est plus rapide (on bénéficie de la parallélisation des calculs), et l'on peut noter que le nombre de calculs de partition effectué en moyenne par chaque nœud est semblable au nombre d'itérations requises par K-means pour converger. On peut ainsi en déduire que la nature perturbée du processus n'impacte pas significativement la poursuite de l'objectif J_1. Par contraste, J_2 subit des variations plus chaotiques, du fait des perturbations induites par les auto-corrections fréquentes. Néanmoins, J_2 exhibe une convergence exponentielle et le nombre de messages envoyés par chaque nœud reste inférieur à N, ce qui implique que notre méthode est plus efficiente en coût de calcul qu'une méthode basée sur un mécanisme de diffusion (*broadcast*) qui consommerait ici environ $9.N = 900$ messages par nœud (*i.e.*, 9 itérations de K-means).

Les figures 2.3 et 2.4 évaluent AGKM sur des jeux de données réels largement utilisés dans la littérature, respectivement Inria Holidays [136] composé de 4,5 millions de descripteurs SIFT de dimension 128 et MNIST [156] composé de 60000 images en niveau de gris de chiffres manuscrits en dimension 784. Dans la figure 2.3 $M = 30$ garantit une convergence systématique de l'algorithme. Dans la figure 2.4 M est fixé à 20 et l'on fait varier la taille du réseau $N = \{10, 100, 1000, 2000\}$. On remarque alors que si J_1 tend toujours assez rapidement vers un minimum local (avec une légère imprécision souvent négligeable), $M = 20$ n'est pas adapté pour $N > 1000$, puisque J_2 ne converge plus vers 0. Ce phénomène illustre l'existence d'un

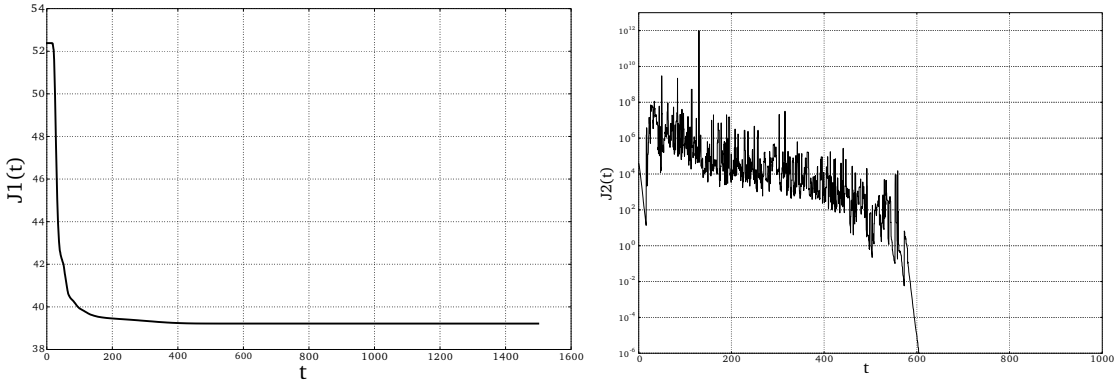

FIGURE 2.3 – Convergence d'AGKM dans un environnement large-échelle typique sur le jeu de données académique Inria Holidays ($n \simeq 4.10^6, D = 128$), réparti sur 5000 noeuds ($M = 30$).

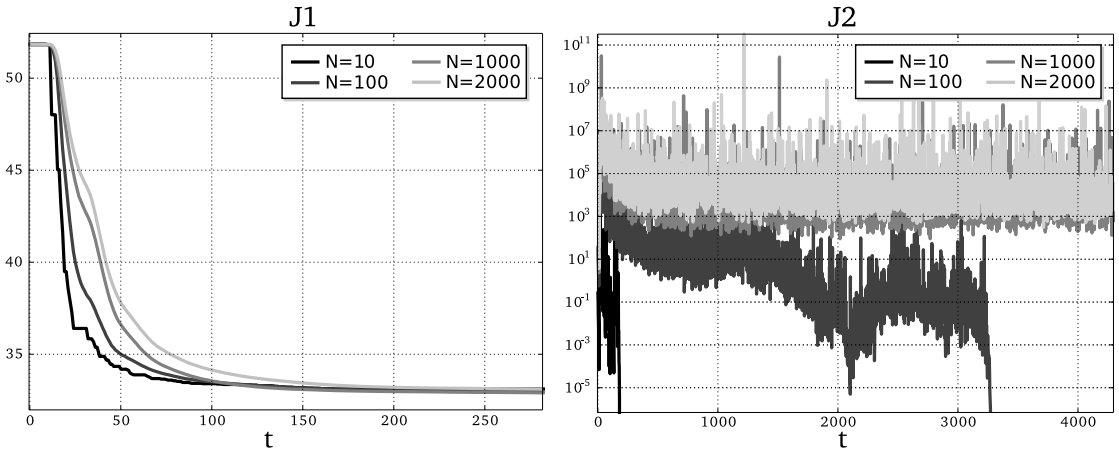

FIGURE 2.4 – Convergence d'AGKM sur le jeu de données MNIST avec $M = 20$ pour différentes tailles de réseau N. Bien que J_1 tende rapidement à proximité d'un local minimum, la convergence de J_2 peut être perdue lorsque $N \geq 1000$, illustrant la criticité du paramètre M et sa dépendance sur N.

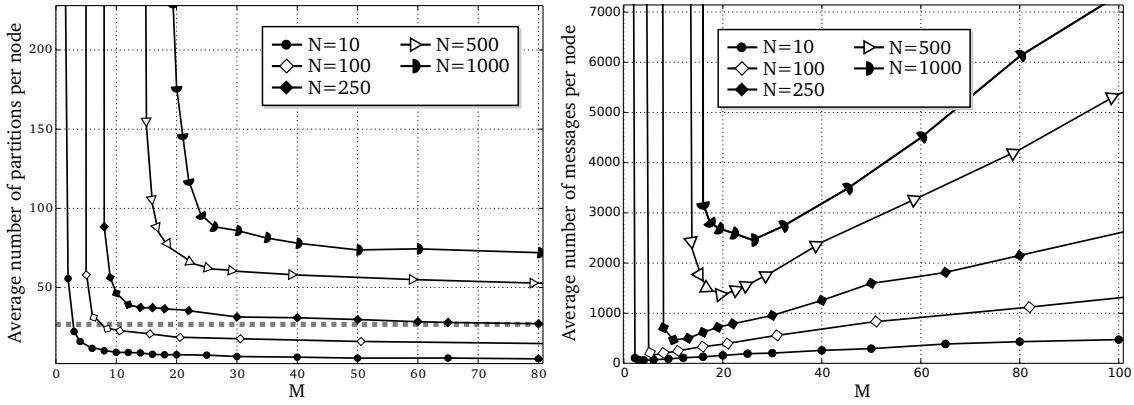

FIGURE 2.5 – Temps de convergence d'AGKM pour différentes tailles de réseau N et valeurs de M, en termes de (a) nombre moyen de partitions calculées en chaque nœud, (b) nombre moyen de messages envoyés par chaque nœud.

seuil critique en dessous duquel M ne peut pas descendre sans mettre en péril la convergence de J_2, ce seuil étant fonction de la taille du réseau N. Notons toutefois que cette criticité ne concerne que la convergence finale du processus d'optimisation, puisqu'en pratique J_1 atteint systématiquement un minimum local avec une précision acceptable.

Étude du paramètre critique M

Dans la Figure 2.5, nous analysons l'effet du paramètre M sur la convergence d'AGKM à travers les coûts de calcul et de communication impliqués. On considère ici que la convergence est atteinte lorsque J_1 se stabilise parfaitement (sa variation dans le temps est devenue nulle) et que J_2 tombe sous 10^{-10}. Les coûts de calcul et de communication sont respectivement mesurés en termes du nombre moyen de partitions calculées et du nombre moyen de messages émis par chaque nœud sur le jeu de données MNIST, en moyennant les résultats sur 100 exécutions avec des partitions initiales aléatoires.

La figure 2.5 confirme l'intuition que plus M est élevé, plus le coût de communication s'accroît mais le nombre de calculs de partitions diminue. La détermination d'une valeur de M adaptée relève donc d'un compromis entre les coût de calcul et de communication adapté aux ressources offertes par le contexte d'exécution (typologie du réseau, vitesse des processeurs, etc). Les asymptotes verticales remarquables dans le graphe de droite montrent la singularité qui se produit lorsque M est fixé sous son seuil critique. Au dessus de ce seuil, les coûts de communication augmentent linéairement tandis que sous ce seuil l'algorithme ne converge plus.

Ce seuil de validité de M apparaît clairement comme une fonction sous-linéaire de N (par exemple, l'augmentation de M requise par le passage de $N = 500$ à $N = 1000$ et moindre que celle pour passer de $N = 250$ à $N = 500$). Notons que la loi précise de ce seuil ne peut être précisément estimée expérimentalement, car la variance de notre mesure entre deux exécutions explose lorsque l'on approche de la singularité. Déterminer une valeur fixe idéale pour M semble donc difficile et hasardeux en pratique.

Finalement, les meilleurs coûts de communication obtenus restent élevés comparés à des approches par diffusion (*broadcast*) ou par arbre couvrant, dont le coût en nombre de message par nœud est respectivement de l'ordre de $\mathcal{O}(N)$ et $\mathcal{O}(1)$.

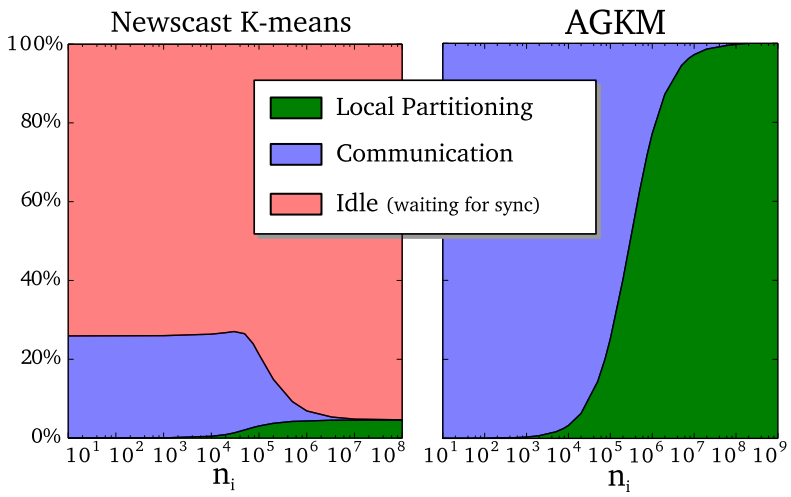

FIGURE 2.6 – Comparaison entre AGKM et un algorithme de type Newscast. A la différence de Newscast où l'attente de réponse des nœuds voisins devient la principale activité des nœuds, dans AGKM cette attente est par conception totalement absente.

Avantages vis-à-vis des méthodes existantes

Nous illustrons ici les avantages d'AGKM par rapport à des méthodes basées sur des échanges localement synchrones ou des itérations globalement synchrones.

La figure 2.6 compare AGKM à un algorithme de K-means basé sur le protocole Newscast [140], qui utilise donc des communications synchrones entre couples de noeuds voisins. Nous mesurons le temps total passé par les nœud dans les trois activités possibles : partitionnement, communication ou attente de la réponse d'un voisin. En simulant diverses vitesses respectives de partitionnement et de communication via les paramètres $\lambda_{\text{part}}^{(i)}$ et $\lambda_{\text{com}}^{(i)}$, la figure 2.6, montre que par l'utilisation de Newscast, les nœuds passent la majeure partie de leur temps à attendre la réponse des nœuds voisin qu'ils contactent. Au contraire, AGKM ne comportant aucune composante de synchronisation entre nœuds, ce temps d'attente n'existe pas.

La figure 2.7 compare AGKM à Epidemic K-means [89]. Tous deux étant conçus pour fonctionner en environnement totalement asynchrone, nous ne fixons aucune hypothèse de synchronie globale comme c'est le cas dans [89]. A cause de l'absence de mécanisme d'auto-correction comme mentionné dans la section 5, Epidemic K-means ne conduit jamais J_2 vers 0, puisque les estimées investies dans le protocole Gossip sont remises à zéro à chaque partitionnement local. J_1 est alors fortement perturbée et ne converge pas vers un minimum local.

Influence de la topologie du réseau

Pour illustrer l'impact de la topologie du réseau, la figure 2.8 rapporte l'évolution de J_1 pour un réseau petit-monde de type Barabasi-Albert (BA) [16], c'est à dire, avec une distribution des degrés des noeuds (*i.e.*, nombre de voisins) en loi de puissance. Ce modèle de réseau est construit de manière incrémentale, en ajoutant à un réseau de $N-1$ nœuds un N-ième nœud en le connectant à chaque nœud existant i avec la probabilité $p_i = k_i / \sum_j^{N-1} k_j$, où k_i est le nombre de nœuds déjà connectés à i (*i.e.*, le degré de i). Cette règle dite d'"attachement préférentiel" entraîne que les nœuds ayant beaucoup de voisins tendent à en avoir de plus en

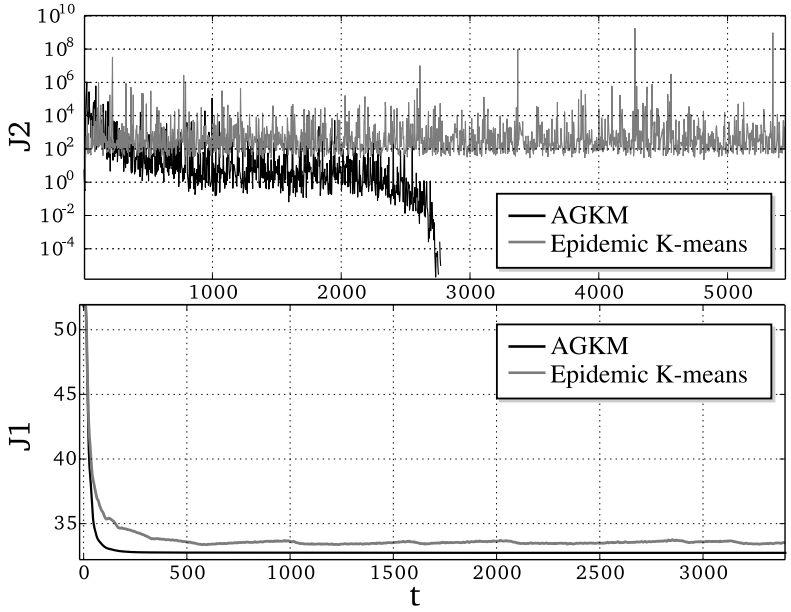

FIGURE 2.7 – Comparaison entre AGKM et Epidemic K-means. A la différence d'AGKM, l'absence de mécanisme d'auto-correction dans Epidemic K-means implique que J_2 ne converge pas et que J_1 ne tend pas vers un minimum local.

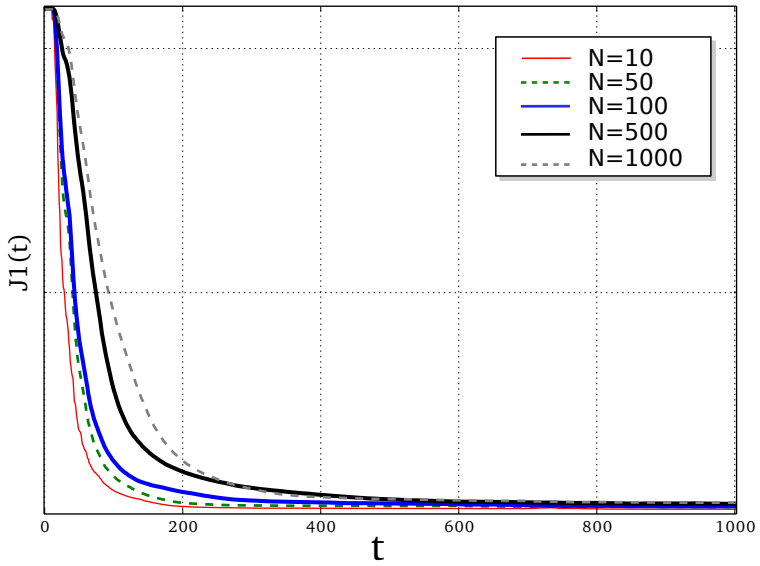

FIGURE 2.8 – Les propriétés de convergence d'AGKM se maintiennent lorsque le graphe du réseau est de type Barabasi-Albert.

plus, tandis que les nœuds plus isolés restent peu connectés (effet de Matthew [177]). On obtient alors un réseau de type petit-monde (*small-world*), caractérisé par

1. Une immense majorité de nœuds peu connectés (très peu de voisins)
2. Une faible distance moyenne entre nœuds, en termes de nombre de liens à franchir (*average path length*). Cette distance est notamment sous-logarithmique par rapport au nombre de nœuds (contrairement aux réseaux de type grille, anneau, etc)

Ce type de construction recouvre une large diversité de réseaux réels point-à-point (*e.g*, réseaux de capteurs, réseaux mobiles, Internet, réseaux sociaux, etc). Formellement, le nombre $N(k)$ de nœuds ayant k voisins suit une loi de puissance :

$$\forall k \in \mathbb{N}, \quad N(k) \sim k^{-3} \tag{2.41}$$

Comme le montre la figure 2.8, la convergence d'AGKM n'est pas significativement affectée par l'usage d'une topologie BA. Cette propriété est clairement héritée de la propriété de mixage rapide des protocoles Gossip et du gap spectral élevé des matrices de communication associées aux graphes BA (cf chapitre 1).

2.7 Conclusion

Dans ce chapitre, nous avons présenté deux méthodes de catégorisation fonctionnant de manière totalement asynchrone et décentralisé. AGKM est dédiée à la quantification vectorielle (*e.g*, K-means), et AGEM optimise des modèles en mélange de Gaussiennes (GMM). Toutes deux ciblent les larges réseaux dynamiques (réseaux de capteurs, réseaux mobiles, etc) pour lesquels ils est difficile voir impossible d'assurer une coordination centrale par un nœud maître ou des communication synchrones, ne serait-ce que localement.

Nous avons montré à la fois théoriquement et expérimentalement que ces algorithmes convergent en optimisant conjointement et simultanément les deux objectifs que sont *(i)* la fidélité d'estimation de la densité des données (erreur quadratique moyenne du quantificateur pour AGKM et vraisemblance de la GMM pour AGEM) et *(ii)* l'égalité des modèles en tous les nœuds. Cela garantit qu'à la suite de la phase d'entraînement distribuée, le traitement d'un échantillon en phase d'utilisation n'impliquera pas de communication entre nœuds et sera rigoureusement identique quel que soit le nœud utilisé.

En dépit de cette convergence formellement établie, la décentralisation et l'asynchronisme des algorithmes proposés leur confère des propriétés comportementales singulières. Notamment, le paramètre critique M, qui détermine le nombre de messages envoyés par chaque nœud entre deux étapes de calcul local, joue un rôle important dans la convergence du processus global.

Enfin, le schéma algorithmique commun à AGKM et AGEM montre que l'on peut appliquer le protocole Gossip asynchrone perturbé (1.88) introduit dans le chapitre 1 aux problèmes non-linéaire de catégorisation. En effet, le terme non-séparable linéaire correspond ici au moyennage des dictionnaires locaux vers un dictionnaire consensus, tandis que le terme non-linéaire séparable f_t est représenté par l'opération de partitionnement local.

Les travaux actuels et futurs que nous menons concernant ce chapitre incluent :

- Une analyse théorique plus fine de la relation liant M à la topologie et la taille du réseau.

- Une caractérisation théorique et expérimentale de l'effet de la densité des données et l'hétérogénéité de sa répartition à travers le réseau sur les garanties de convergence du processus, en posant notamment des hypothèses de distribution *a priori*.

- L'extension de ce schéma algorithmique aux modèles intégrant une composante de réduction de dimension locale aux catégories (MPPCA [45, 235], MoFA [104], MMSC [114])

CHAPITRE 3

Réduction de dimension

La réduction de dimension joue un rôle important dans la résolution de problèmes de Machine Learning large échelle où les données d'entrées consistent en un très grand nombre d'échantillons de très grande dimension. Dans la plupart des cas, la dimension intrinsèque des données est très faible comparée à leur dimension extrinsèque, (*i.e.*, dimension de l'espace d'entrée). La réduction de dimension vise ainsi à projeter les échantillons d'entrée dans un espace de dimension plus petite de sorte à retenir optimalement l'information portée par les données originales. L'objectif final est généralement que l'erreur induite sur les étages suivants de la chaîne de traitement soit minimale, voire progresse (*e.g*, classification, régression, reconstruction, *etc*).

La plupart des méthodes de réduction de dimension classiques on été formulées pour des environnements centralisés, où toutes les données sont disponible en un unique site et où la solution peut être calculée par une seule machine. Les approches centralisées supposent en particulier que cette unique machine possède une mémoire suffisante pour stocker tous les résultats intermédiaires. Cependant, cette solution est irréaliste dans un grand nombre de cas d'application, où l'ensemble des échantillons d'entraînement forme une gigantesque matrice. Par exemple, dans les applications biomédicales, multimédia ou d'imagerie satellitaire, cette matrice peut occuper plusieurs Tera-octets. Par ailleurs, les approches centralisées ne sont pas utilisables lorsque les échantillons proviennent par nature d'un processus distribué et sont donc par définition réparties sur plusieurs machines. Cela justifie par conséquent la conception d'algorithmes de réduction de dimension distribués, qui tiennent compte de cette répartition des données sur un réseau de multiples nœuds de calcul

En environnement distribué, deux scénarios typiques peuvent se présenter. Dans le premier, que nous appellerons "Échantillons Distribués" (ED), les échantillons sont dispersés sur les nœuds de calcul et chaque nœud possède toutes les entrées de chacun de ses échantillons. C'est le scénario le plus courant. Dans le deuxième scénario, que nous nommerons "Coordonnées Distribuées" (CD), chaque nœud possède une partie des entrées de chaque échantillon. Un échantillon donné n'est alors jamais observable dans sa totalité en un nœud donné, chacune de ses composantes pouvant être hébergée par un nœud différent. Ce cas survient typiquement lorsque les échantillons sont des mesures successives de plusieurs grandeurs issues d'un même phénomène géographiquement diffus.

Dans ce chapitre, nous présentons l'algorithme AGPCA (*Asynchronous Gossip Principal Components Analysis*), qui calcule l'Analyse en Composantes Principales (Principal Components Analysis, PCA) d'un jeu d'échantillon reparti sur un réseau de manière décentralisée et

asynchrone. Nous montrons qu'AGPCA est adapté aux deux scénarios ED et CD. Sous certaines conditions de rang, nous démontrons théoriquement et expérimentalement que la solution produite en chaque nœud est identique à celle produite par une PCA centralisée exécutée sur l'ensemble des données du réseau.

Le reste de ce chapitre est organisé comme suit : dans la section 1 nous présentons diverses approches pour la réduction de dimension en environnement centralisé, avant de nous concentrer sur l'Analyse en Composantes Principales. La section 2 recense différentes algorithmes de PCA distribuée. Dans la section 3, nous introduisons une approche naïve de la PCA décentralisée et asynchrone, nommée "Late PCA". Dans la section 4, nous détaillons l'algorithme AGPCA qui lève les limitations soulevées par Late PCA. Dans la section 5, nous montrons qu'AGPCA est également adapté au scenarios CD.

3.1 Réduction de dimension centralisée

Étant donné une matrice d'échantillons $\mathbf{X} \in^{D \times n}$, la réduction de dimension consiste à produire des représentations $\mathbf{Y} \in^{q \times n}$ de dimension q où $q \ll D$, qui préservent au mieux certaines propriétés globales et/ou locales des données d'entrée. En dépit de cet objectif général commun, chaque technique de réduction de dimension définit sa propre notion des "propriétés à préserver".

De ce fait, l'évaluation et la comparaison des performances des techniques de réduction de dimension prises isolément pose généralement des problèmes de pertinence des résultats quantitatifs. La notion même de performance appliquée aux méthodes de réduction de dimension est souvent ambiguë, car les représentations réduites sont la plupart du temps des produits intermédiaires qui servent d'entrée à un prédicteur (classificateur, régresseur, *etc*). Dans ce cas, il est souvent plus pertinent d'évaluer les performances en sortie du prédicteur, pour lequel on possède une vérité de terrain (éventuellement incomplète). Par exemple, il est fréquent d'évaluer la réponse d'un classificateur basique (*e.g.*, plus proche voisin) sur un jeu de données labellisées pour attester de la pertinence des représentations réduites sur une tâche réelle. Une exception notable est la tâche de visualisation, qui exploite directement les représentations compressées, en cherchant à préserver les distances, les groupements ou d'autres propriétés locales aux données d'entrée. Notons toutefois qu'étant limitée à trois dimensions de sortie, la visualisation ne permet pas de caractériser les performances en grandes dimensions.

Parmi les propriétés les plus fréquemment considérées, on trouve la préservation des distances, des dissimilarités, des relations de voisinage, des groupements (maximisation de la séparation entre groupe et minimisation de la variance intra-groupe) ou de la densité de probabilité des données. Chacun de ces critères apporte ses avantages et ses inconvénients, qui se traduisent par des succès sur certaines tâches et des échecs sur d'autres tâches. Ainsi, certaines méthodes très performantes sur des données synthétiques échouent totalement sur des données naturelles. Pour une étude comparative approfondie, le lecteur pourra se référer à [238].

3.1.1 Réduction de dimension linéaire

Les techniques de réduction de dimension linéaire sont à la fois les plus anciennes et les plus populaires. Celles-ci formulent les représentations réduites comme une projection linéaire des

données d'entrée :

$$\mathbf{Y} = \mathbf{U}^\top \mathbf{X}, \quad \text{où} \quad \mathbf{U} \in \mathbb{R}^{D \times q} \tag{3.1}$$

Chaque vecteur d'entrée $\mathbf{x}_i \in \mathbf{X}$ est ainsi projeté sur le sous-espace euclidien de dimension $q \ll D$ engendré par la matrice $\mathbf{U}$.

Analyse en Composantes Principales

L'Analyse en Composantes Principales (*Principal Components Analysis*, PCA [125, 195]) est la plus ancienne approche linéaire à la réduction de dimension. Elle calcule la base orthonormale $\mathbf{U}^\star = [\mathbf{u}_1 \dots \mathbf{u}_q]$, $\mathbf{u}_k \in \mathbb{R}^D$ qui retient le maximum de variance dans les données projetées :

$$\mathbf{U}^\star = \underset{\mathbf{U} \in \mathbb{R}^{D \times q}}{\arg\max} \left\| \mathbf{U}^\top \mathbf{X} \mathbf{X}^\top \mathbf{U} \right\|_F^2 \quad \text{avec} \quad \mathbf{U}^{\star\top} = \mathbf{U}^{\star -1} \tag{3.2}$$

La PCA fait ainsi l'hypothèse raisonnable que l'information importante est véhiculée par la variance des données. Cette information est supposée évoluer dans un sous-espace de dimension q, mais être entachée d'un faible bruit de mesure qui la rend corrélée dans $\mathbb{R}^D$. Sous cette interprétation, la PCA poursuit un but semblable à l'analyse factorielle (*Factor Analysis*, FA [85]).

Le principal avantage de la PCA est l'existence d'une solution en forme close. En effet, on montre facilement que $\mathbf{U}^\star$ est composée des q vecteurs propres dominants de la matrice de covariance $\mathbf{C}$, définie comme suit [125] :

$$\mathbf{C} = \frac{1}{n} \mathbf{X}\mathbf{X}^\top - \mu\mu^\top \quad \text{où} \quad \mu = \frac{1}{n}\mathbf{X}\mathbf{1} \quad \text{est la moyenne des échantillons} \tag{3.3}$$

De manière équivalente, la solution de la PCA est la projection linéaire qui conserve au mieux la matrice de Gram [79] (*i.e.*, la matrice des produits scalaires entre chaque paire d'échantillons) :

$$\mathbf{U}^\star = \underset{\mathbf{U} \in \mathbb{R}^{D \times q}}{\arg\min} \left\| \mathbf{X}^\top \mathbf{X} - \mathbf{X}^\top \mathbf{U}\mathbf{U}^\top \mathbf{X} \right\|_F^2 \quad \text{avec} \quad \mathbf{U}^{\star\top} = \mathbf{U}^{\star -1} \tag{3.4}$$

La conservation optimale du produit scalaire fait de la PCA une solution particulièrement adaptée lorsque les données projetées sont transmises à des algorithmes qui reposent sur les produits scalaires entre échantillons plutôt que sur échantillons eux-mêmes (Machines à Vecteurs Supports [61, 117] et méthodes à noyaux de manière générale [28, 122]).

Positionnement multidimensionnel

La méthode par positionnement multidimensionnel (*Multidimensional Scaling*, MDS [63]) est très similaire à la PCA. Elle consiste à préserver les distances entre les vecteurs d'entées, en minimisant l'erreur quadratique moyenne suivante

$$E = \frac{1}{n^2} \sum_{i,j}^{n} \left(d(\mathbf{x}_i, \mathbf{x}_j) - \delta(\mathbf{y}_i, \mathbf{y}_j) \right)^2 \tag{3.5}$$

où $d(\cdot, \cdot)$ est une certaine distance dans $\mathbb{R}^D$ et $\delta(\cdot, \cdot)$ une certaine distance dans $\mathbb{R}^q$.

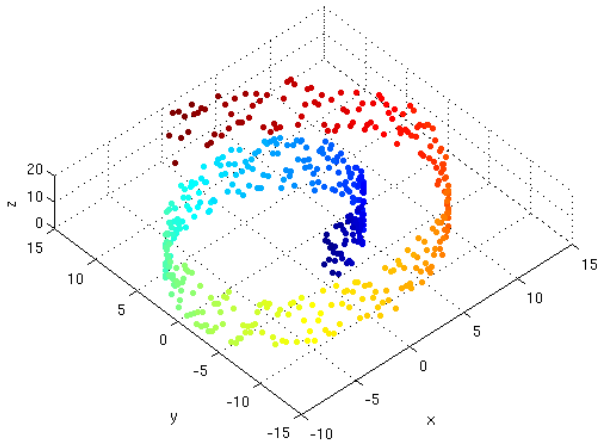

FIGURE 3.1 – Un exemple de variété non-linéaire : le jeu de données *SwissRoll* [234]

Si l'on choisit des distances euclidiennes, le problème d'optimisation correspondant est

$$\mathbf{Y} = \underset{[\mathbf{y}_1,\ldots,\mathbf{y}_n]\in\mathbb{R}^{q\times n}}{\arg\min} \frac{1}{n^2}\sum_{i,j}^{n}\Big(\|\mathbf{x}_i-\mathbf{x}_j\|_2 - \|\mathbf{y}_i-\mathbf{y}_j\|_2\Big)^2 \tag{3.6}$$

La solution de ce problème est alors identique à celle de la PCA :

$$\mathbf{Y} = \mathbf{U}^{\star\top}\mathbf{X} \tag{3.7}$$

On note ainsi que la PCA est également la transformation linéaire qui préservent au mieux les distances euclidiennes et résout donc le MDS euclidien. L'avantage propre au MDS est qu'il repose sur l'expression de la matrice de dissimilarité $\mathbf{D} \equiv [d(\mathbf{x}_i,\mathbf{x}_j)]_{ij}$, qui peut s'appuyer sur une métrique non-euclidienne (*e.g.* fonction de Sammon), voire une relation non-métrique (*e.g.* ordinale) [153].

3.1.2 Réduction de dimension non-linéaire

Les méthodes de réduction de dimension linéaires sont limitées à la préservation des distances euclidiennes. Si ce critère est approprié lorsque l'information est portée par un sous-espace euclidien de $\mathbb{R}^D$, il ne permet pas de représenter des données qui évoluent sur une variété non-linéaire (une illustration populaire de ce type de problème est donnée par le jeu de données *SwissRoll* [234], *cf.* figure 3.1). En effet, deux points dont la distance euclidienne est faible ne sont pas forcément proche sur ce type de variété.

Les techniques de réduction de dimension non-linéaires cherchent donc à préserver des propriétés intrinsèque aux variétés non-linéaires qui supportent les données d'entrée. Pour ce faire, il existe trois familles de méthodes :

1. Les méthodes globales, qui cherchent à reconstruire au mieux une certaine matrice de similarité ou de dissimilarité représentant l'ensemble des vecteurs d'entrée.

2. Les méthodes locales, qui cherchent à représenter la variété localement dans le voisinage que chaque vecteur d'entrée.

3. Les méthode par alignement global de modèles localement linéaire, qui estiment la variété comme l'addition de plusieurs morceaux supposés linéaires.

Méthodes globales

Les méthodes dites globales s'appuient sur la même idée que MDS, dans la mesure où elles construisent une matrice de dissimilarité (ou de similarité) qui rend compte au mieux de la structure de la variété non-linéaire, puis réduisent cette matrice de façon à obtenir des représentations de dimension q pour chaque vecteur d'entrée.

Isomap [234] est similaire au MDS classique, mais considère les distances géodésiques au lieu des distances euclidienne. La distance géodésique est la distance entre deux points mesurée le long de la variété. Comme cette variété est inconnue, sa structure est estimée en créant un graphe de voisinage. Ce graphe de voisinage est construit en reliant chaque point à ses k voisins les plus proches (au sens euclidien), où $k \in \mathbb{N}$ est un paramètre fixé. La distance géodésique le long de la variété est alors approximée par la longueur du plus court chemin dans le graphe de voisinage. On obtient alors une matrice de dissimilarité entre tout couple de vecteurs d'entrée, à laquelle on peut directement appliquer un MDS. Maximum Variance Unfolding (MVU, [250]) est similaire à Isomap, mais tente de maximiser la distance euclidienne dans l'espace d'arriver entre les points qui ne sont pas voisins dans le graphe de voisinage.

Nous avons vu plus haut que la PCA offre une reconstruction linéaire optimale de la matrice de Gram des vecteurs d'entrée. Grâce à l'astuce du noyau, l'approche Kernel PCA [215] propose de remplacer cette matrice de Gram par un noyau de Mercer (semi-défini positif) [176] $\mathbf{K} = [\kappa(\mathbf{x}_i, \mathbf{x}_j)]_{ij}^n$, dont on sait qu'il correspond à une matrice de Gram dans un certain espace induit. En notant $\phi(\mathbf{x})$ l'image d'un vecteur d'entrée $\mathbf{x}$ dans cet espace induit, on a

$$\forall i, j, \quad \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) = \kappa(\mathbf{x}_i, \mathbf{x}_j) \tag{3.8}$$

Or, dans l'espace induit, la projection d'un vecteur d'entrée $\mathbf{x}_i$ sur la $k^{\text{ième}}$ composante principale $\mathbf{v}_k$ de la matrice de covariance est donnée par

$$\mathbf{v}_k^\top \phi(\mathbf{x}_i) = \left(\sum_{j=1}^{n} a_{jk} \phi(\mathbf{x}_j) \right)^\top \phi(\mathbf{x}_i) = \sum_{j=1}^{n} a_{jk} \kappa(\mathbf{x}_j, \mathbf{x}_i) \tag{3.9}$$

On remarque que les $\phi(\mathbf{x}_i)$ n'ont pas besoin d'être calculés explicitement. Par ailleurs, la matrice des coefficients $\mathbf{A} \equiv [a_{ij}]_{ij}^n$ peut être directement obtenue à partir des q vecteurs propres dominants de la matrice du noyau $\mathbf{K}$. Il n'est donc pas nécessaire de diagonaliser ni de calculer la matrice de covariance dans l'espace induit. On peut ainsi utiliser des noyaux dont l'espace induit est de dimension infinie (*e.g.*, noyau Gaussien) pour réaliser une réduction de dimension non-linéaire.

Stochastic Neighbor Embedding (SNE, [119]) représente les données d'entrée par une chaîne de Markov où la probabilité de passer d'un point $\mathbf{x}_i$ à un point $\mathbf{x}_j$ est donnée par un noyau gaussien $\kappa(\mathbf{x}_i, \mathbf{x}_j)$, produisant ainsi une matrice de similarité $n \times n$. SNE cherche alors à préserver cette distribution conditionnelle en minimisant la divergence de Kullback-Leibler entre la distribution d'origine et une distribution de même type (voisinage Gaussien) mais dans l'espace de

dimension réduite q. Le gradient de cette divergence s'apparente alors à une somme de forces attirant les points voisins trop distants et repoussant les points non-voisins trop proche dans l'espace de sortie, à la manière d'un système de ressorts. Des extensions de SNE remplacent la KL-divergence par une mesure symétrique convexe [47, 160]), multi-échelle [159], ou choisissent une distribution à longue traîne (*e.g.*, Student) pour éviter certains problèmes d'agglomération des points [237].

Les auto-encodeurs [69] sont des réseaux de neurones avec un nombre impaire de couches cachée totalement connectées. On attribue q neurones à la couche du milieu, et D neurones à la première et la dernière couche. Un algorithme itératif de rétro-propagation du gradient est alors appliqué de sorte à minimiser l'erreur de reconstruction entre la première couche et la dernière. A convergence, la moitié amont du réseau est un encodeur non-linéaire efficace qui produit une représentation de dimension q des vecteurs d'entrée qui sont optimalement reconstruits par l'autre moitié du réseau. Cette procédure d'optimisation est non-convexe, aussi peut on atteindre des minima locaux non-pertinents. Pour les éviter, diverses techniques de pré-entraînement ont été proposées [120].

Méthodes locales

A la différence des méthodes globales, les méthodes dites locales cherchent à encoder uniquement des propriétés locales de la variété non-linéaire.

Par "propriétés locales", la plupart de ces méthodes entendent "dépendances restreintes à un voisinage entre points". Tout comme Isomap, elles construisent un graphe de voisinage dans lequel chaque point est relier à ses k voisins les plus proches au sens euclidien. Cependant, contrairement à Isomap, elles optimisent des modèles locaux à ces voisinages de manière indépendante. Notons $\mathcal{N}(i,j)$ l'indice du $j^{\text{ième}}$ plus proche voisin de $\mathbf{x}_i$.

Local Linear Embedding (LLE, [209]) approxime la variété au voisinage de chaque point $\mathbf{x}$ en exprimant $\mathbf{x}$ comme une combinaison linéaire de ses voisins. Le voisinage de $\mathbf{x}$ est donc modélisé par un hyperplan, qui est ajusté au sens des moindres carrés

$$\mathbf{W}^\star = \underset{[w_{ij}]\in\mathbb{R}^{n\times k}}{\arg\min} \sum_{i=1}^{n} \left\| \mathbf{x}_i - \sum_{j=1}^{k} w_{ij}\mathbf{x}_{\mathcal{N}(i,j)} \right\|_2^2, \tag{3.10}$$

Sachant $\mathbf{W}^\star$, l'hypothèse de linéarité locale de la variété implique que les représentations réduites $\mathbf{y}_i$ sont celles qui sont le mieux reconstruites par $\mathbf{W}^\star$:

$$\mathbf{Y} = \underset{[\mathbf{y}_1,\ldots,\mathbf{y}_n]\in\mathbb{R}^{q\times n}}{\arg\min} \sum_{i=1}^{n} \left\| \mathbf{y}_i - \sum_{j=1}^{k} w^\star_{ij}\mathbf{y}_{\mathcal{N}(i,j)} \right\|_2^2, \tag{3.11}$$

On peut alors montrer que $\mathbf{Y}$ est donnée par les q vecteurs propres associés aux q plus petites valeurs propres non-nulles de $(\mathbf{I}-\mathbf{W}^\star)^\top(\mathbf{I}-\mathbf{W}^\star)$. Plusieurs extensions de LLE telles que Hessian-LLE [76] et Local Tangent Space Analysis (LTSA, [262]) qui estiment l'espace local tangent à chaque point $\mathbf{x}_i$ en appliquant une PCA à ses k plus proches voisins. LTSA cherche alors la transformation linéaire qui reconstruit au mieux ces espaces tangents à partir des représentations réduites, tandis que Hessian-LLE cherche à minimiser la courbure locale en estimant

puis en décomposant la matrice Hessienne locale. Tous deux obtiennent les représentations réduites $\mathbf{Y}$ à partir des q vecteurs propres associés aux plus petites valeurs propres non-nulles de leurs estimateurs respectifs.

Laplacian Eigenmaps (LE, [17]), comme LLE, construit un graphe de voisinage, mais pondère ce graphe par un noyau Gaussien $\kappa_{\mathrm{G}}(\cdot,\cdot)$: le poids w_{ij} d'une arête entre deux points i et j est donnée par $\kappa_{\mathrm{G}}(\mathbf{x}_i,\mathbf{x}_j)$ si j est un des k plus proches voisins de i et 0 sinon. La matrice d'adjacence $\mathbf{W} \equiv [w_{ij}]_{ij}$ n'a donc que k coefficients non-nuls sur chacune de ses lignes et de ses colonnes. On définit alors la matrice des représentations réduites $\mathbf{Y}$ comme suit :

$$\mathbf{Y} = \underset{[\mathbf{y}_1,\ldots,\mathbf{y}_n]\in\mathbb{R}^{q\times n}}{\arg\min} \sum_{i,j}^{n} w_{ij} \left\|\mathbf{y}_i - \mathbf{y}_j\right\|_2^2 \tag{3.12}$$

Si l'on introduit la matrice $\mathbf{D} \equiv \mathrm{diag}(\mathbf{W1})$ et que l'on calcule le laplacien du graphe pondéré défini par $\mathbf{L} \equiv \mathbf{D} - \mathbf{W}$, la solution de ce problème est alors donnée par les q vecteurs propres généralisés de $\mathbf{L}$ associés aux q plus petites valeurs propres non-nulles, c'est à dire

$$\mathbf{Y} = [\mathbf{v}_1,\ldots,\mathbf{v}_q]^\top \quad \text{avec} \quad \mathbf{L}\mathbf{v}_i = \lambda_i \mathbf{D}\mathbf{v}_i \tag{3.13}$$

Alignement global de modèles locaux

Les méthodes par alignement global de modèles locaux combinent l'ajustement d'un modèle capturant les propriétés locales et l'optimisation d'un critère consistance global. Ces méthodes fonctionnent ainsi en deux étapes

1. Un mélange de modèles linéaires probabilistes (*e.g.*, MoFA [104], MPPCA [235]) est ajusté aux données d'entrée par un algorithme type EM [70]. Chaque vecteur d'entrée $\mathbf{x}_i \in \mathbb{R}^D$ est ainsi converti en K vecteurs $\mathbf{z}_{ik}$ où K est le nombre de composantes du mélange. On obtient également une matrice de responsabilité $\mathbf{R} = [r_{ik}] \in \mathbb{R}^{n\times K}$ où r_{ik} est la probabilité d'appartenance de la donnée $\mathbf{x}_i$ à la composante k.

2. Les représentations réduites $\mathbf{Y} = [\mathbf{y}_i] \in \mathbb{R}^{q\times n}$ sont alors obtenues à partir des représentations locales $\tilde{\mathbf{z}}_{ik}$ en effectuant un alignement global des composantes. Cet alignement a pour but de "dérouler" la variété via une certaine transformation des espaces locaux à chaque composante, de sorte à ceux-ci deviennent linéairement dépendants.

Dans *Manifold Charting* [41], chaque représentation $\mathbf{y}_i$ est ainsi définie comme une combinaison linéaire de représentations $\mathbf{y}_{ik}$ locales à chaque composante, pondérée par les responsabilités r_{ik} associées. Pour chaque composante k, les représentations locales $\mathbf{y}_{ik}$ sont modélisés par une transformation affine $\mathbf{A}_k$ des $\mathbf{z}_{ik}$ issus de l'étape 1 :

$$\forall i, \quad \mathbf{y}_i \equiv \sum_{k=1}^{K} r_{ik}\mathbf{y}_{ik}, \quad \text{où} \quad \mathbf{y}_{ik} \equiv \mathbf{A}_k\mathbf{z}_{ik} \tag{3.14}$$

L'alignement des modèles locaux consiste alors à déterminer le jeu de transformations $\{\mathbf{A}_k\}_k^K$ qui fait correspondre au mieux les représentations locales sur les différentes composantes :

$$J(\mathbf{Y}) = \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} r_{ik}r_{il} \left\|\mathbf{y}_{ik} - \mathbf{y}_{il}\right\|_2^2 \tag{3.15}$$

La pondération par le produit $r_{ik}r_{il}$ implique que l'alignement s'effectue sur les régions où les composantes se superposent, c'est à dire les régions où un même vecteur d'entrée est capturé par plusieurs composantes. Manifold Charting cherche ainsi à réduire l'ambiguïté de projection de ces exemples. On peut montrer [41] que le jeu de transformations qui minimise l'objectif convexe (3.15) peut être obtenu à partir des vecteurs propres généralisés $\mathbf{v}_i$ suivants :

$$(\mathbf{D} - \mathbf{U}^\top \mathbf{U})\mathbf{v}_i = \lambda_i \mathbf{U}^\top \mathbf{U}\mathbf{v}_i \tag{3.16}$$

où $\mathbf{D}$ est une matrice bloc-diagonale de blocs $\mathbf{D}_{(k)}$ et $\mathbf{U}$ est composée des blocs $\mathbf{U}_{(i,k)}$, avec

$$\mathbf{D}_{(k)} = \sum_{i=1}^{n} r_{ik}\text{cov}\{\tilde{\mathbf{z}}_k\} \quad \text{et} \quad \mathbf{U}_{(i,k)} = r_{ik}\tilde{\mathbf{z}}_{ik} \quad \text{où} \quad \tilde{\mathbf{z}}_{ik} = [\mathbf{z}_{ik} \;\; 1] \tag{3.17}$$

et $\text{cov}\{\tilde{\mathbf{z}}_k\}$ désigne la matrice de covariance des $\{\tilde{\mathbf{z}}_{ik}\}_{i=1}^n$.

Dans *Locally Linear Coordination* (LLC, [233]), l'alignement est réalisé entre la matrice $\mathbf{U}$ définie par (3.17) et la matrice $\mathbf{W}^\star$ utilisée dans LLE et définie par (3.10). La première caractérise ainsi la densité des données projetée sur les composantes du modèle mélange tandis que la seconde représente les données comme des combinaisons linéaires entre voisins. L'alignement est optimal est donné par la résolution du problème de valeurs propres généralisées suivant :

$$\mathbf{U}^\top \mathbf{M}\mathbf{M}^\top \mathbf{U}\mathbf{v} = \lambda \mathbf{U}^\top \mathbf{U}\mathbf{v}, \quad \text{avec} \quad \mathbf{M} = (\mathbf{I} - \mathbf{W}^\star)^\top (\mathbf{I} - \mathbf{W}^\star) \tag{3.18}$$

En regroupant les q vecteurs propres généralisés $\mathbf{v}$ associées aux plus petites valeurs propres λ dans une matrice $\mathbf{V}$, on obtient alors les représentations réduites $\mathbf{Y}$ par projection de $\mathbf{U}$ sur $\mathbf{V}$.

3.1.3 Resolution des problèmes de valeurs propres

On constate qu'une grande majorité des techniques de réduction de dimension, qu'elles soient linéaires ou non-linéaires, repose sur la résolution d'un problème de valeurs propres (éventuellement généralisé). Dans la plupart des cas, la matrice considérée est semi-définie positive (ses valeurs propres sont donc réelles et $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$). On ne s'intéresse de plus qu'à un nombre réduit q de ses vecteurs propres, ce qui ne requiert pas de la diagonaliser complètement. On peut alors distinguer ces méthodes selon qu'elles cherchent les valeurs propres dominantes ou les valeurs propres les plus petites.

Considérons une matrice semi-définie positive $\mathbf{A} \in \mathbb{R}^{n\times n}$ dont on souhaite déterminer q vecteurs propres. Pour les calculer, il existe des nombreuses méthodes numériques. Selon les caractéristiques de $\mathbf{A}$, l'une ou l'autre de ces méthodes s'avérera plus rapide, plus précise ou plus stable. En particulier, le conditionnement du problème, déterminé par l'écart entre deux valeurs propres distinctes, joue un rôle important dans leur convergence.

Méthode de la puissance itérée

La méthode de la puissance itérée (*Power Iteration*, PI) est la plus simple. En partant d'un vecteur $\mathbf{x} \in \mathbb{R}^n \setminus \mathbf{0}$ quelconque, elle modifie itérativement $\mathbf{x}$ de sorte à ce que $\mathbf{x}$ converge vers le vecteur propre dominant de $\mathbf{A}$. Pour ce faire, $\mathbf{x}$ est multiplié par $\mathbf{A}$ et normalisé de manière répétée :

$$\forall t \geq 0, \quad \mathbf{x}(t+1) = \frac{\mathbf{A}\mathbf{x}(t)}{\|\mathbf{A}\mathbf{x}(t)\|} \tag{3.19}$$

Observons que cette récursion entraîne $\mathbf{x}(t) = \frac{\mathbf{A}^t\mathbf{x}(0)}{\|\mathbf{A}^t\mathbf{x}(0)\|}$. Introduisons la diagonalisation de $\mathbf{A} = \mathbf{P}\mathbf{D}\mathbf{P}^\top$ avec $\mathbf{P} = [\mathbf{v}_1, \ldots, \mathbf{v}_n]$ et $\mathbf{D} = \text{diag}(\lambda_1, \ldots, \lambda_n)$, et exprimons $\mathbf{x}(0)$ dans la base orthonormale $\mathbf{P}$ par $\mathbf{x}(0) = \sum_{i=1}^n \alpha_i \mathbf{v}_i$. On a alors

$$\mathbf{A}^t\mathbf{x}(0) = \mathbf{P}\mathbf{D}^t\mathbf{P}^\top\mathbf{x}(0) = \sum_{i=1}^n \lambda_i^t \alpha_i \mathbf{v}_i = \alpha_1\lambda_1^t \left(\mathbf{v}_1 + \sum_{i=2}^n \frac{\alpha_i}{\alpha_1}\left(\frac{\lambda_i}{\lambda_1}\right)^t \mathbf{v}_i\right) \tag{3.20}$$

$$= \alpha_1\lambda_1^t\mathbf{v}_1 + \mathcal{O}\left(\frac{\lambda_2}{\lambda_1}\right)^t \tag{3.21}$$

Donc $\mathbf{x}(t) = \mathbf{v}_1 + \mathcal{O}\left(\frac{\lambda_2}{\lambda_1}\right)^t$. Ainsi, $\mathbf{x}(t)$ converge géométriquement vers le vecteur propre dominant de $\mathbf{A}$ avec un taux en $\mathcal{O}(\lambda_2/\lambda_1)$. Remarquons que cette technique ne fonctionne que si $\lambda_1 > \lambda_2$, c'est à dire que la multiplicité (géométrique et algébrique) de λ_1 est 1 (comme détaillé dans le chapitre 1, les matrices satisfaisant cette propriété sont qualifiées de *primitives*). Ajoutons que la vitesse de convergence de PI est ainsi directement dépendante du gap spectral $\lambda_1 - \lambda_2$.

La méthode de la puissance itérée ne permet que de trouver le vecteur propre dominant. Pour trouver d'autres vecteurs propres sous-dominants, il faut ré appliquer le processus à la matrice $(\mathbf{A} - \lambda_1\mathbf{I})$, qui n'a plus λ_1 comme valeur propre. On trouve alors pour cette matrice une valeur propre dominante λ qui, additionnée à λ_1 constitue la seconde valeur propre de $\mathbf{A}$ et ainsi de suite.

Pour trouver le vecteur propre associé à la plus petite valeur propre, il suffit d'appliquer la méthode de la puissance itérée à $\mathbf{A}^{-1}$. De même, lorsque

Itérations Orthogonales

Dans de nombreuses situations, on ne peut se permettre de calculer les vecteurs propres de manière séquentielle. L'algorithme des itérations orthogonales (OI) étend PI afin de permettre le calcul simultané de q vecteurs propres. Au lieu de mettre à jour un unique vecteur $\mathbf{x}$, OI fait évoluer une base orthonormale $\mathbf{U}$ composée de q vecteurs.

En partant d'une matrice base orthonormale $\mathbf{U}(0) \in \mathbb{R}^{D\times q}$ quelconque, OI répète donc les deux étapes suivantes :

1. Calculer $\hat{\mathbf{U}}(t+1) = \mathbf{A}\mathbf{U}(t)$
2. Orthonormaliser $\hat{\mathbf{U}}(t+1)$ par décomposition QR pour obtenir $\mathbf{U}(t+1)$.

Rappelons ici que la décomposition QR d'une matrice réelle carrée l'exprime comme un produit d'une matrice orthogonale $\mathbf{Q}$ et d'une matrice triangulaire supérieure $\mathbf{R}$. Lorsque la matrice est de rang plein, les colonnes de $\mathbf{Q}$ en forment une base orthogonale. La décomposition QR se généralise aux matrices rectangulaires de taille $n \times q$ avec $n \geq q$, en produisant une base orthogonale $\mathbf{Q} \in \mathbb{R}^{n\times q}$ et une matrice triangulaire supérieure $\mathbf{R} \in \mathbb{R}^{q\times q}$ (cette décomposition est souvent appelée décomposition QR économique, ou réduite). La décomposition QR peut être calculée de nombreuses manières plus ou moins efficientes et stables, par exemple par le procédé de Gram-Schmidt, les transformation de Householder (notamment en mettant la matrice en question sous forme Hessenberg), ou les rotations de Givens [109].

A convergence, OI fournit une matrice $\mathbf{Q}$ qui contient les q vecteurs propres principaux de $\mathbf{A}$. Notons que les valeurs propres correspondantes se trouvent alors sur la diagonale de la matrice triangulaire $\mathbf{R}$ issue de la décomposition QR.

Algorithme QR

Il est possible de se passer de la multiplication de $\mathbf{A}$ par $\mathbf{Q}(t)$ effectuée dans OI. C'est ce que réalise l'algorithme QR [100, 154] qui répète les deux étapes suivantes :

1. Exécuter une décomposition QR de $\mathbf{U}(t) = \mathbf{Q}(t)\mathbf{R}(t)$
2. Calculer $\mathbf{U}(t+1) = \mathbf{R}(t)\mathbf{Q}(t)$

L'algorithme QR offre ainsi une meilleure stabilité numérique, car il ne passe pas par la matrice intermédiaire $\hat{\mathbf{U}}$ qui peut être mal conditionnée. De plus, il se prête à une réécriture particulièrement efficiente lorsque la matrice d'entrée est fournie sous forme Hessenberg (seuls les éléments situés dans le triangle supérieur et sur la sous-diagonale sont non-nuls) [191].

Des extensions à cette procédure ont été proposées, telles que l'algorithme de Lanczos [155] qui tire profit de toutes les estimées précédentes $\mathbf{U}(1), \ldots, \mathbf{U}(t)$ pour obtenir une vitesse de convergence accrue.

3.1.4 Pertinence de l'Analyse en Composantes Principales

Une large partie des méthodes non-linéaires présentées dans les sections précédentes s'appuient sur la construction d'un graphe de voisinage. Cette opération, bien qu'elle ne soit réalisée qu'une seule fois, peut s'avérer très coûteuse. Elle est de plus victime de la "malédiction de la dimension" dont les effets notamment sur la concentration des distances induisent :

1. L'apparition de court-circuits, *i.e.*, des voisinages non-pertinents qui relient en réalité des points très éloignées sur la variété qui supporte les données [162].
2. L'apparition de hub/anti-hubs, *i.e.*, des points dont presque tous les autres points (respectivement aucun) sont les voisins [205].
3. Une croissance exponentielle du nombre de points requis pour caractériser localement la variété dans un espace de grande dimension [21].

Ces effets s'avèrent particulièrement néfastes pour les performances de prédiction lorsque les données sont de grande dimensions et sont utilisées par exemple en entrée d'un classificateur.

Certaines de ces méthodes non-linéaires reposent sur l'estimation des plus petites valeurs propres et des vecteurs propre associés. Malheureusement, dans la plupart des cas, leur estimation numérique et soit particulièrement instable, soit coûteuse en temps de calcul. Leur mise en œuvre est donc généralement délicate.

Enfin, plusieurs méthodes non-linéaires reposent sur la résolution d'un problème d'optimisation non-convexe (auto-encodeurs, SNE, LLC, Manifold Charting). Elles sont donc sensibles aux conditions d'initialisation et peuvent atteindre des minima locaux non-pertinents.

A l'heure actuelle et malgré la simplicité de sa formulation, l'Analyse en Composantes Principales reste donc un des meilleurs compromis pour les problèmes de grande taille (grande dimension et grand nombre d'échantillons). Dans ce chapitre, nous nous concentrerons donc sur

cette méthode linéaire. Comme nous le détaillons par la suite, celle-ci se prête particulièrement bien à la décentralisation et l'asynchronisme, qui réduisent d'ailleurs significativement son coût de calcul et son emprunte mémoire.

3.2 Réduction de dimension distribuée

Rappelons que la PCA nécessite le calcul puis la décomposition de la matrice de covariance $\mathbf{C}$ pour trouver ses q vecteurs propres dominants. Lorsque la dimension D de l'espace d'entrée est très grande, cette matrice de covariance ne peut plus être stockée en mémoire centrale. Par exemple, si l'ensemble d'entraînement contient plusieurs millions d'échantillons de plusieurs millions de dimensions, $\mathbf{X}$ et $\mathbf{C}$ occupent toutes deux plusieurs Tera-octets. De plus, la complexité en $O(D^3)$ de la décomposition en vecteur-valeurs propres induit un coût de calcul prohibitif de l'ordre de l'exa-flop. Ces limitations ont entraîné l'apparition de nombreux algorithmes de PCA distribuée.

Dans les deux scénarios ED (échantillons distribués) et CD (coordonnées distribuées) décrits précédemment, l'objectif de la PCA distribuée est souvent double :

(i) On souhaite d'abord produire en chaque nœud une représentation compressée des échantillons qu'il héberge.

(i) Par la suite, on souhaite également que chaque nœud puisse encoder de futurs échantillons de manière autonome. Le projeté obtenu devra être identique quel que soit le nœud utilisé.

Pour remplir ce double objectif, chaque nœud doit donc obtenir localement la même base globalement optimale, autrement dit la solution $\mathbf{U}^\star$ de la PCA calculée sur l'ensemble des échantillons du réseau $\mathbf{X}$. Parce que dans un environnement distribué le calcul de μ et $\mathbf{C}$ nécessite la matrice complète des échantillons $\mathbf{X}$ qui est localement inconnue, tout algorithme de PCA distribuée doit comporter une composante d'optimisation locale aux nœuds et une composante des communication entre nœuds.

Nous présentons ici quelques représentants des différentes approches utilisées dans la littérature. Toutes les méthodes décrites dans cette section ne satisfont pas nécessairement les deux objectifs, notamment celui de pouvoir projeter de futurs échantillons. Nous introduisons tout d'abord les méthodes qui utilise une information *a priori* sur les données d'entrée. Puis nous comparons diverses méthodes distribuées et les confrontons à nos contraintes de décentralisation et d'asynchronisme. Enfin, nous discutons les avantages et inconvénients de méthodes basées sur une agrégation de modèles vis-à-vis des méthodes par optimisation itérative.

3.2.1 Information a priori sur les échantillons

Les méthodes de PCA distribuée existantes peuvent être distinguées par le niveau d'information à priori qu'elles supposent sur leurs données d'entrée. En effet, la matrice des échantillons $\mathbf{X}$ peut véhiculer des observations locales aux nœuds indépendamment de la topologie qui les lie, ou a contrario intégrer des propriétés du réseau lui-même.

Dans ce second cas, un objet d'intérêt typique est la matrice d'adjacence du graphe pondéré associé au réseau, dont les entrées correspondent à une certaine relation scalaire entre nœuds. Par exemple, lorsque ces entrées représente des estimées de distance géographique entre nœuds

voisins, leur position réelle absolue peut être retrouvée en calculant les trois composantes principales au moyen d'une PCA distribuée [146].

D'autres méthodes considèrent le cas ou la distribution des données hérite certaines propriété spécifiques de la structure du réseau, telles que des dépendances statistiques. Ce cas se présente par exemple lorsque les données proviennent des flux d'informations à travers le réseau, introduisant ainsi des corrélations statistiques des échantillons provenant de nœuds amonts vers les nœuds avals tout en générant une indépendance mutuelle entre branches de propagation distinctes. En modélisant ce type de dépendances statistiques par des Modèles Graphiques, soit non-orientés (*e.g.*, Decomposable Gaussian Graphical Models [251]) ou orientés (*e.g.*, Directed Gaussian Acyclic Graphical Models [175]), on peut utiliser le caractère épars (sparsity) induit sur la matrice de concentration (*i.e.*, matrice inverse de la covariance) via sa décomposition de Cholesky pour estimer le sous-espace principal avec un coût de communication réduit.

A contrario, dans ce chapitre nous nous plaçons dans le premier cas, où la topologie du réseau ne joue aucun rôle sur la distribution des données ou sur la solution, dans la mesure où l'information est uniquement portée par les nœuds, et non par les connections. Néanmoins, toute information relative aux connections peut naturellement être vue comme une observation relative à l'une de ses extrémités, autorisant ainsi les méthodes focalisées sur les données relatives aux nœuds à intégrer des données relatives aux liens.

3.2.2 Décentralisation et asynchronisme

Un second critère de classification des approches existantes sépare celles-ci qui répondent aux critères de décentralisation et d'asynchronisme, tels que définis dans le chapitre 1.

Dans [190], les statistiques suffisantes $\mathbf{X}_i\mathbf{1}$ et $\mathbf{X}_i\mathbf{X}_i^\top$ sont calculées localement en chaque nœud i et transmises à un nœud maître qui en calcule une Décomposition en Valeurs Singulières (*Singular Value Decomposition*, SVD). Dans un second temps, le nœud maître diffuse alors les q premiers vecteurs propres obtenus à tous les autres nœuds pour obtenir la solution de la PCA en chaque nœud. Cette méthode suppose que le nœud maître puisse supporter le coût de calcul en $O(D^3)$ induit par la SVD et que les matrices de covariance de dimensions $D \times D$ puissent être échangées sur le réseau, hypothèses irréalistes quand D est grand.

Dans [24], un algorithme de PCA distribuée est proposé pour les scénarios CD. En échangeant des matrices de dimensions seulement $q \times q$, les nœuds maximisent itérativement la variance retenue par la base réduite. Bien que ce processus soit décentralisé, les nœuds doivent mettre à jour leurs estimées de manière synchrone, ce qui contredit (**C2**). Une conséquence notable de ce synchronisme est que les performances globales du système sont limitées à celles du nœud le plus lent.

Une méthode de la puissance itérée asynchrone et décentralisée est proposée dans [150] au moyen de matrices de communications aléatoires éparses et un protocole de moyennage Sum-Weight [145]. La sparsité des matrices de communication réduit significativement les coûts de communication. Cependant, la formulation originale de cet algorithme formulation ne fournit que la première composante principale. Des passes successives seraient nécessaires pour obtenir les composantes suivantes, rendant ainsi la méthode générale globalement synchrone. Cette approche est donc incompatible avec (**C2**).

3.2.3 Agrégation de modèles *versus* optimisation itérative

Une propriété importante de certaines méthodes est de ne requérir qu'une seule passe sur le jeu de données d'entraînement. C'est le cas notamment de [190] et [44], contrairement à [146, 150] qui nécessitent de multiples parcours des $\mathbf{X}_i$ à cause de leur schémas d'optimisation de nature itérative. Les approches en ligne (*online*) considèrent souvent un accès seulement temporaire aux données [24], leur permettant ainsi de traiter les problèmes "en flux" (*streaming data*) sous certaines hypothèses d'ergodicité de de stationnarité à court-terme.

Dans [44], les auteurs proposent un opérateur d'agrégation de Mélanges de PCA probabilistes (Mixtures of Probabilistic PCA ,MPPCA, [235]) qui maximise la vraisemblance du modèle agrégé sans recourir aux données originales qui ont servi à l'entraînement des modèles fournis en entrée. Ainsi, de multiples modèles peuvent être entraînés dans un premier temps, puis agrégés dans un second temps. Utilisés en conjonction avec le paradigme d'optimisation Gossip proposé dans [188], on obtient alors simplement un algorithme de PCA décentralisé. Cependant, comme développé dans [187], l'agrégation de tels modèles donne de bonnes performances seulement si l'opérateur de fusion est couplé à un mécanisme de validation croisée des modèles produits, étape qui apparaît non-triviale à réaliser de manière asynchrone.

Dans [168], les auteurs calculent tout d'abord des statistiques suffisantes, les agrègent dans un second temps au moyen d'un algorithme de consensus distribué, avant d'effectuer une décomposition en valeurs-vecteurs propres en chaque noeud de ces statistiques agrégées. Cela suppose d'une part l'échange d'estimées matricielles de taille $D \times D$ et d'autre part un coût de calcul de la décomposition en $\mathcal{O}D^3$ en chaque nœud, ce qui est de nouveau inenvisageable en grande dimension. Par ailleurs, il est regrettable que malgré l'identité des statistiques une fois agrégées et donc l'égalité de leurs valeur-vecteurs propres en tout nœud, chaque nœud doive effectuer de son côté la décomposition de la même matrice.

L'approche que nous proposons dans ce chapitre est similaire à [168],mais résout les limitations en matière de coûts de communication et de calcul en calculant la décomposition *avant* l'agrégation distribuée. Par ailleurs, en dépit d'objectifs quasi-identiques, les méthodes sus-citées ne traitent qu'un seul des deux scénarios ED et CD. Dans ce chapitre, nous montrons qu'un même algorithme, AGPCA, peut être formulé à la fois dans le cas d'échantillons distribués (ED) et dans le cas de coordonnées distribuées (CD), toujours avec un coût de calcul et de communications réduit.

3.3 Late PCA

Dans cette section, nous introduisons une approche naïve nommée *Late PCA* pour résoudre le problème de la PCA distribuée.

Late PCA s'appuie sur les deux intuitions suivantes :

- La PCA distribuée peut être formulée comme un problème de calcul de moyenne entre des matrices de covariance suivie de décompositions locales en couples valeur-vecteur propres.
- Le calcul de cette moyenne distribuée peut être réalisé de manière asynchrone et décentralisée par un protocole de consensus de type *Sum-Weight* tel que décrit au chapitre 1.

Dans le même esprit que [168], chaque nœud calcule des statistiques locales suffisantes,

moyenne itérativement ces statistiques avec ses voisins de manière décentralisée et asynchrone, puis décompose la matrice locale obtenue pour en extraire les q premiers vecteurs propres.

Les limitations de Late PCA mises en évidence dans cette section sont levées dans la section 4 par la formulation de l'algorithme AGPCA.

La PCA distribuée comme un problème d'estimation de moyenne

D'après (3.4), la matrice de corrélation $\mathbf{X}\mathbf{X}^\top$ et la somme des échantillons $\mathbf{X}\mathbf{1}$ sont des statistiques suffisante pour calculer la solution de la PCA. Comme observé dans [168, 190], ces statistiques peuvent être obtenues en calculant localement des statistiques partielles $\mathbf{X}_i\mathbf{1}$ and $\mathbf{X}_i\mathbf{X}_i^\top$ en chaque nœud i et en sommant sur tous les noeuds. Par conséquent, la moyenne globale de $\mathbf{X}$ et sa matrice de covariance peuvent être calculées comme une moyenne distribuée des moyennes partielles $\mu_i = \frac{1}{n_i}\mathbf{X}_i\mathbf{1}$ et des matrices de corrélation $\mathbf{B}_i = \mathbf{X}_i\mathbf{X}_i^\top$ localement calculables :

$$\mu = \frac{1}{n}\mathbf{X}\mathbf{1} = \frac{1}{n}\sum_i^N \mathbf{X}_i\mathbf{1} = \frac{1}{\sum_i n_i}\sum_i^N n_i\mu_i \tag{3.22}$$

$$\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^\top - \mu\mu^\top = \frac{1}{n}\sum_i^N \mathbf{X}_i\mathbf{X}_i^\top - \mu\mu^\top = \frac{1}{\sum_i n_i}\sum_i^N \mathbf{B}_i - \mu\mu^\top \tag{3.23}$$

Si l'on fournit aux nœuds la capacité d'estimer localement ces moyennes, chacun peut donc obtenir la matrice de covariance $\mathbf{C}$ de l'ensemble des échantillons du réseau. La décomposition de cette matrice $\mathbf{C}$ en chaque nœud nous donne finalement la solution de la PCA globale $\mathbf{U}^\star$.

Le fonctionnement de Late PCA est détaillé par l'Algorithme 5. Remarquons que $\mathbf{X}_i\mathbf{1}$ et $\mathbf{X}_i\mathbf{X}_i^\top$ peuvent être calculées simultanément via une unique passe sur les données $\mathbf{X}_i$. Notons également qu'une fois que les échantillons ont été intégrés à ces statistiques suffisantes, ils peuvent être supprimés sans impact sur les opérations suivantes. Cette approche et donc parfaitement adaptée aux cas où les données parviennent en flux (*streaming data*) et ne sont pas stockée de manière persistante.

Protocole de moyennage Gossip asynchrone

En tant que moyennes pondérées, μ et $\mathbf{C}$ peuvent être estimées grâce au protocole Gossip asynchrone décrit dans le chapitre 1, en définissant en chaque nœud des estimées locales $\mathbf{a}_i(t), \mathbf{B}_i(t)$ et des poids $w_i(t)$ telles que :

$$\mathbf{a}_i(0) = \mathbf{X}_i\mathbf{1} \qquad \mathbf{B}_i(0) = \mathbf{X}_i\mathbf{X}_i^\top \qquad w_i(0) = n_i \tag{3.24}$$

En appliquant le protocole asynchrone de calcul de moyennes AGAVG présenté dans le chapitre 1 à $\mathbf{a}_i(t)$,$\mathbf{B}_i(t)$ et $w_i(t)$, on obtient une estimée locale de la matrice de covariance $\mathbf{C}_i(t)$ comme suit :

$$\mathbf{C}_i(t) = \frac{\mathbf{B}_i(t)}{w_i(t)} - \frac{\mathbf{a}_i(t)\mathbf{a}_i(t)^\top}{w_i(t)^2} \tag{3.25}$$

Notons que les estimées initiales $\mathbf{C}_i(0)$ correspondent aux matrices de covariance des échantillons locaux $\mathbf{X}_i$. La limite établie par le Théorème 7 montre que chaque $\frac{\mathbf{a}_i(t)}{w_i(t)}$ tend vers la

Algorithme 5 Late PCA (en chaque noeud i)

Entrées : Jeu d'échantillons $\mathbf{X} \in \mathbb{R}^{n_i \times D}$
Paramètres : q : nombre de composantes principales désirées

∘ **Procédure émission**

$w \leftarrow n_i$; $\mathbf{a} \leftarrow \mathbf{X}^\top \mathbf{1}$; $\mathbf{C} \leftarrow \mathbf{X}\mathbf{X}^\top$
Boucle
  $\mathbf{a} \leftarrow \mathbf{a}/2$; $\mathbf{C} \leftarrow \mathbf{C}/2$; $w \leftarrow w/2$
  $j \leftarrow$ nœud voisin aléatoire
  Envoyer $(\mathbf{a}, \mathbf{C}, w)$ à j
Fin Boucle
$\mathbf{\Sigma} \leftarrow \mathbf{C}/w - \mathbf{a}\mathbf{a}^\top/w^2$
$(\mathbf{U}, \mathbf{L}) \leftarrow \text{eig}_q(\mathbf{\Sigma})$

∘ **Procédure réception**

Boucle
  Attendre réception d'un message $(\mathbf{a}', \mathbf{C}', w')$
  $\mathbf{a} \leftarrow \mathbf{a} + \mathbf{a}'$; $\mathbf{C} \leftarrow \mathbf{C} + \mathbf{C}'$; $w \leftarrow w + w'$
Fin Boucle

moyenne globale μ et que chaque $\mathbf{C}_i$ tend vers la matrice de covariance globale $\mathbf{C}$:

$$\forall i, \begin{cases} \lim\limits_{t\to\infty} \dfrac{\mathbf{a}_i(t)}{w_i(t)} = \dfrac{\sum_i \mathbf{X}_i^\top \mathbf{1}}{\sum_i n_i} = \dfrac{\sum_i n_i \mu_i}{\sum_i n_i} = \mu \\ \lim\limits_{t\to\infty} \mathbf{C}_i(t) = \dfrac{\sum_i \mathbf{B}_i(0)}{\sum_i w_i(0)} - \mu\mu^\top = \dfrac{1}{n}\sum\limits_i \mathbf{X}_i\mathbf{X}_i^\top - \mu\mu^\top = \mathbf{C} \end{cases} \tag{3.26}$$

Dès que chaque nœud a obtenu une estimée suffisamment précise pour $\mathbf{C}$, le résultat final de la PCA peut être localement calculé en tout nœud i via la décomposition en valeurs-vecteurs propres de $\mathbf{C}_i$.

Cette stratégie souffre d'un défaut majeur : la mise à jour des estimées $\mathbf{B}_i$ par (1.54) requiert la transmission de matrices $D \times D$, ce qui est inacceptable dans un contexte de grande dimension et de réseau à faible débit. Dans [93, 95], nous avons ainsi proposé de réduire les matrices $\mathbf{B}_i$ par le biais d'une PCA locale *avant* leur transmission sur le réseau. L'algorithme résultant, nommé AGPCA est décrit dans la section suivante.

3.4 AGPCA pour échantillons distribués (ED)

Dans cette section, nous introduisons l'algorithme AGPCA que nous avons proposé dans [93, 95], et qui répond aux deux scénarios ED et CD considérés. Cette section détaille la formulation d'AGPCA pour les scénarios ED. Nous présentons son extension aux scénarios CD dans la section suivante.

Dans un scénario ED (échantillons distribués), chaque nœud héberge une matrice d'échantillons locaux $\mathbf{X}_i \in \mathbb{R}^{D\times n_i}$ composée de n_i observations $[\mathbf{x}_1, \ldots, \mathbf{x}_{n_i}]$ dans $\mathbb{R}^D$. Dans un tel scénario, AGPCA fournit à tous les nœuds une base orthonormale identique $\mathbf{U}^\star \in \mathbb{R}^{D\times q}$ qui engendre le sous-espace q-principal de la matrice de covariance complète $\mathbf{C}$ telle que définie

par (3.3). Chaque nœud i peut ainsi projeter ses propres données locales $\mathbf{X}_i$ pour obtenir une représentation compressée $\mathbf{Y}_i = \mathbf{U}^{\star\top}\mathbf{X}_i$, où $\mathbf{Y}_i \in \mathbb{R}^{q\times n_i}$. Il est important de noter que toute observation future $\mathbf{x}_{new}$ peut être compressée par le nœud qui la reçoit de la même façon : $\mathbf{y}_{new} = \mathbf{U}^{\star\top}\mathbf{x}_{new}$, même lorsque $\mathbf{X}_i$ a été perdu ou n'était accessible que durant la phase d'entraînement (*e.g.*, cas de données en flux).

L'intuition derrière AGPCA s'appuie sur 2 principes additionnels à ceux de Late PCA :

- En définissant des opérateurs de multiplication par un scalaire et de sommes sur les bases orthonormales qui ne repose pas sur une reconstruction explicite des matrices initiales, on peut étendre les protocoles de moyennage *Sum-Weight* pour n'échanger que les q vecteurs propres dominants au lieu des matrices de covariances complètes. Les problèmes de coût de communications sont ainsi résolus.
- En utilisant la dualité entre la décomposition de la matrice de covariance et celle de la matrice de Gram via la Décomposition en Valeurs Singulières (SVD), on peut obtenir les vecteurs propres dominants des matrices de covariance de taille $(D \times D)$, sans jamais les calculer explicitement, mais en décomposant plutôt les matrices de Gram locales de taille $(n_i \times n_i)$. Le coût de calcul et de stockage s'en trouve alors significativement réduit.

Algorithme 6 AGPCA-DS : Asynchronous Gossip PCA (échantillons distribués) (en chaque nœud i)

Entrées : Jeu d'échantillons $\mathbf{X} \in \mathbb{R}^{n_i\times D}$
Paramètres : q : nombre de composantes principales désirées

∘ **Procédure émission**

$\mathbf{a} \leftarrow \mathbf{X1}$; $\mathbf{G} \leftarrow \mathbf{X}^\top\mathbf{X}$; $w \leftarrow n_i$
$(\mathbf{V}, \mathbf{L}) \leftarrow \text{eig}_q(\mathbf{G})$
$\mathbf{U} \leftarrow \mathbf{XVL}^{-\frac{1}{2}}$
Boucle
  $\mathbf{a} \leftarrow \mathbf{a}/2$; $\mathbf{L} \leftarrow \mathbf{L}/2$; $w \leftarrow w/2$
  $j \leftarrow$ nœud voisin aléatoire
  Envoyer $(\mathbf{a}, \mathbf{U}, \mathbf{L}, w)$ à j
Fin Boucle

∘ **Procédure réception**

Boucle
  Attendre réception d'un message $(\mathbf{a}', \mathbf{U}', \mathbf{L}', w')$
  $\mathbf{a} \leftarrow \mathbf{a} + \mathbf{a}'$; $w \leftarrow w + w'$
  $\mathbf{Q}_0 \leftarrow \mathbf{U}$
  Pour t **de** 1 **à** max_t **faire**
    $(\mathbf{Q}_{t+1}, \mathbf{R}_{t+1}) \leftarrow \text{QR}(\mathbf{UL}(\mathbf{U}^\top\mathbf{Q}_t) + \mathbf{U}'\mathbf{L}'(\mathbf{U}'^\top\mathbf{Q}_t))$
  Fin Pour
  $\mathbf{U} \leftarrow \mathbf{Q}_{max\_t}$; $\mathbf{L} \leftarrow \text{diag}(\mathbf{R}_{max\_t})$
Fin Boucle

Grâce à ces deux observations, la PCA en très grande dimension avec un très grand nombre d'exemples d'apprentissage devient traitable de manière décentralisée et asynchrone. L'algo-

rithme AGPCA que nous proposons pour les scénarios ED est constitué de deux procédures exécutées en parallèle en chacun des nœuds (Algorithme 6). Les itérations de la procédure de réception ne sont déclenchées qu'à réception d'un message (ligne 2) tandis que la procédure d'émission envoie des messages régulièrement à des nœuds voisins indépendamment des messages reçus (lignes 5-7). A la différence d'AGKM et AGEM décrits dans le chapitre 2, aucun paramètre critique n'intervient et l'algorithme converge dans tous les cas, quelles que soient la distribution des données ou les caractéristiques du réseau. Précisons que lors de l'implémentation d'AGPCA, on doit prendre soin que les lignes 5-7 de la procédure d'emission et les lignes 3-8 de la procédure de réception soient mutuellement exclusives (*i.e.*, synchronisées par l'usage de mutex par exemple), pour prévenir tout conflit de mise à jour des variables partagées $(\mathbf{a}_i, \mathbf{U}_i, \mathbf{L}_i, w_i)$. Toutefois, cette contrainte ne contredit pas l'hypothèse d'asynchronisme, puisqu'il n'y a clairement aucune synchronisation *entre* nœuds.

Protocole Gossip dans le domaine compressé : schéma Early PCA

Late PCA révèle deux inconvénients principaux, tous deux provenant du fait qu'un tel schéma ne bénéficie pas de la déficience en rang des matrices de covariance locales, typiquement caractérisées par $n_i \ll D$ du fait de l'éparpillement des échantillons sur de multiples nœuds.

Premièrement, la quantité d'information échangée sur le réseau est homogène à la taille des statistiques d'entrée ($D \times D$), et non à celle du résultat désiré ($D \times q$). Par conséquent, Late PCA échange vainement de l'information qui sera supprimée par l'opération de réduction de dimension qui survient *après* la phase d'estimation de moyenne distribuée.

Deuxièmement, la nature distribuée du processus n'offre aucun avantage computationnel en grande dimension, puisque tous les nœuds doivent réaliser localement la décomposition d'une matrice $D \times D$ avec un coût typique en $\mathcal{O}(D^3)$. Par conséquent, si la décomposition de $\mathbf{C}$ est à première vue le principal facteur limitant que l'on souhaite lever par la distribution du calcul, Late PCA n'apporte ici aucun gain notable et présente donc peu d'intérêt pratique.

AGPCA adopte un schéma que nous nommons *Early-PCA*, dans la mesure où il cherche à déplacer l'étape de réduction de dimensions *avant* l'étape d'estimation de moyenne distribuée, de sorte à tirer avantage de la déficience en rang des covariances locales $\mathbf{C}_i$. Pour ce faire, nous reformulons les règles de mise à jour d'AGAVG pour qu'elles ne dépendent plus que des couple valeur-vecteur propres au lieu des matrices complètes $\mathbf{B}_i$. Cela nous permet de calculer implicitement la moyenne de matrices arbitrairement grandes dès lors que leur rang est faible, sans jamais recourir à leur reconstruction explicite mais uniquement à leur forme factorisée.

Rappelons que la règle de mise à jour à l'émission (procédure emission de l'Algorithme 2)) est une simple multiplication par un scalaire. En supposant que le nœud émetteur i possède la forme décomposée $\mathbf{U}_i\mathbf{L}_i\mathbf{U}_i^\top$ de son estimée $\mathbf{B}_i$, la règle de mise à jour devient simplement :

$$\mathbf{U}_i(t+1) = \mathbf{U}_i(t) \quad \mathbf{L}_i(t+1) = \frac{1}{2}\mathbf{L}_i(t) \tag{3.27}$$

Cette formulation conduit à la procédure d'émission d'AGPCA que l'on retrouve dans l'Algorithme 6.

L'adaptation de la mise à jour du récepteur (procédure réception de l'Algorithme 2) est légèrement moins triviale puisque nous devons calculer la décomposition d'une somme de deux matrices sachant seulement leur propre décomposition. Nous proposons d'utiliser la méthode des Iterations Orthogonales (OI), dans un esprit proche de [146], pour aligner une base orthonormale

initialement quelconque sur le sous-espace q-principal engendré par la somme des matrices d'entréee, via des décompositions QR itérées. Supposons que le nœud i reçoit un message du nœud j, $\mathbf{B}_i(t)$ et $\mathbf{B}_j(t)$ étant représentées par le formes factorisées, respectivement $\mathbf{U}_i(t)\mathbf{L}_i(t)\mathbf{U}_i(t)^\top$ et $\mathbf{U}_j(t)\mathbf{L}_j(t)\mathbf{U}_j(t)^\top$. En partant d'une base $D \times q$ quelconque $\mathbf{Q}_0$, et en notant $QR(\cdot)$ l'opération de décomposition QR économique (*thin* en littérature anglophone), on calcule itérativement

$$\mathbf{Q}_{\tau+1}\mathbf{R}_{\tau+1} = QR((\mathbf{B}_i + \mathbf{B}_j)\mathbf{Q}_\tau) \tag{3.28}$$

$$= QR(\mathbf{U}_i\mathbf{L}_i(\mathbf{U}_i^\top\mathbf{Q}_\tau) + \mathbf{U}_j\mathbf{L}_j(\mathbf{U}_j^\top\mathbf{Q}_\tau)) \tag{3.29}$$

Par ce processus, $\mathbf{Q}_\tau$ devient progressivement un base orthonormée du sous-espace q-principal de $\mathbf{B}_i(t) + \mathbf{B}_j(t)$, avec les valeurs propres associées sur la diagonale de $\mathbf{R}_\tau$. Ainsi, $\mathbf{Q}_\infty$ donne $\mathbf{U}_i(t+1)$ et la diagonale de $\mathbf{R}_\infty$, notée $diag(\mathbf{R}_\infty)$, donne $\mathbf{L}_i(t+1)$.

La position des parenthèses dans l'équation (3.29) a une grande importance. En effet, observons qu'au lieu de développer $\mathbf{U}_s\mathbf{L}_s\mathbf{U}_s^\top$ et $\mathbf{U}_r\mathbf{L}_r\mathbf{U}_r^\top$, nous multiplions d'abord $\mathbf{U}_s^\top$ et $\mathbf{U}_r^\top$ par $\mathbf{Q}_\tau$ pour obtenir une matrice $q \times q$. De ce fait, nous ne stockons jamais de matrices $D \times D$, la décomposition QR étant alors effectuée sur une matrice $D \times q$.

Notons que dans AGPCA, nous initialisons $\mathbf{Q}_0$ à $\mathbf{U}_i(t)$. Ainsi, lorsque $\mathbf{B}_i(t) = \mathbf{B}_j(t)$, $\mathbf{Q}_0$ forme déjà une base orthonormale de $\mathbf{B}_i(t)+\mathbf{B}_j(t) = 2\mathbf{B}_i(t)$. On espère ainsi qu'à mesure de la convergence vers un consensus, un plus petit nombre d'Itérations Orthogonales seront requises pour obtenir une précision donnée sur la base estimée. Toutefois, par soucis de simplicité, nous ne considérerons ici que le nombre d'itérations orthogonales est fixé a priori à max_t.

Rappelons que nous cherchons les q vecteurs propres dominants $\mathbf{U}^\star$ de $\mathbf{C}_i$, qui peut simplement s'écrire comme une somme de deux matrices factorisées :

$$\mathbf{C}_i = \frac{1}{w_i(t)}(\mathbf{U}_i\mathbf{L}_i)(\mathbf{U}_i^\top) - \frac{1}{w_i(t)^2}(\mathbf{a}_i(t))(\mathbf{a}_i(t)^\top). \tag{3.30}$$

Par conséquent, nous pouvons utiliser la même stratégie d'Itérations Orthogonales (3.28) que pour $\mathbf{B}_i + \mathbf{B}_j$ pour finalement obtenir $\mathbf{U}^\star$ à un coût de calcul réduit et sans coût mémoire additionnel (les parenthèses sont également ajoutées ici pour mettre en évidence que la reconstruction explicite des deux termes n'est pas nécessaire). Cette stratégie conduit ainsi à la procédure de réception d'AGPCA, exécutée concurremment en tout nœud.

AGPCA n'échange donc sur le réseau que des matrices $D \times q$ au lieu de $D \times D$. Dans la mesure ou l'on choisit typiquement $q \ll D$, le gain en coûts de communication est important, tout en garantissant que le résultat reste équivalent à Late PCA décrit précédemment et donc à une PCA centralisée exécutée sur l'ensemble des données, à condition que le rang de $\mathbf{X}\mathbf{X}^\top$ soit inférieur à q. Cette propriété est formellement établie dans la section 6.

D'un point de vue computationnel, dès lors que l'on s'appuie sur des décompositions QR économique dans (13), le coût de calcul des max_t Itérations Orthogonales est de l'ordre de $\mathcal{O}(max\_t.Dq^2)$ flops. Ainsi, le coût total d'AGPCA en un nœud quelconque est en $\mathcal{O}(max\_t.MDq^2)$ où M désigne le nombre total de message reçus par ce nœud.

3.4.1 Relation duale entre les décompositions de la matrices de covariance et de la matrice de Gram

Le schéma d'optimisation Early PCA nous dispense d'échanger des matrices $D \times D$ sur le réseau. Mais subsistent les problèmes liés au stockage locale de ces matrices. Lorsque D est grand,

le stockage des matrices de covariance locales complètes n'est pas envisageable. Heureusement, nous pouvons éviter le calcul explicite et le stockage des matrices $\mathbf{B}_i(0)$ calculées localement via (3.24), en rappelant que nous en cherchons seulement le sous-espace q-principal.

En effet, les décompositions $\mathbf{B}_i = \mathbf{U}_i\mathbf{L}_i\mathbf{U}_i^\top$ peuvent être obtenue à partir de celle des matrices de Gram locales $\mathbf{X}_i^\top\mathbf{X}_i = \mathbf{V}_i\mathbf{\Lambda}_i\mathbf{V}_i^\top$, de dimensions $n_i \times n_i$, à travers leur relation bien connue avec la Décomposition en Valeurs Singulières (SVD) de $\mathbf{X}$:

$$\mathbf{B}_i^2 = \mathbf{X}_i\mathbf{X}_i^\top\mathbf{X}_i\mathbf{X}_i^\top = \mathbf{X}_i\mathbf{V}_i\mathbf{\Lambda}_i\mathbf{V}_i^\top\mathbf{X}_i^\top = (\mathbf{X}_i\mathbf{V}_i\mathbf{\Lambda}_i^{-\frac{1}{2}})\mathbf{\Lambda}_i^2(\mathbf{X}_i\mathbf{V}_i\mathbf{\Lambda}_i^{-\frac{1}{2}})^\top$$
$$\text{and} \qquad (\mathbf{X}_i\mathbf{V}_i\mathbf{\Lambda}_i^{-\frac{1}{2}})^\top(\mathbf{X}_i\mathbf{V}_i\mathbf{\Lambda}_i^{-\frac{1}{2}}) = \mathbf{I}$$

Par conséquent, $\mathbf{U}_i$ et $\mathbf{L}_i$ peuvent être obtenues à partir des matrices $\mathbf{V}_i$ de taille $n_i \times q$ des q vecteurs propres dominants de $\mathbf{X}_i^T\mathbf{X}_i$, les valeurs propres associées et la matrice d'échantillons locale $\mathbf{X}_i$:

$$\mathbf{U}_i = \mathbf{X}_i\mathbf{V}_i\mathbf{\Lambda}_i^{-\frac{1}{2}} \qquad \text{and} \qquad \mathbf{L}_i = \mathbf{\Lambda}_i \tag{3.31}$$

Ainsi, on calcule, stocke et factorise $\mathbf{X}_i^\top\mathbf{X}_i$ (de taille $n_i \times n_i$) au lieu de $\mathbf{X}_i\mathbf{X}_i^\top$ (de taille $D \times D$) pour obtenir $\mathbf{U}_i$ et $\mathbf{L}_i$ sans aucun coût de stockage supplémentaire.

D'un point de vue computationnel, le coût de calcul total devient $\mathcal{O}(n_i^3 + max\_t.MDq^2)$ flops en chaque nœud. En supposant que les données sont équitablement dispersées sur le réseau, on a $n_i^3 \simeq n^3/N^3 \leq D^3$ pour tout N suffisamment grand (hypothèse de large réseau). Comparé au coût en $\mathcal{O}(D^3)$ de la décomposition requise en chaque nœud par l'approche Late PCA, AGPCA est donc particulièrement avantageux sur le plan computationnel lorsque $max\_t.Mq^2 \ll D$. Par hypothèse, on peut écrire $q^2 \leq D$ puisque $q \ll D$. Grâce à la convergence exponentielle d'AGPCA démontrée dans la section 6, on peut notamment choisir $M = \mathcal{O}(\ln N)$ si le graphe du réseau est "bien-connecté" (épandeur, clique, etc, voir [18] pour plus de détails). Ainsi, on peut raisonnablement espérer un gain en temps de calcul vis-à-vis de Late PCA dès que $\ln N \ll D$, ce qui est typiquement le cas dans le contexte applicatif considéré. Finalement, les contraintes de communication étant notre soucis prioritaire, la propriété clé d'AGPCA réside dans la réduction de la taille des messages échangés de $\mathcal{O}(D^2)$ à $\mathcal{O}(Dq)$.

3.5 AGPCA pour coordonnées distribuées (CD)

Dans cette section, nous montrons qu'AGPCA peut également résoudre les problèmes de PCA distribuée pour les scénarios de type CD (Coordonnées Distribuées).

Dans un scénario CD, contrairement au cas ED, chaque nœud héberge un jeu de données local $\mathbf{Z}_i \in \mathbb{R}^{D_i \times n}$ composée de $D_i \ll D$ dimensions d'un seul et même ensemble global de n observations. Ce jeu d'échantillons global $\mathbf{Z}$ correspond donc à l'empilage des lignes des matrices $\mathbf{Z}_i$:

$$\mathbf{Z}^\top = [\mathbf{Z}_1^\top, \ldots, \mathbf{Z}_N^\top] \tag{3.32}$$

L'espace des observation est donc la somme directe des sous-espaces localement observés $\bigoplus_i \mathbb{R}^{D_i}$. Notons $\mu = \mathbf{Z}\mathbf{1}/n$ la moyenne globale des données et $\tilde{\mathbf{Z}} = \mathbf{Z} - \mu\mathbf{1}^\top$ la matrices des échantillons centrés.

Algorithme 7 AGPCA-DC : Asynchronous Gossip PCA (coordonnées distribuées) (en chaque noeud i)

Entrées : Jeu d'échantillons $\mathbf{X} \in \mathbb{R}^{n \times D_i}$
Paramètres : q : nombre de composantes principales désirées
max_t : nombre d'itérations orthogonales pour combiner deux bases

∘ **Procédure émission**

$\mathbf{C} \leftarrow \mathbf{X}\mathbf{X}^\top$; $w \leftarrow D_i$
$(\mathbf{U}, \mathbf{L}) \leftarrow \text{eig}_q(\mathbf{C})$
$\mathbf{V} \leftarrow \mathbf{X}^\top \mathbf{U}\mathbf{L}^{-\frac{1}{2}}$
Boucle
  $j \leftarrow$ nœud voisin aléatoire
  $\mathbf{V} \leftarrow \mathbf{V}/2$; $\mathbf{L} \leftarrow \mathbf{L}/2$; $w \leftarrow w/2$
  Envoyer $(\mathbf{V}, \mathbf{L}, w)$ à j
Fin Boucle

∘ **Procédure réception**

Boucle
  Attendre réception d'un message $(\mathbf{V}', \mathbf{L}', w')$
  $w \leftarrow w + w'$
  $\mathbf{Q}_0 \leftarrow \mathbf{V}$
  Pour t **de** 1 **à** max_t **faire**
    $(\mathbf{Q}_{t+1}, \mathbf{R}_{t+1}) \leftarrow \text{QR}(\mathbf{V}\mathbf{L}(\mathbf{V}^\top \mathbf{Q}_t) + \mathbf{V}'\mathbf{L}'(\mathbf{V}'^\top \mathbf{Q}_t))$
  Fin Pour
  $\mathbf{V} \leftarrow \mathbf{Q}_{max\_t}$; $\mathbf{L} \leftarrow \text{diag}(\mathbf{R}_{max\_t})$
Fin Boucle

Dans un scénario DC, il serait inutile de fournir à chaque nœud une base orthonormale $D \times q$, puisque chacun ne possède qu'une sous partie des coordonnées de ses échantillons et ne peut donc pas en réaliser la projection localement. Dans ce scénario, AGPCA fournit donc directement en chaque nœud une représentation compressée $\mathbf{Y} \equiv \mathbf{U}^{\star\top}\tilde{\mathbf{Z}}$ qui tient compte de *toutes* les dimensions de *toutes* les observations.

En nous appuyant sur les mêmes intuitions que dans la section précédente, nous tirons parti de la dualité entre la décomposition de la matrice de covariance et celle de la matrice de Gram. Rappelons que

$$\mathbf{C} = \frac{1}{n}\tilde{\mathbf{Z}}\tilde{\mathbf{Z}}^\top = \mathbf{U}^\star\mathbf{L}^\star\mathbf{U}^{\star\top} \tag{3.33}$$

En introduisant la SVD des échantillons centrés $\tilde{\mathbf{Z}} = \mathbf{U}\mathbf{L}^{\frac{1}{2}}\mathbf{V}^\top$, où $\tilde{\mathbf{Z}}\tilde{\mathbf{Z}}^\top = \mathbf{U}\mathbf{L}\mathbf{U}^\top$ et $\tilde{\mathbf{Z}}^\top\tilde{\mathbf{Z}} = \mathbf{V}\mathbf{L}\mathbf{V}^\top$, on a $\mathbf{U}^\star = \mathbf{U}$ et $\mathbf{L}^\star = \mathbf{L}/n$. Par conséquent,

$$\mathbf{Y} = \mathbf{U}^{\star\top}\tilde{\mathbf{Z}} = n\mathbf{L}^{\star\frac{1}{2}}\mathbf{V}^\top \tag{3.34}$$

Definissons $\mathbf{D} = \tilde{\mathbf{Z}}^\top\tilde{\mathbf{Z}}/D$.
Clairement sa décomposition en valeurs-vecteurs propres est $\mathbf{V}(\mathbf{L}/D)\mathbf{V}^\top = \mathbf{V}(n\mathbf{L}^\star/D)\mathbf{V}^\top$,

et

$$\begin{aligned}\mathbf{D} &= \frac{1}{D}(\mathbf{Z} - \frac{1}{n}\mathbf{Z}\mathbf{1}\mathbf{1}^\top)^\top(\mathbf{Z} - \frac{1}{n}\mathbf{Z}\mathbf{1}\mathbf{1}^\top) \\ &= \left(I - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right)\frac{\mathbf{Z}^\top\mathbf{Z}}{D}\left(I - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right) \end{aligned} \tag{3.35}$$

Observons que l'équation 3.35 ne fait intervenir que $\mathbf{Z}^\top\mathbf{Z}/D$. On remarque alors que $\mathbf{Z}^\top\mathbf{Z}/D = (\sum_i \mathbf{Z}_i^\top\mathbf{Z}_i)/(\sum_i D_i)$ n'être autre qu'une moyenne pondérée des matrices de Gram locales non centrées $\mathbf{Z}_i^\top\mathbf{Z}_i$, calculables en chaque nœud i. Cette moyenne peut être calculée via le même protocole que dans le cas ED, tel que défini par (3.27-3.29). A l'aide des facteurs $\mathbf{V}$ et $\mathbf{L}/D$ de $\mathbf{D}$ dans l'équation (3.34), on peut obtenir les représentations compressées à un facteur $\sqrt{D}$ près. Dans la plupart des applications, ce facteur d'échelle est sans impact. Néanmoins, dans les contextes applicatifs qui nécessitent une base orthonormale (et pas seulement orthogonale), le facteur d'échelle peut être retrouvé en moyennant les D_i locaux par un protocole Sum-Weight classique avec un poids initial de 1, sous réserve que la taille du réseau soit connue en chaque nœud.

Agrégation Early PCA pour les scénarios CD

Le schéma d'agrégation Early PCA décrit dans la section 4 s'applique également : nous avons seulement besoin de calculer et échanger les formes décomposées $\mathbf{Z}_i^\top\mathbf{Z}_i = \mathbf{V}_i\mathbf{L}_i\mathbf{V}_i^\top$, où $\mathbf{V}_i \in \mathbb{R}^{n\times q}$ et $\mathbf{L}_i \in \mathbb{R}^{q\times q}$ est diagonale. Les matrices de Gram locales de dimensions $n \times n$ étant bien plus volumineuses que les matrices de corrélation locales de taille $D_i \times D_i \ll n \times n$, nous utilisons la même propriété duale que dans la section 4 pour obtenir les décompositions des matrices de Gram à partir de celles des matrices de corrélation $\mathbf{Z}_i\mathbf{Z}_i^\top = \mathbf{U}_i\mathbf{L}_i\mathbf{U}_i^\top$, où $\mathbf{U}_i \in \mathbb{R}^{D\times q}$:

$$\mathbf{Z}_i = \mathbf{U}_i\mathbf{L}_i^{\frac{1}{2}}\mathbf{V}_i^\top \quad \Rightarrow \quad \mathbf{V}_i = \mathbf{Z}_i^\top\mathbf{U}_i\mathbf{L}_i^{-\frac{1}{2}}$$

Les formes factorisées $\mathbf{V}_i\mathbf{L}_i\mathbf{V}_i^\top$ initiales peuvent ainsi être obtenue à un coût de calcul et de stockage réduit. Les procédures d'émission et de réception résultantes, dont le fonctionnement général et l'ordonnancement est identique à l'Algorithme 6, sont détaillées par l'Algorithme 7.

Une fois que le protocole de moyennage Gossip a fourni à tous les nœuds une forme factorisée de $\mathbf{Z}^\top\mathbf{Z}/D$ avec une précision jugée suffisante, on peut appliquer localement une dernière passe d'Itérations Orthogonales à l'équation (3.35) pour obtenir la décomposition de $\mathbf{V}(\mathbf{L}^\star/D)\mathbf{V}^\top$ de $\mathbf{D}$. Chaque noeud calcule finalement la matrice des échantillons compressés par $\mathbf{Y} = n(\mathbf{L}^\star/D)^{\frac{1}{2}}\mathbf{V}^\top$.

Cette possibilité d'inversion des rôles entre matrices de covariance et matrices de Gram fait d'AGPCA une solution unifiée pour résoudre efficacement les problèmes de PCA distribuée aussi bien dans les scénarios ED que CD, avec des coûts de communication, de calcul et de stockage maîtrisés. Une conséquence particulièrement avantageuse est que les résultats théoriques de convergence valides dans un cas le sont naturellement dans l'autre.

3.5.1 Projection d'observations futures

Dans un scénario CD, la projection d'observations futures n'est pas aussi triviale que dans le cas ED, car chaque nouvelle observation génère de nouvelles entrées sur *tous* les nœuds. Tous les nœuds doivent donc participer à la projection de chaque observation.

Rappelons que la représentation compressée $\mathbf{y}$ d'une nouvelle observation $\mathbf{z}$ est calculée comme :

$$\mathbf{y} = n(\mathbf{L}^\star/D)^{-\frac{1}{2}}\mathbf{V}^\top \frac{\mathbf{Z}^\top \mathbf{z}}{D} \tag{3.36}$$

En observant que $\mathbf{Z}^\top \mathbf{z}/D = \sum_i \mathbf{Z}_i^\top \mathbf{z}_i/D$ est également une moyenne pondérée, on calcule donc $\mathbf{y}$ sans difficulté en appliquant le même protocole de moyennage Gossip à $\mathbf{Z}_i^\top \mathbf{z}_i$ avec pour poids initiaux les D_i associés, puis en combinant le résultat avec les facteurs obtenus précédemment $\mathbf{V}$ et $\mathbf{L}/D$.

3.6 Analyse Théorique

Dans cette section, nous démontrons formellement qu'AGPCA produit en chaque noeud une solution identique à celle d'une PCA centralisée classique, moyennant une hypothèse sur le rang de la matrice des échantillons d'entrée.

Nous ne détaillons que l'analyse pour les scénarios ED, dans la mesure où les même arguments se traduisent aisément aux cas CD en inversant les rôles des matrices de covariance et des matrices de Gram. En notant $\mathbf{X} \in \mathbb{R}^D$ la matrice des échantillons d'entrée input data, le résultat principal de cette section est donné par le théorème suivant :

Theorème 12. *Si $rang(\mathbf{X}) \leq q$, AGPCA produit en tout noeud i un résultat identique à une PCA réalisée sur $\mathbf{X}$:*

$$\forall i, \mathbf{U}_i^\top \mathbf{X}_i = \mathbf{U}^{\star\top} \mathbf{X}_i \tag{3.37}$$

Démonstration. La PCA centralisée correspond à la décomposition suivante :

$$\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^\top - \mu\mu^\top = \mathbf{U}^\star \mathbf{L}^\star \mathbf{U}^{\star\top} \tag{3.38}$$

Nous adoptons le schéma de preuve suivant : nous montrons tout d'abord que les décompositions locales $\mathbf{X}_i\mathbf{X}_i^\top = \mathbf{U}_i\mathbf{L}_i\mathbf{U}_i^\top$ reconstruisent parfaitement les données locales $\mathbf{X}_i$. Ensuite, nous montrons que le protocole d'agrégation Gossip des bases $\mathbf{U}_i$ autorise une reconstruction parfaite de $\sum_i \mathbf{X}_i\mathbf{X}_i^\top$ en tout noeud. La preuve est complétée en observant qu'AGPCA fournit exactement les q vecteurs propres dominants de $\sum_i \mathbf{X}_i\mathbf{X}_i^\top$.

Pour montrer que les factorisations locales n'induisent aucune perte d'information, nous faisons appel à l'hypothèse que $rang(\mathbf{X}) \leq q$. On a ainsi :

$$rang(\mathbf{X}) \leq q \Rightarrow rang(\mathbf{X}\mathbf{X}^T) \leq q \tag{3.39}$$

Soit $\mathcal{H}_q = span(\mathbf{X}) \subset \mathbb{R}^q$ l'espace engendré par $\mathbf{X}$. Puisqu'en tout noeud i, $\mathbf{X}_i$ est un sous-ensemble de $\mathbf{X}$, on a $span(\mathbf{X}_i) \subseteq \mathcal{H}_q$, et donc $rang(\mathbf{X}_i\mathbf{X}_i^\top) \leq q$. Il s'ensuit que les décompositions locales initiales tronquées aux q vecteurs propres dominants (les autres étant associés à des valeurs propres nulles) reconstruisent parfaitement les matrices de corrélation locales :

$$\forall i, \mathbf{U}_i\mathbf{L}_i\mathbf{U}_i^\top = \mathbf{X}_i\mathbf{X}_i^\top = \mathbf{B}_i \tag{3.40}$$

Considérons maintenant le protocole d'agrégation Gossip. Il nous faut prouver que les mises à jours (3.27) et 3.29) sont équivalentes au protocole AGAVG. C'est trivialement le cas pour $\mathbf{a}$ et w, qui sont explicitement mises à jour selon l'algorithme 2. Concernant la mise à jour de $\mathbf{B}_s$, la procédure d'émission (ligne 5 de l'Algorithme 6) est également équivalente à celle d'AGAVG :

$$\mathbf{B}_s(t+1) = \mathbf{U}_s(t)\frac{1}{2}\mathbf{L}_s(t)\mathbf{U}_s(t)^T = \frac{1}{2}\mathbf{B}_s(t).$$

Du point de vue de la réception, si $span(\mathbf{U}_s(t)) \subseteq \mathcal{H}_q$ et $span(\mathbf{U}_r(t)) \subseteq \mathcal{H}_q$, la méthode des Itérations Orthogonales utilisée pour calculer les q vecteurs propres dominants de la somme pondérée révèle deux propriétés intéressantes : La reconstruction est parfaite, c'est à dire :

$$\mathbf{U}_r(t+1)\mathbf{L}_r(t+1)\mathbf{U}_r(t+1) = \mathbf{U}_r(t)\mathbf{L}_r(t)\mathbf{U}_r(t)^\top + \frac{1}{2}\mathbf{U}_s(t)\mathbf{L}_s(t)\mathbf{U}_s(t)^\top, \tag{3.41}$$

et la base obtenue est contenue dans le même sous-espace, *i.e.*, $\mathbf{U}_r(t+1) \subseteq \mathcal{H}_q$. Ces propriétés viennent du fait que $span(\mathbf{U}_r \cup \mathbf{U}_s) \subseteq \mathcal{H}_q$, et donc

$$rang(\mathbf{U}_r(t)\mathbf{L}_r(t)\mathbf{U}_r(t)^\top + \mathbf{U}_s(t)\mathbf{L}_s(t)\mathbf{U}_s(t)^\top/2) \leq q.$$

En remarquant qu'initialement $\forall i, span(\mathbf{U}_i(0)) \subset \mathcal{H}_q$, on obtient par induction qu'à tout instant t

$$\begin{aligned}
\mathbf{B}_r(t+1) &= \mathbf{U}_r(t+1)\mathbf{L}_r(t+1)\mathbf{U}_r(t+1) \\
&= \mathbf{U}_r(t)\mathbf{L}_r(t)\mathbf{U}_r(t)^\top + \frac{1}{2}\mathbf{U}_s(t)\mathbf{L}_s(t)\mathbf{U}_s(t)^\top \\
&= \mathbf{B}_r(t) + \frac{1}{2}\mathbf{B}_s(t)
\end{aligned}$$

Autrement dit, l'équation (3.29) est équivalente à (1.84).

Rappelons que la convergence des équations (1.83) et (1.84) vers la moyenne globale du réseau a été montrée dans le chapitre 1 par le théorème 7. Nous exploitons ce résultat pour garantir que (3.27) et (3.29) conduisent tous les $\mathbf{C}_i$ vers $\mathbf{C}$. Introduisons les erreurs Euclidiennes E_i entre les covariances locales reconstruites et la covariance globale :

$$\begin{aligned}
\forall i, \quad E_i(t) &= \|\mathbf{C}_i(t) - \mathbf{C}\|_F^2 \\
&= \left\| \frac{1}{w_i(t)}\mathbf{B}_i(t) - \frac{1}{w_i(t)^2}\mathbf{a}_i(t)\mathbf{a}_i(t)^\top - \frac{1}{n}\mathbf{X}\mathbf{X}^\top + \mu\mu^\top \right\|_F^2 \\
&\leq \left\| \frac{1}{w_i(t)}\mathbf{B}_i(t) - \frac{1}{n}\mathbf{X}\mathbf{X}^\top \right\|_F^2 + \left\| \frac{\mathbf{a}_i(t)\mathbf{a}_i(t)^\top}{w_i(t)^2} - \mu\mu^\top \right\|_F^2
\end{aligned}$$

D'après (3.24) et (3-4), on a $n = \sum_i w_i(0)$, $\mathbf{X}\mathbf{X}^\top = \sum_i \mathbf{B}_i(0)$ and $\mu = (\sum_i \mathbf{a}_i(0))/(\sum_i w_i(0))$, et donc

$$E_i(t) \leq \left\| \frac{\mathbf{B}_i(t)}{w_i(t)} - \frac{\sum_j \mathbf{B}_j(0)}{\sum_j w_j(0)} \right\|_F^2 + \left\| \frac{\mathbf{a}_i(t)\mathbf{a}_i(t)^\top}{w_i(t)^2} - \frac{(\sum_j \mathbf{a}_j(0))(\sum_j \mathbf{a}_j(0))^\top}{(\sum_j w_j(0))^2} \right\|_F^2 \tag{3.42}$$

Puisque selon les équations (1.83-1.84) chaque entrée de $(\mathbf{B}_i, \mathbf{a}_i, w_i)$ est mise à jour de manière identique, le théorème 6 garantit que

$$\forall i, \quad \lim_{t\to\infty} \frac{\mathbf{B}_i(t)}{w_i(t)} = \frac{\sum_j \mathbf{B}_j(0)}{\sum_j w_j(0)} \quad \text{and} \quad \lim_{t\to\infty} \frac{\mathbf{a}_i(t)}{w_i(t)} = \frac{\sum_j \mathbf{a}_j(0)}{\sum_j w_j(0)} \tag{3.43}$$

Par conséquent, les deux termes de l'équation (3.42) tendent vers zéro, ce qui entraîne finalement la convergence les estimées de covariance locales vers la covariance globale recherchée :

$$\lim_{t\to\infty} \mathbf{C}_i(t) = \mathbf{C} \tag{3.44}$$

□

3.7 Résultats Expérimentaux

Dans cette section, nous évaluons AGPCA sur des données synthétiques et des jeux de données académiques. Les résultats pour les scénarios de type ED (échantillons distribués) et de type CD (coordonnées distribuées) sont rapportés séparément. Dans les deux cas, les données synthétiques ont été générées suivant une superposition de lois normales avec des matrices de covariance aléatoires, telles que $\mathbf{X} \in \mathbb{R}^{D\times n}$ mais avec une dimension intrinsèque (rang) $p \ll D$ et telle que toutes les coordonnées soient corrélées dans $\mathbb{R}^D$. Concernant les données réelles, nous avons utilisé le jeu de données académique MNIST [156] où $n = 60000$ et $D = 784$. Ce jeu de donnée est caractérisé par une dimension intrinsèque faible mais une forte corrélation dans l'espace d'entrée. Sauf spécification contraire, le réseau considéré est totalement connecté et les destinataires sont sélectionnés uniformément.

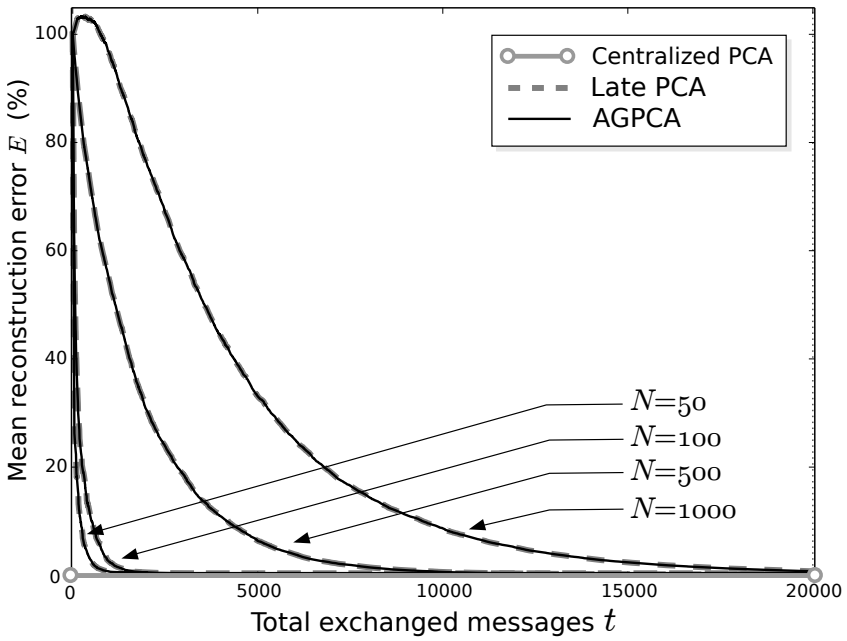

FIGURE 3.2 – Convergence d'AGPCA comparée à une stratégie Late PCA et à la solution analytique de la PCA sur un jeu de données synthétiques généré suivant un mélange de lois normales ($D = 200$, $p = 30$, $n = 10000$), réparties sur $N = 100$ nœuds. Ici, q est fixé à D.

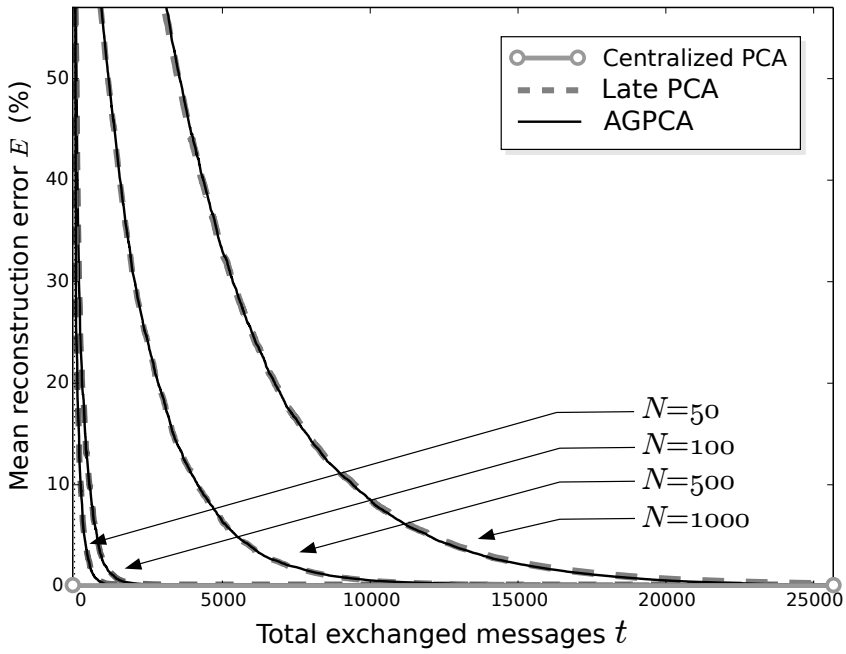

FIGURE 3.3 – Convergence d'AGPCA pour les mêmes données et le même réseau que la figure 3.2, mais avec $q = 30 = p \ll D$.

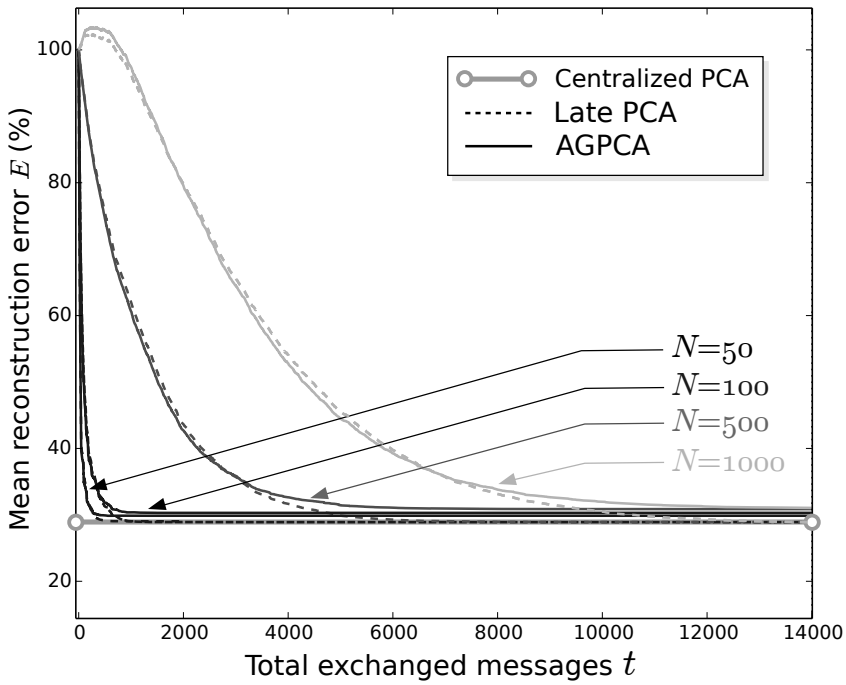

FIGURE 3.4 – Convergence d'AGPCA pour les mêmes données et le même réseau que la figure 3.2, mais avec $q = 10 < p$.

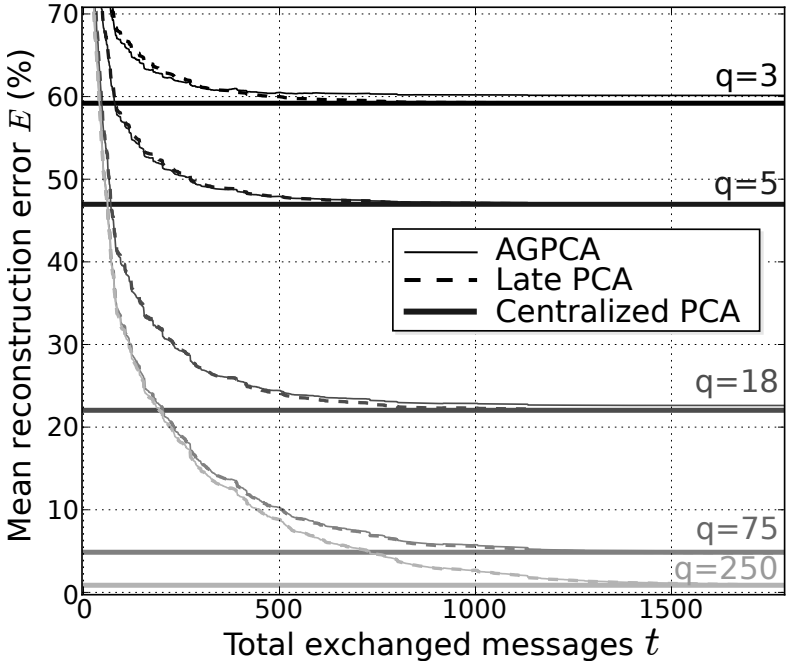

FIGURE 3.5 – Convergence d'AGPCA sur le jeu de données MNIST dispersé uniformément sur $N = 100$ noeuds, pour plusieurs dimensions de sortie q.

3.7.1 Scénarios ED

Dans un scénario ED, rappelons que l'objectif d'AGPCA est double : *(i)* fournir à tous les nœuds la même matrice de projection $\mathbf{U}$, *(ii)* faire converger $\mathbf{U}$ vers la solution analytique $\mathbf{U}^\star$ d'une PCA traditionnelle appliquée à l'ensemble des données du réseau $\mathbf{X}$. Nous mesurons donc l'erreur Euclidienne moyenne de reconstruction entre les matrices de covariance locales $\mathbf{C}_i$ et la covariance exacte $\mathbf{C}$, définie à chaque temps t (*i.e.*, t messages échangés sur le réseau) par

$$E_i(t) = \frac{\|\mathbf{C}_i(t) - \mathbf{C}\|_F^2}{\|\mathbf{C}\|_F^2} \tag{3.45}$$

Lorsque $q = D$, nous avons prouvé dans la section précédente qu'AGPCA converge en tout noeud vers la solution analytique $\mathbf{U}^\star$ de la PCA appliquée à l'ensemble des échantillons du réseau. La Figure 3.2 confirme ce résultat sur des données synthétiques. La Figure 3.3 étend ce résultat au cas où $q = p \ll D$, conformément au Théorème 1 de la section 6. Notons que dans ce cas, aucune matrice $D \times D$ n'est échangée sur le réseau ni même stockée localement, ce qui autorise l'usage d'AGPCA lorsque la dimension d'entrée est trop élevée pour permettre le calcul, le stockage ou l'échange explicite de matrices de covariance complètes.

En revanche, lorsque $q < p$, la projection optimale $\mathbf{U}^\star$ ne rend elle-même plus compte de la totalité de la variance des données d'entrée. L'erreur de reconstruction E n'est donc jamais nulle quel que soit l'algorithme utilisé. Dans un tel cas, AGPCA assure toujours que tous les nœuds convergent vers la même matrice de projection $\mathbf{U}$, garantissant ainsi que toute observation aura le même projeté quel que soit le noeud utilisé pour la traiter. Néanmoins, l'erreur induite par les Itérations Orthogonales se cumule au cours du temps et résulte en une légère déviation par rapport à la solution optimale $\mathbf{U}^\star$. La Figure 3.4 révèle toutefois que même pour des taux de

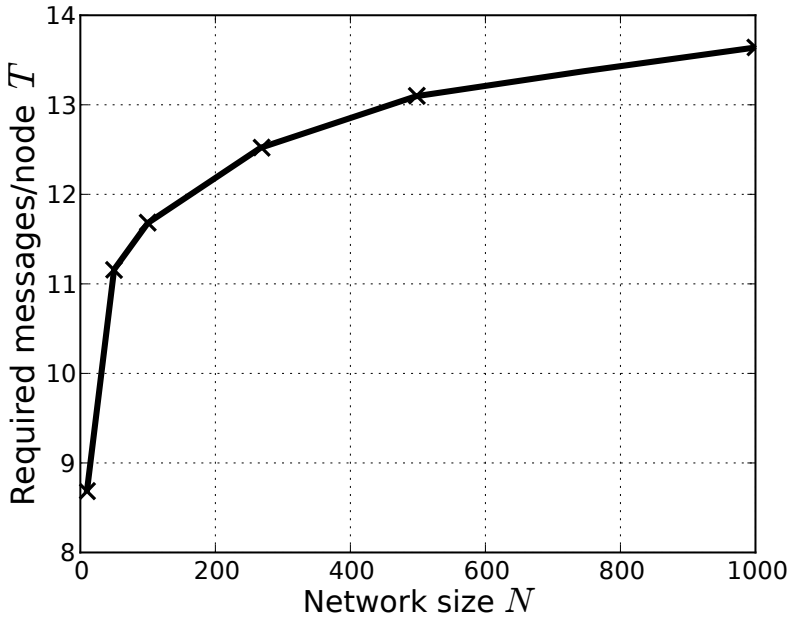

FIGURE 3.6 – Nombre moyen de message émis par noeud à convergence en fonction de la taille du réseau N, évaluée sur MNIST avec $q = 50$

compression élevés (*i.e.*, faible dimension de sortie q), cette déviation reste raisonnablement faible.

Les expériences sur données réelles, conduites sur le jeu de données MNIST et présentées en Figure 3.5, révèlent qu'AGPCA réalise une erreur de reconstruction égale à celle d'une PCA classique à la précision numérique pour $q \geq 75$. Pour des valeurs plus faibles, AGPCA est légèrement sous-optimal dans la mesure où ma déviation est inférieure 1% jusqu'à $q = 3$, et toujours sous les 2%. La Figure 3.5 montre également qu'en dépit de l'erreur propagée par les Itérations Orthogonales, le taux de convergence du protocole d'agrégation n'est pas impacté par le passage d'une stratégie Late PCA à une stratégie Early PCA. On observe d'ailleurs que pour les valeurs faibles de q, la convergence est en réalité plus rapide.

En ce qui concerne les coûts de communication, l'efficacité d'AGPCA est illustrée par la Figure 3.6 en fonction du nombre de nœuds dans le réseau. Celle-ci rapporte le nombre de message que chaque noeud doit émettre en moyenne pour atteindre la convergence du processus. Nous estimons ici que l'algorithme a convergé lorsque l'objectif E ne s'améliore plus que de 0.01% au maximum. La Figure 3.6 met ainsi en évidence la dépendance quasi-logarithmique de ce nombre de messages requis par noeud sur la taille du réseau, ce qui encourage l'utilisation d'AGPCA sur de large réseaux.

3.7.2 Scénarios CD

Dans un scénario CD, notre critère de précision est toujours l'erreur Euclidienne de reconstruction, mais elle est maintenant calculée comme suit :

$$E_i(t) = \|\mathbf{Y}_i(t)^T\mathbf{Y}_i(t) - \mathbf{X}^T\mathbf{X}\| \tag{3.46}$$

Pour évaluer AGPCA dans un scénario CD, nous avons dispersé les coordonnées d'un jeu de données synthétique similaire à la Figure 3.2 ($D = 200$) en affectant un nombre aléatoire uni-

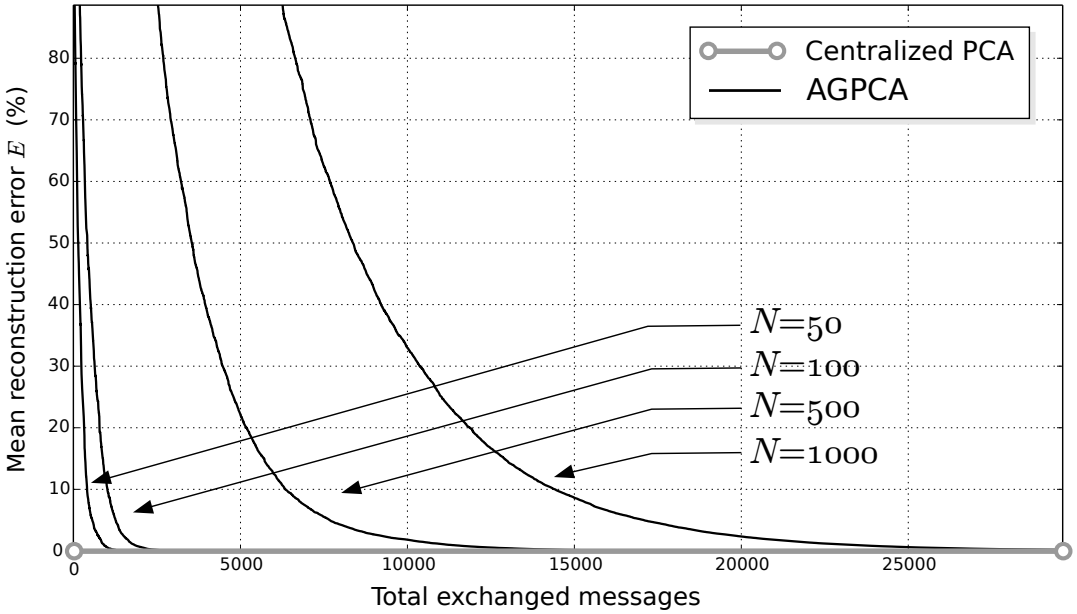

FIGURE 3.7 – Convergence d'AGPCA dans un scénario CD sur des données synthétiques similaires à la Figure 3.2, avec $D = 10000$, $n = 200, p = 30$. Chaque noeud héberge un nombre de coordonnées de chaque échantillon tiré uniformément dans $\{1, \ldots, \frac{2D}{N} - 1\}$. Ici, $q = p = 30$.

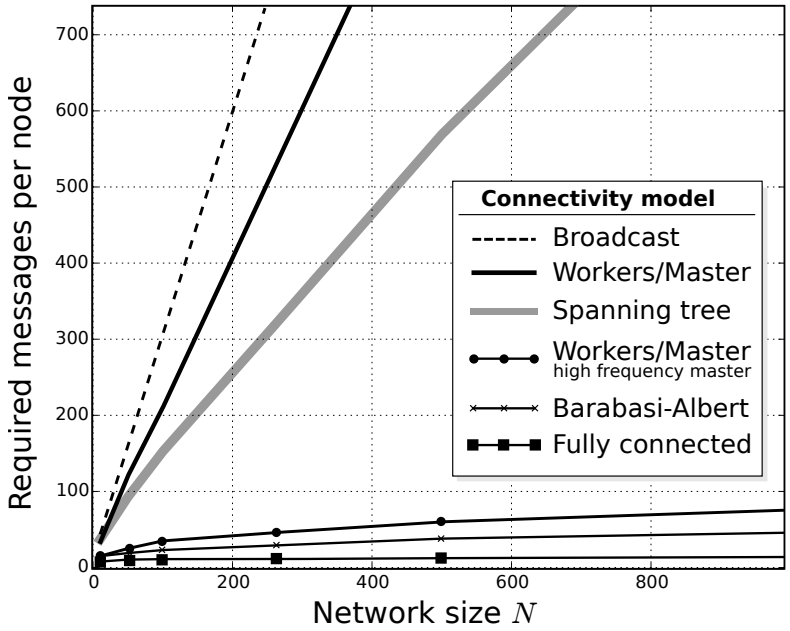

FIGURE 3.8 – Nombre moyen de messages par noeud pour atteindre la convergence pour différents modèles de connectivité réseau. Ici, $q = p = 30$ (même données que pour la Figure 3.7).

forme de dimensions D_i à chaque noeud i, tel que le noeud 1 en possède les D_1 premières dimensions, le noeud 2 les D_2 suivantes, etc, et tel que $\sum_i D_i = D = 200$. La Figure 3.7 montre qu'AGPCA se comporte de manière identique au cas ED, conséquence direct de l'utilisation de la même stratégie d'optimisation pour les deux types de scénarios ED et CD.

3.7.3 Influence de la topologie du réseau

Afin d'évaluer l'influence de la topologie du réseau sur la convergence d'AGPCA, nous avons implémenté divers modèles de connectivité, en soumettant chacun à un fonctionnement totalement asynchrone.

Nous avons tout d'abord considéré un schéma de communication par diffusion (*broadcast*) sur graphe complet, où les émetteurs s'activent régulièrement, sans coordination entre nœuds, et chaque message est envoyé vers tous les voisins. Pour obtenir un tel protocole par diffusion, il nous suffit de remplacer le facteur $1/2$ dans les équations du protocole AGAVG par $1/N$ de sorte à respecter la propriété de conservation de masse.

Nous avons également considéré une structure d'arbre couvrant minimal (*spanning-tree*), où le réseau ne possède que $N-1$ liens, nombre minimal de liens qui conserve la connexité forte du réseau. Dans nos expériences, nous avons utilisé la structure d'arbre non-dégénérée la plus mal conditionnée où un noeud ne peut pas avoir plus que 3 voisins.

Puis nous avons considéré une topologie à noeud maître (*workers-master*), où $N-1$ nœuds appelés *workers* hébergent des échantillons locaux et ne peuvent envoyer de message qu'à un seul noeud identifié comme *maître*. Seul ce noeud maître peut envoyer des messages au *workers*, et celui-ci peut héberger des données locales ou non (si non, ses estimées sont initialisées à 0). Nous avons considéré deux protocoles expérimentaux basés sur cette architecture : dans la première, les *workers* et le maître émettent des messages à une même fréquence (cette situation illustre le cas où les machines physiques utilisées seraient de même performances). Dans la seconde, la fréquence d'émission du maître est N fois plus grande que celle des workers (illustrant le cas où la machine utilisée comme maître est beaucoup plus performante que les autres et possède un débit d'entrée et de sortie élevé caractéristique des solutions industrielles).

Enfin nous avons adopté une connectivité de type Barabási-Albert (BA) [16].

Nous avons appliqué AGPCA sur des donnes synthétiques similaires à la Figure 3.2 pour chacun de ces modèles de connectivité, en faisant varier le nombre de nœuds du réseau. La Figure 3.8 rapporte le nombre moyen de messages par noeud requis pour atteindre la convergence (telle que définie pour la Figure 3.6) et la compare à la connectivité complète adoptée jusqu'à présent dans ce chapitre. Les architectures par diffusion, par noeud maître et par arbre couvrant ne passent pas à l'échelle (le nombre de message par noeud croît quasi-linéairement avec la taille du réseau). Par contraste, les connectivité complètes et BA montrent une dépendance quasi-logarithmique des coûts de communication sur le nombre de nœuds. Dans le cas d'une topologie à noeud maître, le seul moyen de retrouver un impact plus raisonnable de la taille du réseau est de considérer un noeud maître qui communique N fois plus rapidement que les autres nœuds (de telles performances matérielles sont en pratique inatteignables dès que le réseau dépasse une certaine taille critique).

Le comportement décevant de l'approche par diffusion peut être expliqué en remarquant que les émetteurs envoie systématiquement $N-1$ messages parfaitement identiques. Au contraire, en considérant seulement des communications point-à-point chaque message intègre les contribution de *toutes* les mises à jour précédent, rendant la combinaison des estimées beaucoup plus

rapide. Concernant la topologie par arbre couvrant, ses propriétés de mélange très lent ont été largement étudiées dans la littérature sur les protocoles Gossip et la diffusion de rumeur. En raison de sa longueur moyenne de chemin (*average path length*) élevée), l'information doit circuler, en moyenne, à travers un nombre plus grand de nœuds intermédiaires pour se transmettre entre deux nœuds quelconques. Ce type de topologie est donc particulièrement inapproprié à l'usage de protocoles Gossip asynchrones.

Pour résumer, AGPCA, tout comme les protocoles Gossip asynchrones de manière générale, sont plus adaptés aux réseaux utilisant des communications aléatoires, redondantes et beneficiant de la plus faible longueur de chemin moyen possible.

3.8 Conclusion

Dans ce chapitre, nous avons présenté un algorithme décentralisé et asynchrone nommé AGPCA, qui résout le problème de l'Analyse en Composantes Principales lorsque les échantillons d'entraînement sont dispersés sur un réseau. Basé sur l'intégration d'un opérateur de réduction de dimension au sein du protocole de moyennage Gossip de type Sum-Weight décrit dans le chapitre 1, AGPCA est particulièrement adapté aux configurations où la dimension extrinsèque de l'espace d'entrée, le nombre d'échantillons et la taille du réseau sont tous trois élevés. Contrairement à la plupart des algorithmes de réduction de dimension distribuée, AGPCA est utilisable aussi bien dans le scénario où les échantillons sont distribués que dans celui où ce sont les coordonnées des échantillons qui sont dispersées sur le réseau, grâce à la dualité entre les décompositions de la matrice de covariance et la matrice de Gram. Notre analyse théorique et expérimentale montre que la solution produite correspond exactement à la solution analytique de la PCA appliquée à l'ensemble des données du réseau, sous réserve que le rang des échantillons d'entrée soit inférieur à la dimension de sortie. Lorsque cette hypothèse n'est pas vérifiée, les résultats expérimentaux montrent que l'erreur entre la base produite et la solution optimale est faible. Dans le chapitre 5, nous montrons d'ailleurs que lorsque la PCA est utilisée comme opération intermédiaire dans une chaîne de traitement complète, ce niveau d'erreur n'impacte pas significativement les performances des étages avals.

Nos travaux actuels et futurs concernant ce chapitre incluent :

- L'extension d'AGPCA à la réduction de dimension non-linéaire, notamment via des opérateurs d'agrégation localement linéaires (*e.g.*, mélanges probabilistes de lois Gaussiennes [45]).
- L'analyse formelle de l'impact de la distribution (notamment spectrale) des données sur l'erreur d'approximation induite lorsque l'hypothèse de rang faible n'est pas vérifiée.
- L'évaluation du comportement d'AGPCA sur des données dynamiques non-stationnaires, subissant par exemple une dérive sémantique (*concept drift*) au cours du temps. Grâce aux invariances aux changements de topologie du réseau et donc à l'introduction et la disparition d'échantillons, on peut espérer qu'AGPCA offre de bonnes propriétés comportementales sur des tâches de poursuite de sous-espace en ligne (*online basis pursuit*).

CHAPITRE 4

Classification supervisée et optimisation convexe

Les succès récents des méthodes d'optimisation stochastiques telles que les techniques de descente de gradient moyenné (*stochastic average gradient*) ont renouvelé l'interêt des approches par descente de gradient pour résoudre les problèmes convexes de grande taille [184, 208]. Cependant, lorsque le nombre de paramètres dépasse le million et que l'ensemble d'entraînement approche du Téra-octet, la resolution d'un problème convexe reste computationnellement très couteuse, même lorsque l'expression du gradient est très simple. Ainsi, les progrès considérables réalisés par les méthodes d'approximation stochastiques ne suffisent pas à contrer l'explosion de la quantité de données à manipuler.

Par ailleurs, la nature stochastique de ces approches favorise la formulation d'équivalents parallèlisés voire distribués à ces algorithmes. Toutefois, nous mettons en évidence dans ce chapitre qu'il n'est pas trivial d'en proposer une extension décentralisée et asynchrone qui conserve de bonne propriétés de stabilité.

L'optimisation convexe consiste a minimiser une certaine fonction $f : \mathbb{R}^D \to \mathbb{R}$ convexe, c'est à dire telle que :

$$\forall \mathbf{v}, \mathbf{w} \in \mathbb{R}^D, \quad f(\mathbf{w}) \geq f(\mathbf{v}) + \nabla f(\mathbf{v})^\top (\mathbf{w} - \mathbf{v}) \tag{4.1}$$

L'hypothèse de convexité assure que tout minimum local de f en est également un minimum global. Notons que f n'est pas nécessairement dérivable. En tout point où f est dérivable, $\nabla f(\mathbf{w})$ désigne le gradient de f en $\mathbf{w}$. Pour les points où f n'est pas lisse, $\nabla f(\mathbf{w}) \in \partial f(\mathbf{w})$ désigne alors un sous-gradient de f en $\mathbf{w}$. Parmi les problèmes convexes, une sous-classe importante est constituée des problèmes fortement convexes, pour lesquels

$$\forall \mathbf{v}, \mathbf{w} \in \mathbb{R}^D, \quad f(\mathbf{w}) \geq f(\mathbf{v}) + \nabla f(\mathbf{v})^\top (\mathbf{w} - \mathbf{v}) + \frac{\sigma}{2} \|\mathbf{w} - \mathbf{v}\|_2^2, \tag{4.2}$$

où $\sigma > 0$ est la constante de convexité de f. Les problèmes fortement convexes sont caractérisés par l'existence d'un unique minimiseur $\mathbf{w}^\star$.

Dans de nombreuses taches d'apprentissage statistique, on cherche à optimiser l'espérance d'une certaine fonction d'erreur ℓ, convexe, que l'on estime empiriquement comme la moyenne

de ses évaluations sur un ensemble de n échantillons d'entraînement $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$:

$$\forall \mathbf{w} \in \mathbb{R}^D, \quad f(\mathbf{w}) = \mathbb{E}\ell(\mathbf{w}) \approx \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}, \mathbf{x}_i) \tag{4.3}$$

On adjoint parfois à cette erreur empirique un terme de régularisation, afin de favoriser certaines caractéristiques sur $\mathbf{w}$ (*e.g.*, parcimonie par une norme ℓ_1 ou plus généralement par une norme ℓ_p fractionnaire, où p se situe entre 0 et 2). Par exemple, dans le cas d'une régularisation ℓ_2, f devient

$$\forall \mathbf{w} \in \mathbb{R}^D, \quad f(\mathbf{w}) = \lambda\|\mathbf{w}\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}, \mathbf{x}_i) \tag{4.4}$$

où $\lambda \in \mathbb{R}^+$ contrôle le niveau de régularisation souhaité. Ce type de reformulation a l'avantage d'assurer la convexité forte (on a ici $\sigma \geq \lambda$) et ainsi de privilégier, parmi les solutions offrant une risque empirique identique, celle qui minimise également un certain risque structurel [242]. Le pouvoir de généralisation s'en trouve généralement accru, limitant ainsi les risques de sur-apprentissage sur les données d'entraînement.

Ce chapitre est organisé comme suit. La section 1 présente un exemple emblématique d'optimisation convexe : la classification supervisée par Machines à Vecteurs Support (SVM). Dans la section 2, nous détaillons quelques méthodes de résolution primales basées sur la descente de gradient stochastique (SGD). La section 3 définit l'optimisation convexe en environnement distribué. Dans la section 4, nous présentons un algorithme de résolution primal qui s'appuie sur le moyennage à court terme des gradients (*Short Term Average Gradient*, STAG). La section 5 est consacrée à son extension décentralisée et asynchrone (*Asynchronous Gossip Short Term Average Gradient*, AGSTAG). Son evaluation expérimentale est présentée en section 6, avant de conclure le chapitre.

4.1 Classification supervisée par Machines à Vecteurs Supports (SVM)

Dans ectte section, nous présentons les problèmes d'optimisation liés à la classification supervisée, en particulier les Machines à Vecteurs Supports (SVM). En effet, ce sont ces algorithmes que nous souhaitons adapter aux contexte décentralisé.

4.1.1 Définition du problème

La classification supervisée consiste à estimer une fonction $h : \mathbb{R}^D \to \{0, \ldots, C-1\}$ appelée classificateur qui prédit pour tout vecteur d'entrée son appartenance à une des C classes identifiées par un label entier entre 0 et $C-1$. La classe réelle correspondant à un vecteur donné est supposée inconnue, le but de la tâche étant d'estimer cette classe à partir d'un ensemble de couples d'apprentissage $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ où $y_i \in \{0, \ldots, C-1\}$ est le label qui doit être associé au vecteur $\mathbf{x}_i \in \mathbb{R}^D$. Lorsque $C > 2$, on parle de classification multi-classe. Lorsque $C = 2$, on parle de classification binaire, et pour des facilités de notation, les deux classes possibles sont labellisées $\{-1, +1\}$.

Un modèle de classification binaire linéaire, ou par hyperplan, estime h comme le signe d'un discriminant linéaire qui projette le vecteur d'entrée sur la normale $\mathbf{w}$ à un hyperplan de

dimension D :

$$\forall \mathbf{w} \in \mathbb{R}^D, b \in \mathbb{R}, \mathbf{x}_i \in \mathbb{R}^D, \quad h(\mathbf{w}, b, \mathbf{x}_i) = \text{sgn}(\mathbf{w}^\top \mathbf{x}_i + b) \tag{4.5}$$

Ainsi, l'hyperplan défini par la normale $\mathbf{w}$ et la distance à l'origine b sépare linéairement $\mathbb{R}^D$ en deux classes -1 et 1.

Le Perceptron [206] fut l'un des premiers algorithmes d'apprentissage de classificateurs linéaires. Il consiste à minimiser itérativement l'erreur de classification sur l'ensemble d'entraînement :

$$\min_{\mathbf{w},b} f(\mathbf{w}, b) = \frac{1}{n} \sum_{i=1}^{n} \Big(y_i - h(\mathbf{w}, b, \mathbf{x}_i) \Big)^2 \tag{4.6}$$

On peut généralement se passer du biais b en ajoutant si besoin une entrée égale à 1 à tous les échantillons $\mathbf{x}_i$, le paramètre correspondant $\mathbf{w}_{D+1}$ jouant alors le rôle de b. Dans la suite de ce chapitre, nous ignorerons donc le paramètre b. L'objectif (4.6) est convexe mais pas fortement convexe, dans la mesure où il existe généralement une infinité d'hyperplans distincts qui offrent une erreur minimale. Par ailleurs, la valeur minimale de f n'est nulle que lorsque les échantillons sont linéairement séparables [179].

4.1.2 Machines à Vecteurs Supports (SVM)

Les Machines à Vecteurs Supports ou Séparatrices à Vaste Marge (*Support Vector Machines*, SVM) proposent trois améliorations majeures au Perceptron qui en ont fait une méthode incontournable depuis la fin des années 1990.

Premièrement, parmi tous les hyperplans optimaux, tous n'offrent pas la même robustesse au bruit [240]. Un moyen de traduire cette robustesse est de considérer la marge de séparation entre les classes [243]. L'introduction de cette marge correspond à fixer la contrainte suivante :

$$\forall i, \quad y_i \mathbf{w}^\top \mathbf{x}_i \geq \gamma \tag{4.7}$$

En utilisant l'invariance de l'hyperplan à la norme de $\mathbf{w}$, on peut sans perte de généralité fixer $\gamma = 1$:

$$\forall i, \quad y_i \mathbf{w}^\top \mathbf{x}_i \geq 1 \tag{4.8}$$

La marge est alors donnée comme la distance entre les deux hyperplans $\mathbf{w}^\top \mathbf{x} = -1$ et $\mathbf{w}^\top \mathbf{x} = 1$ qui est égale à $2/\|\mathbf{w}\|_2$. Ainsi plus la norme ℓ_2 de $\mathbf{w}$ diminue, plus la marge augmente. L'objectif d'une SVM est alors défini comme suit :

$$\text{minimiser} \quad \|\mathbf{w}\|_2 \quad \text{sous la contrainte} \quad \forall i, y_i \mathbf{w}^\top \mathbf{x}_i \geq 1 \tag{4.9}$$

Le problème devient alors fortement convexe est l'hyperplan optimal unique est celui qui maximise la marge de séparation entre les classes.

Deuxièmement, la présence de points aberrants ou de bruit peut rendre l'ensemble d'entraînement non linéairement séparable, même si le phénomène observé l'est en réalité. Il n'est alors plus possible de respecter la contrainte de (4.9). L'introduction d'une marge souple (*soft margin*

[62]) permet de résoudre ce problème en remplaçant cette contrainte par un compromis entre large marge et erreur de classification :

$$\text{minimiser} \quad \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \frac{1}{n}\sum_{i=1}^{n}\max(0, 1 - y_i\mathbf{w}^\top\mathbf{x}_i) \tag{4.10}$$

l'hyper-paramètre $\lambda > 0$ contrôle la préférence entre une large marge et une faible erreur de classification, prévenant ainsi la sensibilité aux points aberrants mal classifiés. En pratique les SVM à marge souple garantissent ainsi une robustesse accrue.

Troisièmement, à l'objectif fortement convexe (4.10) correspond un problème dual équivalent qui n'implique les échantillons qu'à travers leurs produits scalaires [115] :

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j}^{n} y_i y_j \alpha_i \alpha_j \mathbf{x}_i^\top \mathbf{x}_j \quad \text{avec} \quad \forall i, 0 \leq \alpha_i \leq \lambda^{-1}. \tag{4.11}$$

Le théorème des représentants [7, 12] établit en particulier que l'hyperplan optimal s'exprime comme une combinaison linéaire des exemples d'apprentissages :

$$\mathbf{w}^\star = \sum_{i}^{n} \alpha_i y_i \mathbf{x}_i, \tag{4.12}$$

où un bon nombre de coefficients α_i sont nuls. L'hyperplan optimal est donc défini par un petit nombre d'échantillons appelés *vecteurs supports*. Ces échantillons sont justement ceux qui se situent sur la frontière de décision ou l'enfreignent. Par ailleurs, le théorème de Mercer [176] révèle que toute fonction noyau, semi-définie positive, est équivalente à un produit scalaire dans un certain espace de Hilbert (de dimension potentiellement infinie). On peut donc substituer au produit scalaire initial $\mathbf{x}_i^\top \mathbf{x}_j$ une fonction noyau (de Mercer) quelconque $\kappa(\mathbf{x}_i, \mathbf{x}_j)$. La classification linéaire se fait alors dans l'espace induit par ce noyau, et la fonction de décision ne fait plus intervenir que la similarité du vecteur d'entrée aux vecteurs supports donnée par κ :

$$h(\mathbf{x}) = \operatorname{sgn}\Big(\sum_{i}^{n} \alpha_i y_i \kappa(\mathbf{x}, \mathbf{x}_i)\Big) \tag{4.13}$$

Si le problème n'était pas linéairement séparable au départ, il a de grande chances de le devenir dans l'espace induit de dimension supérieure. Ce "truc du noyau" autorise ainsi la classification non-linéaire via la résolution du dual (4.11) muni d'un noyau non-linéaire (*e.g.*, gaussien, polynomial, etc).

Le problème dual (4.11) étant un programme quadratique, il peut être résolu efficacement par des méthodes de point intérieur (IPM [174]), par décomposition [52, 60, 143, 203] ainsi que leurs extensions en ligne [31] ou par ascension de coordonnées duales (DCA [222]). De nombreuses implémentations logicielles de solveurs duals sont librement disponibles[1] (voir par exemple [132]).

4.1.3 Pertinence de la formulation primale

Si l'élégance de la formulation duale a rendu les SVM non-linéaires très populaires jusqu'au milieu des années 2000, des travaux récents remettent en question son utilisation systématique.

[1] La plupart de implémentations de solveurs SVM sont disponible sur `http://mloss.org/software/`

En effet, les nouveaux résultats théoriques et empiriques acquis ces dix dernières années ont mis en évidence la pertinence de solutions primales en s'appuyant sur les arguments suivants :

1. La complexité calculatoire d'un solveur primal, *i.e.*, son temps de convergence, ne dépend généralement pas du nombre d'échantillons n [222], là où la plupart des solveurs duals passent difficilement à l'échelle lorsque n augmente [31]. Les méthodes de gradient stochastique offrent alors une alternative pour les grands ensembles de données [32].

2. Généralement, on ne cherche pas la solution exacte au problème (4.10), dans la mesure où l'erreur empirique est seulement évaluée sur l'ensemble d'apprentissage tandis que l'objectif final est de minimiser l'erreur en phase de test, c'est à dire sur des données inobservables durant l'entraînement. Ainsi, on observe en pratique que lorsqu'une précision suffisante est atteinte sur l'ensemble d'entraînement, il est inutile de raffiner le modèle car l'erreur en utilisation ne progresse plus [32]. Comme nous le verrons dans la section suivante, les méthodes primales convergent bien plus rapidement pendant les premières itérations, puis ralentissent significativement à l'approche de l'optimum. Puisqu'une convergence approximative est suffisante, les méthodes primales produisent ainsi en pratique une solution acceptable beaucoup plus rapidement que leurs homologues duales.

3. Pour les problèmes de classification linéaires, un solveur primal est généralement plus rapide qu'un solveur dual [55]. Par ailleurs, de nouveaux noyaux explicites, où l'image d'un vecteur d'entrée dans l'espace de redescription peut être effectivement calculée et stockée efficacement ont récemment montré des performances de classification par SVM linéaire égales voire supérieures aux méthodes à noyaux implicites non-linéaires (par exemple en classification d'images : [137, 182, 197, 200]). La combinaison de ces deux facteurs conduit à une utilisation croissante d'encodeurs de caractéristiques explicites suivis d'une SVM linéaire lorsque la quantité d'échantillons d'apprentissage devient très grande.

4. Dans le cas spécifique des SVM souples utilisant une fonction de coût de type *hinge loss* comme (4.10), de nombreuses astuces calculatoires accélèrent remarquablement les méthodes stochastiques primales [32, 221].

Pour ces raisons nous nous concentrons dans le présent chapitre sur les méthodes de résolution primales linéaires. Dans le chapitre 5, nous verrons que ces méthodes primales produisent très rapidement de très bons résultats sur des problèmes de classification large échelle.

4.2 Résolution par descente de gradient

Pour résoudre un problème convexe, tel que l'expression primale (4.10), une des méthodes les plus utilisées est la descente de gradient [51]. Elle consiste à suivre itérativement la direction négative du gradient de la fonction à optimiser :

$$\forall t, \quad \mathbf{w}(t+1) = \mathbf{w}(t) - \eta \nabla f(\mathbf{w}(t)), \tag{4.14}$$

où $\nabla f(\mathbf{w})$ représente le gradient de f en $\mathbf{w}$, et η est un petit pas d'apprentissage, qui peut être constant ou variable dans le temps. La descente de gradient approxime donc f par son développement de Taylor d'ordre 1 (*i.e.*, affine) pour en trouver à chaque étape la direction de plus forte décroissance. Pour des valeurs appropriées de η, on a clairement $f(\mathbf{w}(t+1)) \leq$

$f(\mathbf{w}(t))$. La descente de gradient converge ainsi vers le minimum global $\mathbf{w}^\star$. Pour η constant, il est prouvé que cette convergence est linéaire :

$$f(\mathbf{w}(t)) - f(\mathbf{w}^\star) = \mathcal{O}(\rho^t), \quad \text{où} \quad 0 < \rho < 1 \tag{4.15}$$

On peut même obtenir un taux de convergence quadratique en intégrant des informations du second ordre (matrice Hessienne), par des méthodes de type Newton [55] ou Quasi-Newton [11, 30, 216]. Dans le cas où f est quadratique, la méthode de Newton trouve de fait l'optimum en une seule étape.

4.2.1 Descente de Gradient Stochastique (SGD)

Le principal défaut des méthodes par descente de gradient déterministes (*full gradient*, FG) est le coût calculatoire élevé de leurs itérations. En effet, lorsque f est la somme d'un grand nombre de termes convexes f_i, comme c'est le cas pour (4.10), l'évaluation du gradient total de f implique le calcul du gradient de chaque f_i. Ainsi, le coût d'une itération croît linéairement avec le nombre d'échantillons d'entraînement. En dépit de leur taux de convergence favorable, les méthodes déterministes ont en pratique un temps de convergence long. Une alternative est d'utiliser une Descente de Gradient Stochastique (*Stochastic Gradient Descent*, SGD [35]), où chaque itération ne considère qu'un seul exemple d'apprentissage sélectionné aléatoirement (*e.g.*, uniformément) :

$$\forall t, \quad \mathbf{w}(t+1) = \mathbf{w}(t) - \eta \nabla f_i(\mathbf{w}(t)), \quad \text{où} \quad i \sim \mathcal{U}_{\{1,\dots,n\}} \tag{4.16}$$

Ici, le coût computationnel d'une itération ne dépend plus du nombre d'échantillons. De plus, on tire un avantage direct de la redondance éventuelle dans les données d'entrée. Intuitivement, dans le cas extrême où tous les échantillons $\mathbf{x}_i$ sont égaux, les termes f_i et leur dérivées sont alors identiques, d'où $\nabla f = n \nabla f_i$. Dans ce cas, une itération de SGD apporte le même progrès vers l'optimum qu'une itération de FG, pour un coût calculatoire n fois moindre. Dans le cas général, toute redondance dans les gradients avantage SGD, qui se révèle par conséquent beaucoup plus rapide que FG sur des problèmes pratiques impliquant un grand nombre d'échantillons.

Malheureusement, la restriction du calcul de gradient à un seul exemple d'apprentissage par itération induit une grande instabilité du processus, qui devient très sensible au bruit. Par conséquent, lorsque le pas d'apprentissage η est choisi constant, l'estimée oscille autour de l'optimum, dans un rayon qui croît avec η (cycle limite). Ainsi, malgré une première phase de convergence très rapide comparée à FG, on ne peut plus espérer une convergence exacte et l'on doit trouver un compromis entre un η faible pour une convergence précise mais lente et un η élevé pour une convergence rapide mais imprécise.

Une alternative est d'adopter un pas d'apprentissage η_t décroissant dans le temps [220], afin d'assurer une convergence plus rapide dans les premières itérations tout en garantissant une convergence plus fine dans la dernière phase. Pour éviter la convergence vers un non-minimum par une décroissance trop rapide du pas, les contraintes suivantes doivent être respectées :

$$\sum_t^\infty \eta_t = \infty \quad \text{et} \quad \sum_t^\infty \eta_t^2 < \infty \tag{4.17}$$

Malheureusement, la décroissance du pas η_t empêche toute convergence linéaire. En effet, il a été montré formellement que le meilleur taux de convergence atteignable par tout algorithme

de SGD n'exploitant pas la structure spécifique du problème considéré est sous-linéaire [77] :

$$\mathbb{E}f(\mathbf{w}(t)) - f(\mathbf{w}^\star) = \mathcal{O}(1/t) \tag{4.18}$$

4.2.2 Approches par gradient moyenné

En 2009, Nesterov [184] propose une explication au taux de convergence désavantageux de SGD : les gradients nouvellement évalués sont intégrés au modèle avec un poids inférieur aux gradients plus "âgés". Cette propriété étant indissociable des hypothèses (4.17) et donc inhérente à toute méthode de mise à jour par gradients stochastiques, Nesterov suggère l'introduction d'un second processus évoluant dans l'espace dual et dirigeant les mises à jour du processus primal :

$$\mathbf{z}(t+1) = \frac{t\mathbf{z}(t) + \nabla f(\mathbf{w}(t))}{t+1} \quad \text{et} \quad \mathbf{w}(t+1) = \arg\min_{\mathbf{w}} \left[\mathbf{w}^\top \mathbf{z}(t+1) + \frac{1}{\eta_t}\psi(\mathbf{w})\right] \tag{4.19}$$

où $\psi(\cdot)$ est une fonction dite proximale, fortement convexe sur $\mathbb{R}^D$ et telle que $\psi(\mathbf{w}(0)) = 0$ et η_t un pas d'apprentissage décroissant. Cette procédure, nommée *Dual Averaging* (DA), n'est pas une descente de gradient en tant que telle, puisque $\mathbf{w}(t+1)$ n'est plus mis à jour à partir de $\mathbf{w}(t)$ mais par projection de l'estimée duale $\mathbf{z}$ via l'opérateur proximal. De son côté, $\mathbf{z}$ est la moyenne (non-pondérée) de tous les gradients évalués jusqu'à t. Dans (4.19), on observe donc que *tous* les gradients précédents interviennent dans la mise à jour de $\mathbf{w}$ avec un poids *identique*. La convergence est assurée non plus par la décroissance des poids associés aux gradients, mais par la "tension" décroissante avec $\mathbf{w}(0)$ induite par l'opérateur proximal. DA a fait l'objet de nombreuses variantes et extensions, notamment pour les problèmes d'apprentissage régularisés [226, 254]. Bien que DA atteigne un taux de convergence en $\mathcal{O}(\frac{1}{\sqrt{t}})$, la convergence exponentielle de FG reste toutefois inaccessible. C'est également le cas des méthodes intégrant un terme de moment, où l'estimée précédente est réinjectée dans le calcul de l'estimée suivante avec un poids $\alpha < 1$:

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta\nabla f_i(\mathbf{w}(t)) + \alpha(\mathbf{w}(t) - \mathbf{w}(t-1)) \tag{4.20}$$

La mise à jour (4.20) équivaut à intégrer *tous* les gradients précédents avec un poids décroissant géométriquement. Cette approche échoue également à obtenir une convergence linéaire.

Dans [29], Blatt *et al.* proposent une méthode de descente de gradient incrémentale accélérée (*Incremental Accelerated Gradient*, IAG) qui obtient avec un pas constant une convergence linéaire pour les problèmes quadratiques. Elle consiste à mettre à jour $\mathbf{w}$ en considérant la moyenne des n gradients les plus récemment évalués au lieu du seul gradient le plus récent :

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \frac{\eta}{n}\sum_{l=0}^{n-1} \nabla f_{(t-l)_n}(\mathbf{w}(t-l)), \tag{4.21}$$

où $(k)_n$ dénote k modulo n. En soi, IAG n'est pas une méthode stochastique, puisque les échantillons d'apprentissage ne sont pas choisis aléatoirement mais cycliquement. En dehors des objectifs quadratiques, la convergence linéaire d'IAG n'est pas formellement établie. Certains résultats empiriques [222] montrent par ailleurs qu'une telle sélection strictement cyclique des échantillons d'apprentissage entraîne une convergence plus lente, car les échantillons sont toujours visités dans le même ordre.

En 2012, Le Roux *et al* [208] lèvent les limitations d'IAG en proposant une méthode de Descente de Gradient Moyennés Stochastique (*Stochastic Average Gradient*, SAG). En effet, celle-ci exhibe une convergence linéaire pour tout objectif fortement convexe avec un pas constant. Au prix de constantes multiplicatives légèrement défavorables [213], elle renoue ainsi avec le taux de convergence en $\mathcal{O}(\rho^t)$ de la descente de gradient complète déterministe. Dans le même esprit qu'IAG, la mise à jour considère la moyenne pour chaque échantillon du gradient le plus récemment évalué, mais cette fois-ci la sélection des échantillons se fait uniformément :

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \frac{\eta}{n}\sum_{l=1}^{n}\mathbf{g}_l(t) \quad \text{où} \quad \left\{ \begin{array}{l} \mathbf{g}_i(t) = \nabla f_l(\mathbf{w}(t)) \\ \forall l \neq i, \mathbf{g}_l(t) = \mathbf{g}_l(t-1) \end{array} \right. , \quad i \sim \mathcal{U}_{\{1\ldots n\}} \tag{4.22}$$

SAG conserve ainsi pour chaque échantillon l une mémoire $\mathbf{g}_l(t)$ du gradient le plus récemment évalué sur f_l, qui est mis à jour lorsque l est sélectionné. La procédure (4.22) peut être vue comme une approximation de FG où les gradients partiels $\nabla f_i(t)$ sont évalués avec un retard distribué géométriquement. La preuve présentée dans [213] établit alors que ce retard n'influence pas le taux de convergence asymptotique de la méthode, tout en offrant un coût par itération n fois plus faible. En pratique, SAG se révèle par conséquent beaucoup plus rapide que ses concurrentes sur des problèmes large échelle. Notons que (4.22) ne définit pas les $\mathbf{g}_i(0)$ initiaux. Les auteurs suggèrent deux stratégies d'initialisation :

1. Réaliser la première passe sur l'ensemble des données (première époque) par SGD classique, puis utiliser les gradients mémorisés comme initialisation de SAG pour les époques suivantes.

2. Utiliser directement (4.22) dès la première époque, mais ne considérer que les gradients déjà évalués dans la moyenne des $\mathbf{g}_l(t)$. Cette initialisation se montre dans les faits plus rapide [213].

Des travaux récents ont confirmé et étendu les résultats de SAG en clarifiant notamment le lien qu'elle entretient avec les méthodes proximales telles que DA et en offrant des bornes plus explicites [26, 67, 255]. Par ailleurs, des travaux récents ont mis en évidence de manière principalement empirique la pertinence de stratégies de sélection distinctes de la sélection cyclique déterministe et de la sélection uniforme totalement stochastique. Par exemple, dans [34, 222], il apparaît que la vitesse de convergence de SGD est significativement améliorée en adoptant une stratégie mixte qui effectue des passes déterministes à travers l'ensemble d'entraînement, mais qui réordonne aléatoirement les indices des échantillons après chaque passe (*dataset shuffling*). Ce schéma simule ainsi une sélection uniforme *sans remise*. Intuitivement, une telle sélection sans remise à l'avantage de réduire la redondance des gradients qui peut survenir par la sélection d'un même exemple plusieurs fois dans un laps de temps réduit. En effet, si à l'instant $t = n$ une stratégie uniforme avec remise sélectionne un échantillon non encore visité avec une probabilité constante $\frac{1}{n}$, un schéma sans remise le sélectionnera avec une probabilité 1, puisque les $n-1$ autres ont déjà été exploités. En dépit des gains généralement observés en pratique, les stratégies sans remises sont encore peu considérées d'un point de vue formel, probablement du fait des complications théoriques apportées par la dépendance temporelle, inexistante avec une stratégie uniforme avec remise (la plupart des résultats formels de convergence concernant les méthodes apparentées à SAG, qui font d'emblée une hypothèse d'indépendance et de stationnarité, ne se généralisent pas trivialement au cas sans remise).

4.3 Optimisation convexe distribuée

Dans cette section, nous décrivons l'état de l'art sur l'optimisation convexe distribuée, afin de faire le lien entre les algorithmes de résolution de SVM et notre contexte distribué.

4.3.1 Définition du problème

Étant donné un réseau fortement connecté composé de N nœuds, chacun i hébergeant un ensemble d'échantillons $\mathbf{X}_i = [\mathbf{x}_1^{(i)} \dots \mathbf{x}_{n_i}^{(i)}]$, $\mathbf{x}_j^{(i)} \in \mathbb{R}^D$, on souhaite trouver l'unique minimum global $\mathbf{w}^\star \in \mathbb{R}^Q$ de la fonction objectif fortement convexe suivante :

$$\forall \mathbf{w} \in \mathbb{R}^Q, \qquad f(\mathbf{w}) = \sum_{i=1}^{N} \sum_{j=1}^{n_i} f_j^{(i)}(\mathbf{w}), \tag{4.23}$$

où chaque fonction locale $f_j^{(i)}$, fortement convexe, est associée à l'échantillon $\mathbf{x}_j^{(i)}$.

Chaque noeud i commence avec une estimée locale $\mathbf{w}^{(i)}$ aléatoire et doit la conduire vers $\mathbf{w}^\star$ en utilisant uniquement des calculs locaux sur son propre ensemble d'entraînement $\mathbf{X}^{(i)}$ et des échanges de messages avec ses nœuds voisins. Comme présenté dans le chapitre 1, nous fixons également les contraintes suivantes :

C1. Pas d'échange d'échantillons. Seules les estimées peuvent être échangées entre noeuds.

C2. Asynchronisme - Les nœuds ne doivent jamais s'attendre les uns les autres.

C3. Décentralisation - Tous les nœuds et tous les liens doivent jouer le même rôle (*i.e.*, tous les nœuds exécutent le même algorithme local et les liens sont sélectionnées uniformément).

4.3.2 Approches géométriques

De nombreux algorithmes distribués considèrent le cas spécifique des Machines à Vecteurs Supports, profitant ainsi des propriétés structurelles du problème. Certaines approches reprennent ainsi la formulation du séparateur à hyperplan optimal comme celui qui maximise la marge entre les deux classes. On peut trouver cet hyperplan optimal grâce aux enveloppes convexes des deux classes. En effet, sa normale $\mathbf{w}^\star$ est exactement le vecteur entre les deux points les plus proches des deux enveloppes convexes [22].

Ainsi, Flouri *et al*. [97] proposent de rechercher en chaque noeud les vecteurs définissant les enveloppes convexes des deux classes, puis de fusionner ces enveloppes convexes entre nœuds jusqu'à obtenir un ensemble de points définissant les enveloppes convexes des deux classes pour tous les échantillons du réseau. L'hyperplan optimal est alors calculé simplement comme la différence entre les deux points les plus proches des deux enveloppes obtenues. Dans [97], les nœuds doivent donc échanger à chaque étape l'ensemble des vecteurs supportant l'enveloppe convexe courante. En grande dimension et selon le problème, le nombre de vecteurs à échanger peut devenir très grand, impliquant un coût de communication important. Par ailleurs, le coût de calcul local d'une enveloppe convexe ou d'une union d'enveloppe convexe est élevé, notamment par rapport à une itération de descente de gradient. Enfin, le temps de convergence explose lorsque le réseau considéré a plusieurs milliers de nœuds.

Une autre famille de solveurs SVM distribués résolvent les sous-problèmes correspondant à chaque noeud séparément via une méthode centralisée classique puis transmettent les vecteurs

supports à un noeud maître qui calcule une approximation de l'optimum global [31, 111]. Malheureusement, pour que le résultat corresponde à celui du problème global, la distribution des échantillons et leur réparation sur les nœuds doivent respecter un ensemble d'hypothèses très contraignantes qui en limite l'applicabilité.

4.3.3 Approches duales

Une large classe de méthodes s'attachent à résoudre le problème quadratique dual (4.11) en s'inspirant des méthode de résolution centralisées classiques.

PGPDT [258, 259] est une extension distribuée de SVM^{light} [143] qui repose sur la parallélisation des deux étapes coûteuses de résolution des sous-problèmes quadratiques et de mise à jour.

Dans [75], les auteurs utilisent une approximation bloc-diagonale de la matrice noyau pour rendre le problème séparable en SVM locales, en ajoutant une étape de filtrage des échantillons estimés comme n'étant pas des vecteurs support. Une SVM classique est alors entraînée en prenant comme ensemble d'apprentissage les vecteurs supports collectés sur les sous problèmes.

PSVM [53] approxime la matrice noyau par une factorisation de Cholesky Incomplète dont le calcul est distribué sur les nœuds. Le programme quadratique (4.11) est alors résolu par un algorithme de point intérieur parallélisé. L'erreur d'approximation de l'optimum est alors proportionnelle à l'erreur euclidienne de factorisation [96].

D'autres approches découpent le problème primal (4.10) en sous problèmes locaux soumis à une contrainte de consensus, par exemple :

$$\min_{\mathbf{w}_1,\ldots,\mathbf{w}_N} = \sum_{i=1}^{N} \left(\frac{\lambda}{2} \|\mathbf{w}_i\|_2^2 + \frac{1}{n_i} \sum_{l=1}^{n_i} \max(0, 1 - y_{i,l} \mathbf{w}_i^\top \mathbf{x}_{i,l}) \right) \tag{4.24}$$
$$\text{sous la contrainte } \forall i, j, \mathbf{w}_i = \mathbf{w}_j$$

Ce type de problèmes convexes séparables soumis à une contrainte affine (ici notre contrainte d'égalité) peut être efficacement résolu par la méthode des Directions Alternées (*Alternating Direction Method of Multipliers*, ADMM [40, 101, 106]). MoM-DSVM [98] applique cette approche générale au problème spécifique des SVM, résultant en un solveur décentralisé localement synchrone où seuls les hyperplans locaux sont échangés entre nœuds. MoM-DSVM procède en trois étapes répétées :

1. Chaque noeud met à jour son hyperplan en résolvant un certain programme quadratique dual local via une méthode classique (*e.g*, IPM, gradient projeté). Ce problème dual local associe un multiplicateur au risque empirique de chaque échantillon local, ainsi qu'un multiplicateur pour chaque contrainte d'égalité avec les nœuds voisins.

2. Les hyperplans sont échangés entre nœuds voisins.

3. Les multiplicateurs associés aux contraintes d'égalité sont mis à jour en suivant la direction de la déviation entre l'hyperplan local et les hyperplans des voisins. Cette dernière étape assure que le point fixe de MoM-DSVM correspond au consensus, c'est à dire à l'optimum du problème global.

Pour les problèmes convexes de manière générale, des travaux récents prouvent que les algorithmes décentralisés dérivés d'ADMM [40, 98, 128, 212] ont un taux de convergence linéaire

[123, 129, 131]. Notons que, dans ces preuves, la notion d'itération correspond à la résolution du problème local en chaque noeud suivi de l'échange entre nœuds. L'unité de temps rend ainsi compte uniquement des coûts de communication (nombre de passes d'échanges entre voisins), et non le coût de résolution des problèmes locaux. En pratique, les coûts de calcul locaux ont un effet important sur le temps réel d'exécution de ces méthodes duales. DisDCA (*Distributed Dual Coordinates Ascent* [257]) est une implémentation distribuée de SDCA [222] qui offre également une convergence linéaire mais plus rapide en pratique.

4.3.4 Approches primales

Contrairement au méthode décrites précédemment, les approches primales se concentrent uniquement sur l'objectif (4.10) en ne travaillant que sur les paramètres primaux du problème (*i.e.* l'hyperplan séparateur).

De nombreux travaux considèrent notamment l'implémentation de SGD en environnement distribué. Gadget SVM [118] propose ainsi une implémentation distribuée du solveur primal Pegasos [220], au moyen d'un protocole Gossip de type Push-Sum [145]. Chaque noeud calcule un gradient partiel sur son propre jeu de données, puis ces gradients sont sommés entre nœuds avant de mettre à jour en chaque noeud l'hyperplan en projetant le gradient global obtenu.

L'approche plus généraliste de [25] considère un algorithme d'approximation stochastique en deux étapes alternées :

$$\textbf{Étape locale :} \qquad \forall i,\ \tilde{\mathbf{w}}_i(t+1) = \mathbf{w}_i(t) + \eta_t Y_i(t) \tag{4.25}$$

$$\textbf{Étape Gossip :} \qquad \forall i,\ \mathbf{w}_i(t+1) = \sum_{j=1}^{N} \kappa_{ij}(t)\tilde{\mathbf{w}}_i(t+1) \tag{4.26}$$

$Y_i(t)$ est un processus aléatoire quelconque générant des vecteurs de $\mathbb{R}^D$, correspondant par exemple à l'évaluation d'un gradient stochastique pour le noeud i, et η_t est un pas d'apprentissage décroissant respectant (4.17). Sous ce régime, qui modélise donc une SGD additionnée d'une étape de moyennage entre nœuds, il est prouvé que le processus converge vers l'optimum recherché, mais à un taux sous-linéaire si $Y_i(t)$ est un simple gradient stochastique (comme c'est le cas pour une SGD centralisée à pas décroissant).

DDA (*Distributed Dual Averaging* [4, 77]) reprend l'approche DA [184] en profitant du caractère linéaire de la mise à jour de l'estimée duale (moyenne de tous les gradients précédents). La mise à jour distribuée de cette variable duale est confiée à un algorithme de consensus, la projection proximale pouvant être réalisée indépendamment en chaque noeud :

$$\forall i \in \{1 \dots N\}, \quad \begin{cases} \mathbf{z}_i(t+1) = \displaystyle\sum_{j}^{N} \mathbf{K}_{ji}(t)\mathbf{z}_j(t) - \nabla f_l^{(i)}(\mathbf{w}_i(t)) \\ \mathbf{w}_i(t+1) = \displaystyle\arg\min_{\mathbf{w}\in\mathbb{R}^D} \left[\mathbf{z}_i(t+1)^\top \mathbf{w} + \frac{1}{\eta_t}\psi(\mathbf{w})\right] \end{cases}, \tag{4.27}$$

où $\mathbf{K}(t)$ est une matrice de communication doublement stochastique et ψ une fonction proximale fortement convexe sur $\mathbb{R}^D$ localement calculable en tout noeud i. Il est également prouvé que DDA converge vers l'optimum à un taux sous-linéaire. Push-Sum DDA [236] est une extension décentralisée et asynchrone de DDA qui relâche la contrainte de double stochasticité de la matrice de communication $\mathbf{K}$ de DDA via un protocole de type Push-Sum similaire à l'algorithme

AGAVG du chapitre 1. Les coûts de communications réduits et l'asynchronisme du protocole favorise le passe à l'échelle sur de grands réseaux. Cependant, en tant qu'implémentation de DA, son taux de convergence reste sous-linéaire. Lorsque que l'on ne vise pas une précision très élevée (*i.e.*, atteignable par une SGD à pas constant), une SGD distribuée telle que Gadget SVM correctement paramétrée se révèle beaucoup plus rapide en pratique.

4.4 Short-Term Averaged Gradient (STAG)

Dans cette section, nous proposons une procédure d'optimisation convexe centralisée originale nommée STAG (*Short-Term Averaged Gradient* [93]). Il s'agit d'une méthode par descente de gradient moyennée "pure", dans le sens où elle ne fait pas intervenir de terme proximal. Comme SAG [208], elle se fonde sur l'intuition que moyenner les gradients précédents permet de stabiliser l'évolution du processus avec un pas constant, mais que seuls les plus récents gradients doivent être moyennés sous peine de perde la convergence linéaire rapide.

A la différence de SAG, dans STAG le nombre de gradients moyennés n'est pas nécessairement égal au nombre d'échantillons d'entraînement, mais devient un paramètre fixé selon le cas d'utilisation et les contraintes d'implantation de l'algorithme. En effet, une implémentation basique de SAG nécessite de maintenir une mémoire de n gradients de $\mathbb{R}^D$, entraînant un coût de stockage en $\mathcal{O}(nD)$. Bien que pour les problèmes structurés (gradients épars ou trivialement recalculables à la volée) on puisse réduire ce coût de stockage à $\mathcal{O}(n)$, dans le cas général il reste problématique de mémoriser un gradient par échantillon. Plus important, on sait que la plupart des problèmes de Machine Learning induisent par nature une grande redondance des gradients (celle ci profite notamment à SGD). Ainsi, à tout instant, le contenu de la mémoire des n gradients précédents peut être sans perte importante approximé par un sous-ensemble de capacité bornée. En particulier, il est tout à fait probable que le nombre de gradients qui optimise le compromis entre espace mémoire et stabilité de convergence dépende bien plus de propriétés intrinsèques au problème (densité des données, parcimonie et redondance des gradients, etc) que de propriétés extrinsèques (nombre d'échantillons n, dimension D, etc).

STAG propose donc de ne moyenner que les L gradients les plus récemment évalués en considérant un mécanisme de sélection des exemples d'apprentissage uniforme sans remise (stratégie par *shuffling* [34, 222]). Ainsi, la mémoire des L gradients précédents peut être implémentée par une file d'attente de type FIFO (*First In, First Out*) et l'on peut garantir que l'intervalle entre deux évaluations d'un même échantillon est borné par n avec une probabilité 1 (ce qui n'est pas le cas avec une sélection uniforme avec remise). La mise à jour de STAG est donc définie par

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \sum_{l=0}^{L-1} \mathbf{g}(t-l), \quad \mathbf{g}(t) = \nabla f_j(\mathbf{w}(t)), \tag{4.28}$$

avec j uniforme sans remise dans $\{1 \dots n\}$ et η un pas d'apprentissage constant.

Vitesse de convergence

Lorsque $L = 1$, on retrouve la mise à jour de SGD avec pas constant, qui converge exponentiellement, mais dont l'instabilité empêche la convergence sous une certaine précision. Lorsque $L = n$, on obtient la mise à jour de SAG, qui quand à elle convergence exponentiellement vers

l'optimum avec une précision arbitraire mais requiert le stockage de n gradients. Si dans ces deux cas particuliers la littérature fournit des preuves de convergence formelles, il n'existe à notre connaissance aucune analyse théorique des régimes intermédiaires où $1 < L < n$. Une hypothèse importante pour assurer la convergence linéaire exacte de SAG concerne notamment le caractère fini du jeu de données d'entraînement [67, 213]. La convergence linéaire est donc probablement perdue aussi bien dans les cas où le nombre d'échantillons est infini (*e.g.*, optimisation en ligne) que lorsque l'on découple le nombre de gradients moyennés du nombre d'échantillons d'entraînement. Toutefois, les schémas de preuve présentés dans la littérature ne permettent pas de prédire la perte induite par la réduction du nombre de gradients moyennés. Par ailleurs, il est mentionné dans [208] que malgré sa supériorité asymptotique, SAG exhibe des constantes de temps défavorables vis-à-vis de SGD. SGD restant très utilisée et efficace en pratique notamment grâce à des pas mixtes (constants dans un premier temps puis décroissants pour affiner les dernières étapes [32]), les régimes intermédiaires $1 < L < n$ posent un certain nombre de questions théoriques :

- Quand la convergence linéaire est-elle perdue ?
- Conserve-t-on une convergence exacte ?
- Si non, le rayon de convergence évolue-t-il graduellement ou brusquement lorsque L varie de n à 1 ?
- Les régimes intermédiaires apportent-ils un gain en vitesse de convergence non-asymptotique par des constant multiplicatives plus favorables ?
- Peut-on finalement décorréler le choix de L du nombre d'échantillons n et l'aligner sur une mesure statistique intrinsèque au problème ?

Ces questions apparaissent difficiles à aborder de manière formelle hors des cas évidents (*e.g.*, redondances triviales dans les données) où l'intuition suffit à traduire le problème dans un cadre connu. Par exemple, dans le cas où le nombre d'échantillons est en pratique illimité, pourvu que le processus sous-jacent génère des séquences de L échantillons identiques ou très proches, STAG peut être interprété comme équivalent à SAG sur un ensemble de L échantillons. Enfin, le lien avec Pegasos ne peut pas être ignoré, puisqu'on peut voir STAG comme une version de Pegasos avec des retards sur les gradients considérés à chaque itération.

Implémentation pratique

De nombreuses astuces permettent d'améliorer l'efficacité pratique de STAG dans une implémentation concrète. On note tout d'abord que (4.28) ne nécessitant le stockage et la sommation que des L derniers gradients, on peut utiliser mémoriser ces gradients dans une file d'attente de type FIFO de capacité L, en réécrivant (4.28) comme

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \mathbf{z}(t), \quad \mathbf{z}(t) = \mathbf{z}(t-1) + \mathbf{g}(t) - \mathbf{h}(t), \tag{4.29}$$

où $\mathbf{g}(t)$ désigne le gradient nouvellement évalué qui entre dans la file à l'instant t, et $\mathbf{h}(t)$ désigne le gradient qui quitte la file suite à l'entrée de $\mathbf{g}(t)$. Le coût d'itération de STAG se résume à un stockage dans la file et 3 additions dans $\mathbb{R}^D$. Notons qu'à la différence de SAG, la gestion de la

file peut être confiée à un buffer circulaire (accès mémoire contigus) tandis que SAG nécessite des accès aléatoires (moins efficients) aux gradients mémorisés.

Pour les problèmes où l'évaluation des gradients est très peu coûteuse, notamment lorsque les gradients sont proportionnels aux échantillons correspondants (*e.g.* SVM avec hinge loss), on peut réduire le coût de stockage de $\mathcal{O}(LD)$ à $\mathcal{O}(L)$ en ne stockant que l'indice des échantillons correspondants et en recalculant à la volée les deux gradients $\mathbf{g}(t)$ et $\mathbf{h}(t)$ à partir de leurs indices. Ceci requiert néanmoins une approximation des éventuels termes de régularisation dans le gradient moyenné (puisque $\mathbf{w}(t-L)$ n'est pas stocké donc plus accessible pour le calcul de $\mathbf{h}(t)$). Si $\psi(\cdot)$ et $\ell(\cdot)$ désignent respectivement les termes de régularisation et le terme risque empirique de la fonction objective globale, deux solutions peuvent être adoptées :

- Approximer $\mathbf{w}(t-L)$ par $\mathbf{w}(t)$ dans le calcul du gradient de ψ pour $\mathbf{h}(t)$:

$$\mathbf{h}(t) = \nabla\psi(\mathbf{w}(t)) + \nabla\ell(\mathbf{w}(t-L)) \tag{4.30}$$

- Utiliser une régularisation explicite [254]. On supprime ψ de la fonction objective lors du moyennage des gradients, et, similairement aux méthodes proximales, on ajoute un terme de pénalisation à la mise à jour de l'estimée :

$$\mathbf{w}(t+1) = \underset{\mathbf{w}\in\mathbb{R}^D}{\arg\min} \left[\mathbf{z}(t)^\top \mathbf{w} - \eta\psi(\mathbf{w})\right] \tag{4.31}$$

Dans la pratique, la régularisation explicite apporte les meilleurs résultats, en particulier lorsque le terme de régularisation a pour but d'introduire une parcimonie dans la solution (régularisation ℓ_1 ou $\ell_{p<1}$). En effet, le moyennage des gradients sur le terme de régularisation a tendance à contrer son effet structurant [254].

4.5 AGSTAG

Nous présentons dans cette section une extension de STAG aux environnements distribués nommée AGSTAG (*Asynchronous Gossip Short-Term Averaged Gradient*, Algorithme 8). AGSTAG permet donc de répondre au problème (4.23) exposé dans la section 4.3.1. AGSTAG s'appuie sur l'introduction du protocole AGAVG présenté dans le chapitre 1 pour moyenner des gradients partiels calculés en chacun des nœuds du réseau.

4.5.1 Principe de fonctionnement

Dans AGSTAG, chaque noeud i maintient une estimée locale $\mathbf{w}_i(t)$ et un gradient moyenné local $\mathbf{z}_i(t)$. Ce gradient $\mathbf{z}_i(t)$ est mis à jour par deux processus concurrents de manière asynchrone :

- La mise à jour (4.29) de STAG est effectuée localement à chaque évaluation d'un nouveau gradient local. A chaque mise à jour, le nouveau gradient entre dans une FIFO locale au noeud i et le gradient le plus ancien la quitte. Cette mise à jour est déclenchée soit à réception d'un nouvel échantillon (cas de l'estimation en ligne), soit itérativement en sélectionnant un échantillon local uniformément sans remise.

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) - \frac{\eta}{\omega_i}\mathbf{z}_i(t), \quad \mathbf{z}_i(t) = \mathbf{z}_i(t-1) + \mathbf{g}_i(t) - \mathbf{h}_i(t) \tag{4.32}$$

Algorithme 8 AGSTAG : Asynchronous Gossip Short-Term Average Gradient (en chaque noeud i)

Entrées : n_i fonctions de coût convexes $\{f_1, \ldots, f_{n_i}\}$ dont les gradients (ou sous-gradients) sont localement calculables en i.
Paramètres : (η_t) : suite de pas d'apprentissage.
L : capacité du buffer circulaire en nombre de gradients.
M : nombre de messages envoyés à chaque itération locale.

○ **Procédure émission**

$\mathbf{w} \leftarrow$ vecteur aléatoire de dimension Q
$\mathbf{z} \leftarrow \mathbf{0}_Q$; $\omega \leftarrow 0$
FIFO $\leftarrow$ buffer circulaire vide et de capacité L vecteurs de $\mathbb{R}^Q$
Boucle
  $\mathbf{g} \leftarrow \nabla f_k(\mathbf{w})$, où k tiré aléatoirement dans $\{1, \ldots, n_i\}$
  Si FIFO.taille = L **Alors** $\mathbf{h} \leftarrow$ FIFO.pull()
  Sinon $\mathbf{h} \leftarrow \mathbf{0}_Q$; $\omega \leftarrow \omega + 1$
  Fin Si
  FIFO.push($\mathbf{g}$)
  $\mathbf{z} \leftarrow \mathbf{z} + \mathbf{g} - \mathbf{h}$
  Pour m **de** 1 **à** M **faire**
    $\mathbf{z} \leftarrow \mathbf{z}/2$; $\omega \leftarrow \omega/2$
    $j \leftarrow$ noeud voisin aléatoire
    Envoyer $(\mathbf{z}, \omega)$ à j
  Fin Pour
  $\mathbf{w} \leftarrow \mathbf{w} - (\eta_t/\omega)\mathbf{z}$
Fin Boucle

○ **Procédure réception**

Boucle
  Attendre réception d'un message $(\mathbf{z}', \omega')$
  $\mathbf{z} \leftarrow \mathbf{z} + \mathbf{z}'$; $\omega \leftarrow \omega + \omega'$
Fin Boucle

- Les $\mathbf{z}_i(t)$ sont moyennés entre nœuds par un protocole de consensus asynchrone :

$$\mathbf{z}_i(t+1) = \sum_j^N \mathbf{K}_{ji}(t)\mathbf{z}_j(t), \quad \omega_i(t+1) = \sum_j^N \mathbf{K}_{ji}(t)\omega_j(t), \tag{4.33}$$

Ici, $\mathbf{K}_{ji}(t)$ une matrice de communication telle que $\mathbf{K}\mathbf{1} = \mathbf{1}$ et $\omega_i(t)$ est un poids associé à $\mathbf{z}_i(t)$. Ce poids est initialisé à 0 en début d'algorithme, et est incrémenté chaque fois qu'un gradient entre dans la file et décrémenté quand un gradient la quitte, de sorte à ce que $\sum_j^N \omega_j(t)$ compte toujours le nombre de gradients entrés dans les files du réseau. Cela permet notamment de maîtriser l'influence du pas d'apprentissage η dans les premières itérations. Comme détaillé dans les chapitre 1, lorsque $\mathbf{K}_{ij}(t) \neq 0$ pour un couple de nœuds (i, j) donné, il y a émission d'un message (de taille D) du noeud i vers le noeud j.

Bien qu'il faille s'assurer que les deux mises à jour concurrentes se fassent localement de manière atomique, les deux processus fonctionnent de manière désynchronisée, n'impliquant ainsi aucune phase d'attente entre nœuds.

Notons que dans le cas idéal où les gradients sont évalués n fois plus rapidement que la fréquence de mise à jour des estimées et que les files locales ont une capacité égale au nombre d'échantillons locaux n_i, le gradient moyenné devient équivalent au gradient complet de FG, à la désynchronisation près. Ainsi, en contrôlant les fréquences respectives de mise à jour de $\mathbf{z}_i(t)$ et de $\mathbf{w}_i(t)$, on peut se placer graduellement entre un schéma STAG pur (*i.e.*, à chaque évaluation de gradient succède une mise à jour de l'estimée primale) et un schéma de type "FG asynchrone" où à chaque mise à jour primale est faite à la suite de n évaluations de gradients. Dans la pratique, et suivant les remarques de [33] concernant l'avantage computationnel de de SGD, nous préférons conserver dans AGSTAG une séquentialité entre les mises à jour locales de $\mathbf{z}$ et de $\mathbf{w}$, afin de tirer directement profit de *chaque* nouveau gradient dans l'évolution de $\mathbf{w}$.

Remarquons que du fait de la dispersion des données sur le réseau, on a typiquement $n_i \simeq n/N$. La capacité des files locales peut donc être réduite d'un facteur N sans impacter la convergence. Conséquemment, l'empreinte mémoire d'AGSTAG décroît en proportion inverse de la taille du réseau. Par exemple, si le jeu de données est reparti exactement sur n noeuds (*i.e.*, un échantillon par noeud), chaque noeud n'a besoin de conserver qu'un seul gradient (le dernier évalué) pour obtenir un algorithme distribué équivalent à SAG, pourvu que le consensus sur les $\mathbf{z}_i(t)$ soit atteint à chaque étape. La précision de ce consensus est contrôlé par l'hyper-paramètre M, qui fixe le nombre de messages envoyés entre chaque évaluation de gradient et mise à jour primale.

4.5.2 Outils d'analyse théorique

Il apparaît difficile d'établir une borne théorique sur le taux de convergence d'AGSTAG, pour les mêmes raisons que STAG d'une part (voir section précédente) et d'autre part car la communauté manque encore de résultats et de méthodologies formelles pour analyser les méthodes par gradients moyennés asynchrones et décentralisées. Néanmoins, nous donnons ici quelques pistes et outils d'analyse de la littérature qui semblent pertinent pour établir la borne recherchée.

Premièrement, les stratégies de Descente de Gradient Stochastique distribuées asynchrones (non moyennées) ont fait l'objet d'une étude théorique poussée, notamment dans [3]. Un des résultats importants établit qu'un retard aléatoire (stationnaire dans [3]) entre l'évaluation des gradients et la mise à jour de l'estimée primale n'impacte pas significativement la convergence du processus. Ainsi, toute méthode où le processus d'évaluation des gradients et le processus de mise à jour de $\mathbf{w}$ sont désynchronisés converge de manière quasi-équivalente à son homologue centralisé séquentiel (*i.e.*, en deux passes évaluation du gradient - mise à jour de l'estimée). Ce résultat se généralise au cas ou de multiples processus d'évaluation de gradients sont exécutés en parallèles et communiquent de manière asynchrone avec un processus maître qui met à jour une unique estimée (schéma *workers-master*). Malheureusement, le cas où plusieurs processus de mise à jour de l'estimée primale fonctionnent concurremment n'est pas étudié dans [3]. Il serait nécessaire d'étendre l'analyse à ce cas où les estimées primales et duales sont toutes deux multiples pour prendre en compte l'optimisation décentralisée qui nous intéresse ici.

Deuxièmement, les travaux de [4, 77] sur DDA, étendus par [236] au cas décentralisé asynchrone, formulent une bornes supérieure sur la sous-optimalité primale en chaque noeud du réseau par une somme de termes où la nature "descente de gradient" et la nature "distribuée" du

processus sont séparées :

$$
\begin{aligned}
& f(\hat{\mathbf{w}}_i(t)) - f(\mathbf{w}^\star) \leq A + B \\
& \qquad A = \frac{1}{t\eta_t}\psi(\mathbf{w}^\star) + \frac{L^2}{2t}\sum_{\tau=0}^{t-1}\eta_\tau \\
& \text{où} \qquad B = \frac{2L}{Nt}\sum_{\tau}^{t}\sum_{j}^{N}\eta_\tau\|\bar{\mathbf{z}}(\tau) - \mathbf{z}_j(\tau)\|_* + \frac{L}{t}\sum_{\tau}^{t}\eta_t\|\bar{\mathbf{z}}(\tau) - \mathbf{z}_i(\tau)\|_*
\end{aligned}
\tag{4.34}
$$

Ici, $\hat{\mathbf{w}}_i(t) = \frac{1}{t}\sum_\tau^t \mathbf{w}_i(\tau)$, $\bar{\mathbf{z}}(t) \equiv \frac{1}{N}\sum_i^N \mathbf{z}_i(t)$ est la moyenne des estimées duales à travers le réseau, L est la constante de Lipschitz de f et $\|\cdot\|_*$ désigne la norme duale associée à $\|\cdot\|$. Le terme A est communément rencontré dans les analyses de méthodes basées gradient centralisées, et représente donc la dimension "optimisation par gradient stochastique" du processus. Le terme B correspond au désaccord entre nœuds, en l'occurrence l'erreur entre les gradients moyennés locaux et leur moyenne globale, et représente donc la dimension "consensus distribué" du processus. La séparation de ces deux termes facilite grandement l'analyse, puisque les outils habituels d'optimisation par gradient et de consensus distribué peuvent être utilisés tels quels. Cette propriété permet par exemple à [236] de proposer un schéma de preuve simple et concis pour le cas asynchrone de PushSum-DDA. Toutefois, la réécriture (4.34) n'est valide que pour les méthodes proximales, et non pour les méthodes par descente de gradient pures.

4.5.3 AGSVM

Dans le cas spécifique de la résolution de SVM, l'adaptation d'AGSTAG conduit à un algorithme décentralisé et asynchrone efficient pour l'optimisation primale d'une SVM linéaire (*cf* Algorithme 9). En adoptant un risque empirique de type *hinge loss*, le gradient associé à un couple échantillon-label local $(\mathbf{x}, y)$ en un noeud i se calcule comme suit :

$$
\mathbf{g}(t) \leftarrow \begin{cases} -\lambda\mathbf{w}_i(t) & \text{si} \quad y\mathbf{x}^\top\mathbf{w}_i(t) > 1 \\ -\lambda\mathbf{w}_i(t) + y\mathbf{x} & \text{sinon} \end{cases}
\tag{4.35}
$$

Or, les résultats expérimentaux présentés dans la section suivante révèlent en pratique deux propriétés importantes :

- La convergence de STAG sur un problème régularisé est significativement améliorée par l'usage d'une régularisation explicite. La régularisation explicite consiste à ignorer le terme de régularisation dans le gradient moyenné et de ne le considérer qu'au moment de la mise à jour. Ainsi, si la fonction de coût associée à chaque échantillon i a la forme

$$
f_i(\mathbf{w}) = \psi(\mathbf{w}) + \ell_i(\mathbf{w}),
\tag{4.36}
$$

alors le schéma d'optimisation devient

$$
\mathbf{z}(t) = \sum_{l=0}^{L-1}\mathbf{g}(t-l), \quad \text{où} \quad \mathbf{g}(t) = \nabla\ell_i(\mathbf{w}(t))
\tag{4.37}
$$

$$
\mathbf{w}(t+1) = \mathbf{w}(t) - \eta_t\big(\nabla\psi(\mathbf{w}(t)) + \mathbf{z}(t)\big)
\tag{4.38}
$$

Algorithme 9 AGSVM : Asynchronous Gossip linear SVM solver (en chaque nœud i)

Entrées : Jeu d'échantillons $\mathbf{X} \in \mathbb{R}^{n_i \times D}$
Labels associés $\mathbf{y} \in \{-1, 1\}^{n_i}$
Paramètres : λ : coefficient du terme de régularisation ℓ_2
(η_t) : suite de pas d'apprentissage.
L : age maximum des gradients moyennés.
M : nombre de messages envoyés à chaque itération locale.

○ **Procédure émission**

$\mathbf{w} \leftarrow$ vecteur aléatoire de dimension Q
$\mathbf{z} \leftarrow \mathbf{0}_Q$; $\omega \leftarrow 0$
FIFO $\leftarrow$ file de couples $(t, k) \in \mathbb{N}^2$ vide de capacité maximum L.
$\forall i \in \{1, \dots, n_i\}$, $\mathbf{x}_i \leftarrow y_i \mathbf{x}_i$
$t \leftarrow 0$
Boucle
  $k \leftarrow$ indice aléatoire dans $\{1, \dots, n_i\}$
  Si $\mathbf{w}^\top \mathbf{x}_k \leq 1$ **Alors**
    FIFO.push((t, k)) ; $\mathbf{z} \leftarrow \mathbf{z} + \mathbf{x}_k$
  Fin Si
  Si $t -$ FIFO.front().$t = L$ **Alors**
    $l \leftarrow$ FIFO.front().k ; $\mathbf{z} \leftarrow \mathbf{z} - \mathbf{x}_l$; FIFO.pull()
  Fin Si
  Si $t < L$ **Alors** $\omega \leftarrow \omega + 1$ **Fin Si**
  Pour m **de** 1 **à** M **faire**
    $\mathbf{z} \leftarrow \mathbf{z}/2$; $\omega \leftarrow \omega/2$
    $j \leftarrow$ nœud voisin aléatoire
    Envoyer $(\mathbf{z}, \omega)$ à j
  Fin Pour
  $\mathbf{w} \leftarrow \mathbf{w} - \eta_t\big(\lambda \mathbf{w} + (1/\omega)\mathbf{z}\big)$
  $t \leftarrow t + 1$
Fin Boucle

○ **Procédure réception**

Boucle
  Attendre réception d'un message $(\mathbf{z}', \omega')$
  $\mathbf{z} \leftarrow \mathbf{z} + \mathbf{z}'$; $\omega \leftarrow \omega + \omega'$
Fin Boucle

Notons que cette propriété s'étend naturellement à SAG. Par ailleurs, l'observation de cette propriété pour Dual Averaging est notamment à l'origine des méthodes Regularized Dual Averaging (voir [254]).

- Lorsque que l'on utilise une régularisation explicite, la stratégie de sélection sans remise apporte un gain de performance appréciable. Cette observation va dans le sens des propriétés empiriques relevées sur SGD [34] et SDCA [222].

En se basant sur ces deux remarques, la résolution d'une SVM primale par STAG est gran-

dement simplifiée. En effet, puisque dans ce cas

$$\mathbf{g}(t) = \nabla \ell_i(\mathbf{w}(t)) = \begin{cases} y_i \mathbf{x}_i & \text{si} \quad y_i \mathbf{x}_i^\top \mathbf{w}(t) \leq 1 \\ 0 & \text{sinon} \end{cases} , \tag{4.39}$$

la file des gradients moyennés peut être stockée de manière éparse (*i.e.*, seuls les échantillons violant la contrainte de marge sont explicitement mémorisés). De plus, en pré-multipliant tous les échantillons $\mathbf{x}_i$ par leur label y_i en début d'algorithme, le calcul de $\mathbf{g}(t)$ devient trivial une fois le test (4.39) effectué. On peut donc se contenter de stocker dans la file l'indice des échantillons violant la contrainte de marge sur les L dernières évaluations. L'empreinte mémoire d'AGSVM est donc la plupart du temps largement inférieure à L et en $\mathcal{O}(L)$ dans le pire cas.

Du point de vue calculatoire, le coût de calcul reste identique bien que le gradient quittant la file ne soit pas disponible explicitement. En effet, il suffit de vérifier si l'instant t' auquel le plus ancien gradient de la file avait été évalué satisfait $t - t' \geq L$. Si c'est le cas, il est retiré de la file et retranché à $\mathbf{z}(t)$.

En environnement distribué, l'avantage de la régularisation explicite est particulièrement marqué. En effet, puisque seuls les échantillons produisant une erreur empirique non-nulle influencent l'estimée locale $\mathbf{z}(t)$, la convergence du réseau vers un gradient moyen consensus à travers le réseau n'est perturbée que lorsqu'une erreur est rencontrée par un noeud. Sous réserve que celles-ci soient suffisamment rares (ce qui est souvent le cas en pratique, *i.e.*, le nombre de vecteurs supports est faible comparé au nombre d'échantillons), le consensus est beaucoup moins perturbé que lorsque le terme de régularisation est intégré au gradient moyenné (celui-ci évoluant constamment par définition). La stabilité du consensus et par conséquent celle du processus d'optimisation global sont ainsi augmentées. Comme nous le verrons dans la section suivante, cela nous permet d'utiliser des pas d'apprentissage bien plus grands et donc d'accélérer d'autant la convergence de l'algorithme.

4.6 Résultats Experimentaux

Cette section est dédiée à l'évaluation expérimentale de l'algorithme AGSTAG proposé. Nous considérons la classification binaire de chiffres manuscrits par SVM linéaire sur le jeu de données MNIST [156] ($D = 784, n = 60000$), selon un protocole similaire à [33]. Nous évaluons la capacité du classificateur à séparer le chiffre '1' des autres chiffres (classification *one-versus-all*). La constante de régularisation ℓ_2 est fixée à $\lambda = 10^{-4}$, et le biais est intégré à l'hyperplan $\mathbf{w}$ en ajoutant une coordonnée constante égale à 1 à tous les échantillons, de sorte à ce que la fonction de décision soit simplement $\mathbf{w}^\top \mathbf{x}$, tel que décrit dans la section 4.1.1.

4.6.1 STAG

Nous évaluons tout d'abord le schéma d'optimisation centralisé STAG. Nous cherchons ici à mettre en évidence le rôle du pas η, de la taille de la file L (nombre de gradients moyennés) et de la précision recherchée ε sur la solution.

En premier lieu, la figure 4.1 montre que le régime de fonctionnement le plus favorable à STAG combine la régularisation explicite et la sélection uniforme sans remise. Nous constatons que l'intégration du régularisateur dans le gradient moyenné peut conduire à la divergence de

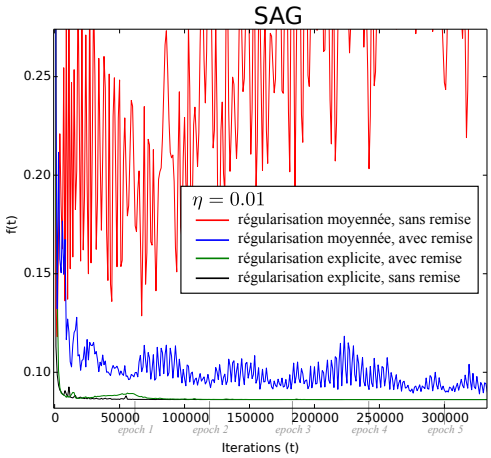

FIGURE 4.1 – Comparaison de la convergence de STAG pour différents régimes de fonctionnement ($L = n$).

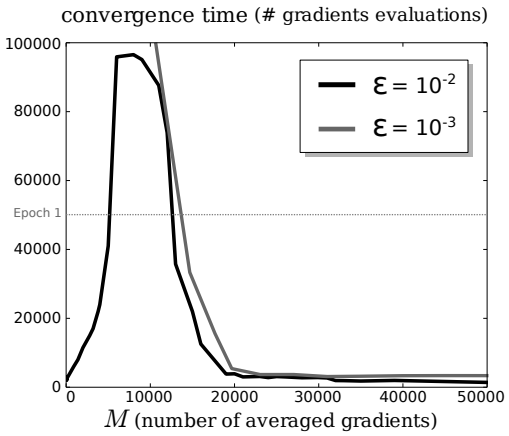

FIGURE 4.2 – Temps de convergence de STAG en nombre d'évaluations de gradient en fonction de la taille de file L

l'algorithme. Nous nous concentrons donc sur le régime où le régularisateur n'est considéré que lors de la mise à jour de l'hyperplan.

La figure 4.2 montre le temps de convergence de STAG en terme de nombre d'évaluations de gradients nécessaires pour atteindre une sous-optimalité ε sur l'ensemble d'entraînement, ceci pour différentes tailles de file L. Pour chaque valeur de L, le meilleur pas constant η a été sélectionné. Rappelons que les cas particuliers $L = 1$ et $L = 60000$ correspondent respectivement à une SGD à pas constant et à SAG. Lorsque l'on cherche une précision élevée ($\varepsilon \leq 10^{-3}$), on observe qu'une capacité de file L trop faible (en dessous de 10000) empêche la convergence du processus, qui finit par osciller dans un rayon qui dépasse 10^{-3}. Pour une précision plus souple, SGD ($L = 1$) converge très rapidement. Par contraste, SAG ($L = n = 60000$) permet d'atteindre une précision élevée en un temps réduit. Dans les cas intermédiaires, remarquons que STAG bénéficie d'un temps de convergence correct quand $L \simeq n/2$.

Toutefois, nous constatons que la convergence de STAG est très sensible au choix du pas d'apprentissage η. Même lorsque le pas est faible, un comportement d'oscillation instable peut apparaître selon l'ordre dans lequel sont sélectionnés les exemples d'apprentissage. Dans certaines situations, ces oscillations s'entretiennent et s'amplifient, conduisant à la divergence de l'algorithme. Il semble difficile de déterminer précisément les conditions de cette divergence. Il est possible que celle-ci provienne du non-remplacement des gradients associés à un même exemple dans la file, cet exemple pouvant alors doublement contribuer au gradient moyen de manière tantôt contradictoire, tantôt auto-renforcée. Ceci pourrait expliquer le phénomène d'oscillations auto-entretenues observé.

En pratique, on s'intéresse rarement à la convergence exacte de la fonction de coût sur l'ensemble d'apprentissage. On souhaite en réalité minimiser l'erreur de classification sur un ensemble distinct d'échantillons non-labellisés (base de test). Une erreur nulle sur la base d'entraînement ne traduit pas forcément un erreur nulle sur la base de test (sur-apprentissage). Par ailleurs, on dispose généralement d'un budget de temps limité qui ne permet pas une convergence complète de l'apprentissage. L'erreur de classification sur la base de test obtenue après t évaluations de gradient constitue alors une mesure de performance plus pertinente. Ici, nous

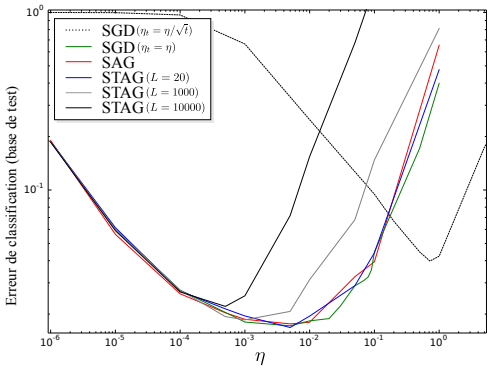

FIGURE 4.3 – Erreur de test obtenue après 5 époques pour différents pas d'apprentissage η.

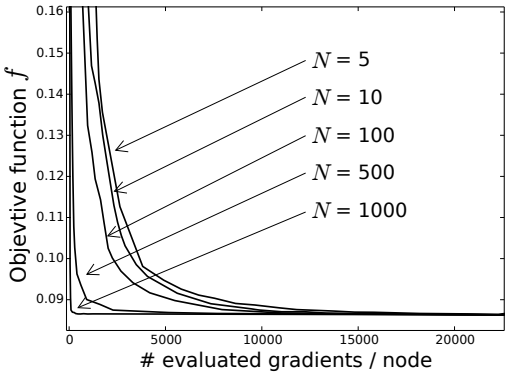

FIGURE 4.4 – Convergence d'AGSTAG pour diverses tailles de réseau N.

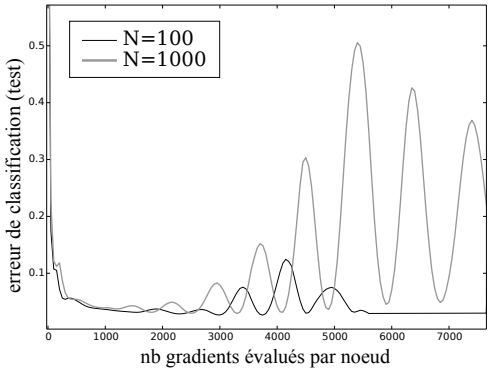

FIGURE 4.5 – AGSTAG amplifie l'instabilité de STAG lorsque la taille du réseau augmente (ici $L = 100$).

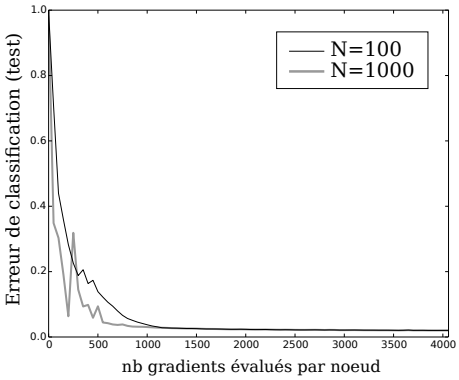

FIGURE 4.6 – L'instabilité est supprimée par le moyennage conjoint des gradients et des hyperplans locaux (ici, $L = 100$).

fixons $t = 5n$ (5 époques). La figure 4.3 évalue l'erreur de test atteinte par STAG pour diverses valeurs de η et L. Contre-intuitivement, on constate que les meilleurs résultats sont atteints par une simple SGD avec un pas fixe de 0.01. Le moyennage des gradients (SAG et STAG) semble n'apporter aucun avantage pratique lorsque le budget de temps est limité à 5 époques. Par ailleurs, l'utilisation d'un pas décroissant, qui en théorie garantit une convergence exacte de la fonction de coût, s'avère inutile dans ces conditions.

4.6.2 AGSTAG

Nous évaluons à présent l'extension distribuée AGSTAG. Sauf mention spécifique, le réseau est supposé complètement connecté et les échantillons sont uniformément distribués sur les nœuds. Le nombre de messages entre deux mises à jour locales M est fixé à 10.

La figure 4.4 montre la convergence d'AGSTAG pour différentes tailles de réseau ($N \in \{5, 10, 100, 500, 1000\}$, évaluée en terme de valeur de la fonction de coût d'apprentissage en fonction du nombre de gradients évalués par noeud. On observe que le nombre d'évaluations de gradient par noeud nécessaires pour converger diminue lorsque la taille du réseau augmente. Ainsi, la distribution du calcul sur un plus grand nombre d'unité apporte un gain significatif en

temps de convergence. Par ailleurs, notons que des réseaux plus larges permettent l'usage de pas d'apprentissage plus grands.

Moyennage des estimées

La sensibilité du schéma STAG au choix du pas d'apprentissage reste problématique dans AGSTAG. Cette sensibilité est aggravée par la distribution du processus sur un grand nombre de nœuds car plus le réseau est large plus les hyperplans locaux sont mis à jour avec des gradients qui leur sont incohérents. La figure 4.5 illustre ce phénomène en montrant que si AGSTAG converge pour $\eta = 0.01$ et $L = 100$ sur 10 nœuds, un pas identique conduit à un divergence sur 1000 nœuds.

Pour encourager la stabilisation, nous investiguons donc le moyennage conjoint des gradients $\mathbf{z}$ et des paramètres des modèles locaux $\mathbf{w}$ entre nœuds. Ceci a pour but de forcer le consensus sur les hyperplans locaux, de sorte à ce que ces derniers n'évoluent pas de manière incohérente.

La figure 4.6 montre clairement la stabilisation apportée par ce moyennage des modèles locaux, pour $\eta = 0.01$ et $L = 100$. On obtient ainsi un algorithme distribué à la fois rapide et stable. Néanmoins, dans notre contexte de budget de temps limité, nous n'avons pas observé d'avantage particulier à moyenner les gradients en comparaison à une SGD standard distribuée (*i.e*, $L = 1$ et pas de moyennage des gradients entre nœuds). Nous concluons donc que l'apport du moyennage des hyperplans et la distribution sur plusieurs nœuds des évaluations de gradients suffisent seules à apporter un gain significatif, le moyennage des gradients n'apportant que plus d'instabilité dans les premières itérations. Notons que ceci peut être du aux spécificité du jeu de données MNIST, et ne représente pas nécessairement une propriété générale du schéma proposé. Néanmoins, nous verrons dans le chapitre suivant que cette observation s'étend à la classification de signatures d'images.

4.7 Conclusion

Dans ce chapitre, nous avons présenté deux algorithmes originaux :

1. Un schéma d'optimisation convexe nommé STAG (Short-Term Averaged Gradient) qui s'appuie sur les résultats récents en matière de descente de gradients moyennés stochastique et y adjoint une dimension court-terme en limitant le nombre de gradients moyennés aux L derniers gradients évalués.

2. Une extension décentralisée et asynchrone de STAG nommée AGSTAG (Asynchronous Gossip Short-Term Averaged Gradient), qui utilise le protocole Gossip asynchrone décrit dans le chapitre 1 pour permettre l'optimisation convexe distribuée dans de larges réseaux potentiellement dynamiques.

En tant que méthodes primales, STAG et AGSTAG peuvent être utilisés dans des scénarios d'optimisation en ligne où les échantillons sont capturés en direct par les nœuds, sans nécessité de stocker plus de L gradients. L étant un paramètre fixé par l'utilisateur, il est découplé des propriétés extrinsèques du problème, à l'inverse de certaines méthodes de la littérature (*e.g.*, nombre d'échantillons d'entraînement pour SAG). On peut ainsi réduire l'empreinte mémoire de l'algorithme tout en conservant une vitesse de convergence raisonnable, en particulier lorsque

le problème possède une structure adaptée (redondance, parcimonie, etc). Pour les problèmes de classification type SVM linéaire, l'implémentation d'AGSTAG est particulièrement simple et efficace. La simplicité des opérations impliquées rend par exemple son implémentation matérielle sur FPGA ou Réseaux sur Puce (*Network-on-Chip*, NoC) accessible.

Néanmoins, notre étude expérimentale n'a pas pu mettre en évidence le gain apporté par le moyennage des gradients dans un scénario où le budget de temps est limité à un certain nombre d'évaluations de gradients fixé *a priori*. Une simple SGD décentralisée asynchrone ($L = 1$) semble ainsi plus performante dans ce cas.

Demeure alors un certain nombre de questions concernant ces deux algorithmes :

- Peut-on établir une preuve de convergence générale pour les méthodes de gradient moyenné à court terme, faisant apparaître de manière explicite l'influence de la taille L de la fenêtre de moyennage choisie ?
- Peut-on reformuler la sous-optimalité d'AGSTAG à un instant t comme une somme de deux termes, le premier isolant la nature "descente de gradient" de STAG et le deuxième caractérisant l'erreur d'estimation du consensus, c'est à dire l'influence du réseau ?
- Quelles propriétés statistiques du processus de génération des échantillons peut assister l'utilisateur dans la détermination de la taille de file L permettant le meilleur compromis entre coût mémoire et coût calculatoire ?
- Peut-on quantifier formellement la contribution de la stratégie de sélection sans remise comparée à la sélection uniforme avec remise utilisée habituellement ?

CHAPITRE 5

Application à l'indexation multimédia par le contenu

Dans les chapitres précédents, nous avons à dessein considéré les données d'entrées comme une matrice d'échantillons quelconque. Notamment, nous n'avons posé aucune hypothèse de structure sur les vecteurs d'entrée et formulé des résultats théoriques généraux. Dans ce chapitre, nous étudions le comportement des méthodes présentées dans les chapitres 2,3 et 4 sur des problèmes concrets de recherche par le contenu (*Content-Based Retrieval*, CBR).

La recherche par le contenu étend les concepts des moteurs de recherches textuels aux documents non textuels (images [164], vidéos [126], son [50], modèles 3D [232], *etc*). Si la maturité et les performance des moteurs de recherche textuels permettent aujourd'hui à des milliards d'utilisateurs d'accéder à des documents textuels au sein de collections extrêmement larges et variées, la recherche de contenus multimédias représente quant à elle un défi scientifique de premier plan. En effet, une part grandissante des documents disponibles sur les réseaux comprend des données non textuelles pour lesquelles on ne dispose d'aucune métadonnée structurée associée. Ces documents requièrent alors des méthodes de recherche spécifiques faisant intervenir directement leur contenu visuel, sonore, etc. Contrairement aux méthodes textuelles, aucune méthode de recherche par le contenu actuelle ne fournit une précision de recherche suffisante à des coûts de calcul et de stockage acceptables pour permettre une recherche à l'échelle du web. Par exemple, en recherche d'images, les techniques les plus performantes restent limitées à quelques millions de documents quand on en recense plusieurs dizaines de milliards sur l'Internet.

Tout comme leurs homologues textuels, les systèmes de recherche par le contenu s'appuient sur la comparaison entre documents. Réalisée via une *mesure de similarité*, cette comparaison permet de répondre à des tâches très variées, parmi lesquelles :

- La recherche par similarité, qui consiste à trier un ensemble de documents par ordre décroissant de similarité avec un document requête formulé par l'utilisateur. L'objectif est alors de retourner en priorité les documents qui ressemblent visuellement le plus à la requête. Cette tâche est généralement non-supervisée, dans le sens où ni l'utilisateur ni le concepteur du système ne fournit d'information autre que les documents bruts.

- L'annotation automatique, qui consiste à associer un ensemble de labels (moyen de transport, paysage, discours politique, *etc*) à chaque document, à partir d'un ensemble de document d'entraînement préalablement labellisés manuellement. L'objectif est ici de cor-

rectement représenter le concept associé à chaque label et de déterminer pertinemment sa présence dans les documents non-labellisés.

- La recherche interactive, où le système propose un ensemble de documents à l'utilisateur, qui indique à son tour au système les résultats qui correspondent à ce qu'il recherche et ceux qui n'y correspondent pas. Le système raffine alors ses résultats en tenant compte du retour de l'utilisateur, propose de nouveaux résultats, et ainsi de suite, jusqu'à converger vers le concept recherché.

Indexation par le contenu

Pour comparer le contenu de deux documents, un système de recherche par le contenu en extrait un ensemble de descripteurs, qui isolent des caractéristiques visuelles, auditives, textuelles, etc, à la fois représentatives et discriminantes. Dans l'idéal, on cherche ainsi à ce que ces descripteurs soient invariants aux perturbations qui empêchent l'isolation du "concept" recherché (changement de point de vue, de fond, de conditions d'éclairage, etc). Les méthodes de description globales extraient un unique vecteur de caractéristiques par document (histogramme de couleurs, décomposition sur une base d'ondelettes, *etc*). La comparaison entre documents est alors directement réalisée via une certaine métrique vectorielle sur leurs descripteurs (*e.g.*, χ^2). Les méthodes locales produisent pour chaque document un ensemble de descripteurs correspondant à des régions spatiales et/ou temporelles locales (on parle de *sac* de descripteurs). Ces méthodes sont beaucoup plus précises que les méthodes globales, mais elles exigent de comparer des sacs de descripteurs. Or le coût computationnel élevé de cette comparaison la rend inexploitable lorsque le nombre de documents à comparer est très grand.

Les techniques modernes recourent donc à une étape hors-ligne dite d'*indexation*, par laquelle chaque sac de descripteurs est converti en un unique vecteur appelé *signature*. L'ensemble des signatures est stocké sous la forme d'un *index*. Les documents sont ainsi directement comparés via une métrique vectorielle sur leur signature. On épargne au passage le coût d'extraction et de stockage des descripteurs qui est effectué une seule fois hors-ligne.

Pour être performante, une technique d'indexation multimédia doit produire des signatures rapides à calculer et à comparer pour un coût de stockage minimal. On cherche donc des signatures de faible dimension comparable par une métrique simple (*e.g.*, linéaire), qui offrent le meilleur pouvoir discriminant vis-à-vis de la tâche de recherche considérée.

Ces dernières années, un nombre croissant de méthodes d'indexation font ainsi appel à des techniques d'apprentissage statistique pour produire des encodeurs de signatures discriminants et invariants aux perturbations [183]. L'espace des descripteurs est modélisé par une densité de probabilité paramétrique (*e.g.*, GMM) ou par catégorisation (*e.g.*, K-means). Les descripteurs projetés sur ce modèle sont compressés via des techniques de réduction de dimension (*e.g.*, PCA). Enfin, les signatures de faible dimension obtenues sont classifiées de manière supervisée (*e.g.*, par une SVM) lorsque la tâche le requiert.

On constate ainsi le lien étroit qu'entretiennent les problématiques d'indexation multimédia actuelles et les questions d'apprentissage statistique large échelle.

Distribution des contenus, passage à l'échelle et confidentialité.

Les techniques d'indexation par le contenu supposent classiquement que l'ensemble des documents soit stockés en un unique site et traités par une même machine. Les algorithmes d'apprentissage impliqués peuvent alors être directement exploités sous leur forme centralisée. Malheu-

reusement, la quantité d'information multimédia disponible sur les réseaux augmente bien plus rapidement que la capacité de stockage et de calcul des machines individuelles. Typiquement, il faut plusieurs semaines à un cœur de calcul séquentiel équipé d'une centaine de giga-octets de mémoire vive pour traiter un million d'images en saturant sa mémoire. Le nombre de cœurs d'une machine étant limité à une centaine, même une implémentation parallélisée échouera à indexer une collection à l'échelle du web en un temps raisonnable. Aucune méthode centralisée ne permet ainsi d'indexer le flux continu de nouveaux contenus disponibles (*e.g.*, en 2015, 300 heures de vidéo sont mises en ligne chaque minute sur Youtube[1], l'équivalent de 30 millions d'images distinctes).

En réalité, ces immenses masses de contenu proviennent d'une multitude de sources plongées dans des réseaux grand public (Internet, réseaux pair-à-pair (P2P), réseaux mobiles...). De l'appareil mobile à l'imageur satellitaire en passant par le serveur de média à la demande, chaque source bénéficie souvent de la disponibilité et des capacités de calcul et de stockage suffisantes pour indexer indépendamment son propre contenu.

Afin de tirer parti de ces capacités et de cette organisation particulière des données, les techniques de recherche par le contenu distribuées (D-CBR) [57, 148, 171, 180, 201] visent à répartir le processus d'indexation sur de multiples unités mises en réseau. Le but ultime est alors que l'indexation soit réalisée par les sources elles-mêmes, au plus près des entités qui créent ou capturent les données brutes.

Un système d'indexation distribué offre ainsi trois avantages principaux :

Passage à l'échelle. Malgré l'explosion de la quantité de données disponibles à l'échelle mondiale, le débit de données offert par une source individuelle est lui systématiquement borné, souvent connu et commensurable. Le passage à l'échelle est alors assuré par la mise en réseau des processus d'indexation et non par l'augmentation de l'efficience des méthodes ou des performances des machines.

Prise en compte de la nature répartie de l'information. L'information étant par nature distribué à son origine, on évite une étape coûteuse de rapatriement des données en un site unique (*e.g.*, datacenter), avec les difficultés que cela implique en terme de débits, de capacité de stockage, d'accès aux données et de maintenance du système [112].

Préservation de la confidentialité. Le stockage d'informations massives en un site unique pose systématiquement des problèmes de protection de la vie privée et de gestion des données confidentielles [5]. En pratique, de telles pratiques, nécessitent la négociation de transferts de droits ou l'établissement de clauses de confidentialité et de restriction des usages. A l'inverse, il n'existe à l'heure actuelle quasiment pas de réglementation en matière de partage d'information agrégée telles qu'un index ou une signature (la force cryptographique de l'encodage peut toutefois être remis en question à long terme). Dans le cas où les documents à indexer contiennent des données sensibles non transférables, l'adoption d'une approche distribuée est alors inévitable [245].

De l'indexation distribuée à l'apprentissage décentralisé asynchrone

Pour indexer une collection de documents multimédia, les méthodes d'indexation les plus performantes construisent un modèle de leur espace de description. Une fois ce modèle construit,

[1] https ://www.youtube.com/yt/press/statistics.html

la conversion des documents en signatures est clairement une étape séparable, dans le sens où chaque document peut être projeté sur le modèle sans faire intervenir les autres documents. Dès lors que chaque machine possède une instance du modèle, elle peut donc indexer son propre contenu sans échanger d'information avec les autres machines. Le défi clé réside donc dans la formulation de méthodologies distribuées pour la construction du modèle lui-même. C'est pourquoi les progrès dans le domaine de l'apprentissage statistique distribué se répercutent directement sous la forme de nouveaux usages et de nouvelles opportunités pour l'indexation par le contenu large échelle.

Motivés par cette étroite relation, nous proposons donc dans ce chapitre d'étudier l'impact des algorithmes d'apprentissage décentralisés et asynchrones proposés dans les chapitres précédents sur un système d'indexation par le contenu. Nous n'abordons pas la question de la comparaison des signatures en environnement distribué, qui a fait l'objet de nombreux travaux dédiés [201, 260]. On supposera ainsi qu'une fois calculées, les signatures sont suffisamment légères pour être échangées sur le réseau et comparées efficacement.

Plan du chapitre

Après un panorama succinct des méthodes d'indexation existantes, centralisées puis distribuées, nous présentons deux systèmes d'indexation mettant chacun en valeur un des algorithmes proposés précédemment :

1. Dans la section 5.3, AGKM (*cf* chapitre 2) est utilisé pour construire un dictionnaire visuel, élément de base de nombreuses méthodes d'indexation.

2. Dans la section 5.4, AGSVM (*cf* chapitre 4) est couplé à AGKM et AGPCA pour résoudre une tâche de classification d'images.

L'évaluation expérimentale de chaque système est réalisée sur des jeux de données académiques, de sorte à mettre en évidence ses capacité de passage à l'échelle et ses performances vis à vis d'une méthode centralisée. Nous concluons le chapitre par une synthèse des observations importantes et une discussions des perspectives qui accompagnent ce nouveau paradigme d'indexation.

5.1 Recherche par le contenu : état de l'art

Un système de recherche par le contenu s'appuie sur une *mesure de similarité* entre documents. Cette mesure de similarité correspond au choix d'un mode de représentation des documents et d'une métrique associée pour les comparer. Dans les méthodes de recherche issues de la vision par ordinateur, les documents sont fournis sous la forme d'un ensemble (sac) de descripteurs locaux. Il existe de nombreuses techniques d'extraction de tels descripteurs, selon le type de contenu considéré. Ces descripteurs sont typiquement des vecteurs dans un même espace de description $\mathbb{R}^D$, en nombre variable selon les caractéristiques du document (longueur, résolution, densité, *etc*). On note par la suite $\mathcal{X}_i = \{\mathbf{x}_{ij}\}_{j=1}^{n_i}$ l'ensemble des descripteurs extraits du document i, avec n_i le nombre de descripteurs du document i, et $\boldsymbol{\mathcal{X}} \equiv \bigcup_i \mathcal{X}_i$ l'ensemble de tous les descripteurs.

Il existe deux familles de mesures de similarité : celles qui effectue une comparaison directement sur les sacs de descripteurs, et celles qui s'appuient sur un encodage de chaque sac en un

unique vecteur, appelé signature. Dans les deux cas, on cherche un mesure de similarité qui ait des coûts de calcul et de stockage les plus faibles possibles pour permettre le passage à l'échelle.

5.1.1 Approches à base de vote

Pour comparer deux documents i et j, les approches à base de vote comparent directement leurs sacs de descripteurs respectifs. Chaque descripteur vote pour ses k plus proches voisins dans $\boldsymbol{\mathcal{X}}$. La similarité $k(\mathcal{X}_i, \mathcal{X}_j)$ entre les sacs $\mathcal{X}_i$ et $\mathcal{X}_j$ est alors donnée par le nombre de votes qui coincident entre i et j :

$$\forall i, j, \quad k(\mathcal{X}_i, \mathcal{X}_j) = \sum_{\mathbf{x}_{il} \in \mathcal{X}_i} \left| k\text{-}\underset{\boldsymbol{\mathcal{X}} \setminus \mathcal{X}_i}{\text{NN}}(\mathbf{x}_{il}) \cap \mathcal{X}_j \right| \tag{5.1}$$

où $k\text{-}\underset{\mathcal{A}}{\text{NN}}(\mathbf{x})$ est l'ensemble des k plus proches voisins de $\mathbf{x}$ dans l'ensemble $\mathcal{A}$. Une implémentation naïve des k plus proches voisins a une complexité linéaire avec le nombre de descripteurs dans $\boldsymbol{\mathcal{X}}$, ce qui est prohibitif pour de grandes collections. Un coût plus raisonnable peut être obtenu en utilisant une variante approximée où la recherche des voisins est limitée à un petit sous-ensemble par une stratégie sous-linéaire *a priori*.

Locally Sensitive Hashing (LSH, [105]) réduit l'ensemble de recherche via une fonction de hachage h telle que

$$\forall \mathbf{x}, \mathbf{y}, \quad \begin{cases} \mathbb{P}[h(\mathbf{x}) = h(\mathbf{y})] \geq P_1 & \text{si } \|\mathbf{x} - \mathbf{y}\| \leq R_1 \\ \mathbb{P}[h(\mathbf{x}) = h(\mathbf{y})] \leq P_2 & \text{si } \|\mathbf{x} - \mathbf{y}\| \geq R_2 \end{cases} \quad \text{avec} \quad \begin{array}{l} P_1 > P_2 \\ R_2 > R_1 \end{array} \tag{5.2}$$

En choisissant les paramètres P_1, P_2, R_1 et R_2 appropriés, on peut garantir que les descripteurs $(\mathbf{x}, \mathbf{y})$ qui entrent en collision (*i.e.*, $h(\mathbf{x}) = h(\mathbf{y})$) ont une probabilité élevée d'être voisins. On peut également accélérer la recherche de voisins par un découpage hiérarchique de l'espace de description [161, 207].

Les approches à base de vote ont deux défauts majeurs : *(i)* la comparaison de deux documents requiert l'ensemble complet des descripteurs (hypothèse inacceptable en environnement distribué), *(ii)* le coût de stockage intermédiaire des algorithmes de k plus proche voisins approximés est trop élevé pour passer à l'échelle.

5.1.2 Noyaux sur sacs

Les noyaux sur sacs sont une extension des fonctions noyaux classiques appliquée aux ensembles de vecteurs. Un noyau sur sac est défini par *(i)* une fonction de similarité entre descripteurs $k : (\mathbb{R}^D, \mathbb{R}^D) \mapsto [0, 1]$, appelé noyau mineur, *(ii)* un opérateur d'agrégation des évaluations du noyau mineur.

Dans [223], les auteurs proposent de calculer la similarité entre deux sacs $\mathcal{X}$ et $\mathcal{Y}$ comme la somme des évaluations du noyau mineur sur tous les appariements de descripteurs :

$$\forall i, j, \quad K(\mathcal{X}, \mathcal{Y}) = \sum_{\mathbf{x} \in \mathcal{X}} \sum_{\mathbf{y} \in \mathcal{Y}} k(\mathbf{x}, \mathbf{y}) \tag{5.3}$$

Pour accroître le pouvoir discriminant, Lyu [167] propose d'élever le noyau mineur à une puissance p :

$$\forall i, j, \quad K(\mathcal{X}, \mathcal{Y}) = \sum_{\mathbf{x} \in \mathcal{X}} \sum_{\mathbf{y} \in \mathcal{Y}} k(\mathbf{x}, \mathbf{y})^p \tag{5.4}$$

En augmentant p, on favorise les appariements de descripteurs fortement similaires. Bien que le premier défaut des méthodes à base de vote soit évité, la comparaison de deux sacs nécessite l'évaluation du noyau mineur sur toutes les paires de descripteurs, opération dont la complexité quadratique est inacceptable. On peut diminuer cette complexité en limitant l'évaluation du noyau mineur aux voisinages locaux :

$$\forall i,j, \quad K(\mathcal{X},\mathcal{Y}) = \sum_{\mathbf{x}\in\mathcal{X}} \sum_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{x},\mathbf{y}) k(\mathbf{x},\mathbf{y}), \quad \text{avec} \quad f(\mathbf{x},\mathbf{y}) = \begin{cases} 1 & \text{si } \|\mathbf{x}-\mathbf{y}\| < R \\ 0 & \text{sinon} \end{cases}$$

Si on peut déterminer les couples $(\mathbf{x},\mathbf{y})$ pour lesquels l'indicatrice f est non-nulle avec une complexité sous-linéaire (*e.g.*, via LSH), le noyau sur sac devient suffisamment rapide à calculer pour passer à l'échelle. Toutefois, cette accélération est toujours contrebalancée par un coût de stockage prohibitif.

5.1.3 Approches fréquentistes

Les approches fréquentistes se distinguent des méthodes présentées précédemment par la conversion de chaque sac de descripteurs en un unique vecteur signature. Pour ce faire, l'espace de description est quantifié en K régions en appliquant un algorithme de catégorisation (*e.g.*, K-means) à un ensemble représentatif de descripteurs (qui proviennent soit des documents à traiter soit d'un jeu d'entraînement distinct).

L'approche Bag-of-Words (BoW, [225]), inspirée des méthodes de recherche textuelles, interprète les descripteurs quantifiés comme des "mots visuels". Sous cette analogie, le quantificateur est appelé "dictionnaire visuel". La signature d'un document est alors un vecteur de dimension K qui contient les fréquences d'occurrence des K mots visuels dans le document. Pour rendre compte de la diversité des descripteurs représentés par un même mot, BOSSA (Bag of Statistical Sampling Analysis, [14]) calcule pour chaque mot un histogramme des distances entre les descripteurs et le centre de leur catégorie, résultant en une signature de dimension $L \times K$, où L est le nombre de classes de des histogrammes de distances.

Cependant, BoW est sensible à l'ambiguïté des mots visuels. Ce phénomène survient lorsqu'un descripteur se situe à proximité de la frontière entre deux mots. Pour résoudre ce problème, on peut remplacer la quantification vectorielle dure par un mélange doux de distributions de probabilité (*e.g.*, mélange de Gaussiennes). Les descripteurs sont ainsi assignés à plusieurs mots visuels en fonction de leur vraisemblance (*soft coding* [239]), produisant une signature de dimension K plus régulière que BoW.

La construction du dictionnaire visuel nécessite une phase préalable d'entraînement, étape coûteuse généralement réalisée hors-ligne.

5.1.4 Approches par codage

Les approches par codage produisent la signature d'un document en deux étapes :

***Coding step* :** Chaque descripteur $\mathbf{x}_i \in \mathbb{R}^D$ est projeté sur une base sur-complète de $K > D$ prototypes $\mathbf{M} = [\mu_1, \ldots, \mu_K]$ apprise par catégorisation (*e.g.*, K-means), pour en obtenir un code $\mathbf{y}_i \in \mathbb{R}^K$. Ce code est calculé de sorte à minimiser l'erreur (euclidienne) de reconstruction intégrant éventuellement des critères de régularisation encourageant la sparsité (*Sparse Coding* [158, 256]). *Locality-constrained Linear Coding* (LLC, [247]) intègre

une contrainte de localité forçant les descripteurs similaires à avoir un code similaire :

$$\forall \mathcal{X}, \qquad \mathcal{Y}^\star \equiv q(\mathcal{X}) = \underset{\mathcal{Y}=\{\mathbf{y}_1,\ldots,\mathbf{y}_{|\mathcal{X}|}\}}{\arg\min} \sum_{i=1}^{|\mathcal{X}|} \|\mathbf{x}_i - \mathbf{M}^\top \mathbf{y}_i\|_2^2 + \lambda \|\mathbf{d}_i \odot \mathbf{y}_i\|_2^2 \tag{5.5}$$

$$\text{avec} \quad \mathbf{d}_i = (d_{i1}, \ldots, d_{iK})^\top,\ d_{ik} = \exp \frac{\|\mathbf{x}_i - \mu_k\|_2}{\sigma} \tag{5.6}$$

***Pooling step* :** Tous les codes $\mathbf{y}_i$ sont condensés en un unique vecteur signature $\mathbf{z}$ de dimension K par un opérateur d'agrégation (*pooling operator*). Les opérateurs d'agrégation les plus communs sont :

- La somme : $\mathbf{z} = \sum_i \mathbf{y}_i$.
- Le max : $\mathbf{z} = \max_i \mathbf{y}_i$.

Les signatures $\mathbf{z}$, calculées pour chaque document, peuvent alors être comparées directement par une métrique vectorielle linéaire (*e.g.*, produit scalaire). Ces méthodes sont à la fois performantes et rapides. Tout comme les approches BoW, elles requièrent toutefois l'apprentissage préalable d'une base de projection sur-complète (généralement K est de l'ordre de quelques milliers, significativement moins que pour BoW).

5.1.5 Approches par déviation à un modèle

Les approches par déviation à un modèle reposent sur l'estimation préalable d'un modèle de l'espace des descripteurs sous la forme d'une distribution paramétrique $u_{\boldsymbol{\lambda}}$. Cette distribution est obtenue via un méthode non-supervisée de type EM ou K-means entraînée sur un large ensemble de documents. Chaque document est alors encodé en mesurant une certaine déviation statistique de la distribution de leurs descripteurs par rapport à l'ensemble d'entraînement.

Dans l'approche par *Fisher Vectors* (FV [133, 197, 210]), cette déviation est définie par le gradient de la log-vraisemblance par rapport aux paramètres du modèle :

$$\mathcal{G}_{\boldsymbol{\lambda}}(\mathcal{X}) = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x}\in\mathcal{X}} \nabla_{\boldsymbol{\lambda}} \ln u_{\boldsymbol{\lambda}}(\mathbf{x}) \tag{5.7}$$

Le modèle utilisé est une GMM diagonale. L'expression du gradient est alors donnée par

$$\forall \mathcal{X}, \forall k, \quad \mathcal{G}_{\mu,k}(\mathcal{X}) = \frac{1}{|\mathcal{X}|\sqrt{w_k}} \sum_{\mathbf{x}\in\mathcal{X}} \gamma_k(\mathbf{x}) \left(\frac{\mathbf{x} - \boldsymbol{\mu}_k}{\boldsymbol{\sigma}_k} \right) \tag{5.8}$$

$$\mathcal{G}_{\sigma,k}(\mathcal{X}) = \frac{1}{|\mathcal{X}|\sqrt{2w_k}} \sum_{\mathbf{x}\in\mathcal{X}} \gamma_k(\mathbf{x}) \left[\frac{(\mathbf{x} - \boldsymbol{\mu}_k)^2}{\boldsymbol{\sigma}_k^2} - 1 \right] \tag{5.9}$$

$$\text{avec} \quad \gamma_k(\mathbf{x}) = \frac{w_k u_k(\mathbf{x})}{\sum_l^K w_l u_l(\mathbf{x})} \tag{5.10}$$

où $(w_k, \boldsymbol{\mu}_k, \boldsymbol{\sigma}_k^2)$ sont respectivement le poids, la moyenne et la variance de la Gaussienne u_k. La signature d'un document est finalement donnée par la concaténation des $(\mathcal{G}_{\mu,k}, \mathcal{G}_{\boldsymbol{\sigma},k})$. Si le nombre de composantes est K, la taille de la signature est donc de $2KD$.

Dans l'approche par agrégation de tenseurs (Vector of Locally Aggregated Tensors, VLAT [200]), le modèle utilisé est une simple quantification vectorielle de l'espace de description (*e.g*, obtenue par K-means). Pour chaque cellule obtenue, on estime alors les moments statistiques (produits tensoriels) des descripteurs à différents ordres (1=moyenne, 2=matrice de covariance, *etc*) sur un large ensemble de documents d'entraînement :

$$\boldsymbol{\mu}_k = \frac{1}{|\mathcal{C}_k|} \sum \{\mathbf{x} \in \mathcal{C}_k\}, \quad \boldsymbol{\Sigma}_k = \frac{1}{|\mathcal{C}_k|} \sum_{\mathbf{x} \in \mathcal{C}_k} (\mathbf{x} - \boldsymbol{\mu}_k)(\mathbf{x} - \boldsymbol{\mu}_k)^\top, \tag{5.11}$$

où $\mathcal{C}_k$ est l'ensemble des descripteurs tombant dans la cellule k. Par la suite, chaque document est encodé en calculant la différence entre ses propres tenseurs et ceux appris sur l'ensemble d'entraînement. Par exemple, à l'ordre 2, on compare les covariances

$$\forall \mathcal{X}, \forall k, \quad \mathbf{T}_k = \sum_{\mathbf{x} \in \mathcal{C}_k} (\mathbf{x} - \boldsymbol{\mu}_k)(\mathbf{x} - \boldsymbol{\mu}_k)^\top - \boldsymbol{\Sigma}_k \tag{5.12}$$

où ici $\mathcal{C}_k$ est le sous-ensemble des descripteurs de $\mathcal{X}$ tombant dans la cellule k. La signature d'un document est alors donnée pare la concaténation des tenseurs $\mathbf{T}_k$ pour toutes les cellules k en un simple vecteur. Pour une quantification en K cellules et un ordre o, on obtient des signatures de dimension $\mathcal{O}(KD^o)$, dont certaines entrées peuvent être ignorée par symétrie des tenseurs. Les signatures VLAT sont généralement d'ordre 2. L'ordre 1 est incarné par la signature VLAD [138].

On peut voir les méthodes par déviation comme une extension des approches par dictionnaire visuel, où l'encodage se focalise sur la divergence des descripteurs par rapport aux mots visuels appris (*mismatch-based*) plutôt que sur leur mise en correspondance (*match-based*). Les méthodes par déviation offrent de très bonnes performances en classification en utilisant uniquement des classificateurs linéaires, simples et rapides à entraîner et évaluer [198]. Notons que leur performances peuvent être largement améliorées par diverses normalisations additionnelles (*e.g.*, L_2, *power-normalization*, ...) [110, 198].

5.1.6 Réseaux convolutifs profonds

Récemment, les réseaux convolutifs profonds (*Deep Convolutional Neural Networks*, CNN [20, 157]) ont reçu un enthousiasme particulier. Extensions des réseaux de neurones multicouches à rétro-propagation (*backpropagation*) développés dans les années 80, leurs performances dépassent aujourd'hui significativement la plupart des méthodes concurrentes [152, 230].

Les CNN combinent plusieurs avantages :

- Leur structure est homogène : toutes les couches d'un CNN sont composées de neurones (quasi-)identiques réalisant une même opération élémentaire (convolution + saturation non-linéaire) [181].
- Tous les paramètres sont appris, et ce via une seule et même règle d'adaptation : la rétro-propagation du gradient de l'erreur [252].
- Leur implémentation parallèle est particulièrement commode [66, 88, 152].

L'élégance des réseaux convolutifs réside principalement dans le fait que leur spécification se résume au dimensionnement des différentes couches (nombre de couches, taille des noyaux

de convolution, taux de recouvrement), plus quelques opérations intermédiaires simples (*max-pooling*, normalisation locale, *etc*).

Malgré leur performances exceptionnelles sur les jeux de données académiques courants et leur relative élégance, le comportement des réseaux de neurones profonds reste largement incompris. Depuis leur introduction il y a près de 50 ans, aucune méthodologie systématique n'a permis jusqu'à présent de lier rigoureusement le dimensionnement d'un réseau de neurone aux performances de reconnaissance (outre le fait empirique qu'augmenter la taille du réseau augmente généralement les performances). La conception d'un CNN est ainsi souvent critiquée comme relevant d'un 'art'. Certains travaux mettent d'ailleurs en évidence les résultats médiocres des CNN lorsqu'ils sont confrontés à des entrées ambiguës [227] (*e.g.*, des images dont les caractéristiques diffèrent de la distribution des images d'entraînement) et peuvent être facilement trompés par des motifs générés aléatoirement [186]. Ces travaux amènent généralement leurs auteurs à tempérer les ambitions de modèles homogènes et entièrement appris en faveur de modèles dont les capacités de généralisation proviennent d'une structure précontrainte et bien maîtrisée (*e.g.*, bases d'ondelettes, agrégation de tenseurs sur plusieurs couches, *etc* [43, 170]).

En dépit de ces divergences non-résolues au moment de la rédaction de ce manuscrit, une propriété essentielle semble obtenir une relative unanimité : l'utilisation de nombreuses couches semblables, par l'effet régularisateur qu'elle induit, permet d'entraîner des modèles dont le nombre gigantesque de paramètres (plusieurs centaines de millions) auraient conduit à un sur-apprentissage désastreux sans une structuration en couches. Cette propriété très générale constitue d'emblée un axe de développement pour les méthodes concurrentes, dans la mesure où l'organisation de modèles à structure plus contrainte (*Fisher Vectors*, VLAT, *etc*) en multiples couches pourrait légitimement combiner leurs avantages.

5.2 Système d'indexation décentralisé et asynchrone proposé

L'objectif d'un système d'indexation est d'encoder une collection de documents sous forme de signatures et d'offrir une mesure de similarité entre signatures qui reflète la similarité (subjective) du contenu des documents correspondants. Dans notre contexte d'indexation distribuée, on suppose de plus que ces documents ne sont pas réunis en un unique site de stockage, mais que la collection est dispersée sur une multitude de machines connectées en réseau. Naturellement, les capacités de stockage, de calcul et de communication des machines individuelles sont supposées limitées.

Le système d'indexation décentralisé et asynchrone que nous proposons se distingue nettement des systèmes conventionnels qui s'appuient sur l'attribution de rôles stricts aux différents acteurs du réseau (hébergeur, serveur, terminal utilisateur, *etc*). Il intègre en effet quatre préceptes principaux :

Interchangeabilité. Tous les nœuds jouent le même rôle : un même noeud fournit à la fois les services d'hébergeur de contenus, de moteur de recherche, et d'interface de formulation de requêtes utilisateur. Aucun noeud n'est crédité d'une confiance particulière en sa capacité à rendre un service de manière privilégiée.

Plasticité. Les contenus et le réseau sont supposés en perpétuelle évolution. Leur stationnarité ne se manifeste qu'à court terme. En particulier, un réseau peut à tout moment se séparer

en plusieurs réseaux disjoints, ou au contraire fusionner avec d'autres réseaux en cours de fonctionnement.

Asynchronisme. Chaque noeud évolue dans sa propre échelle de temps, en lien totalement asynchrone avec les autres acteurs du réseau. Les nœuds ne s'attendent pas les uns les autres. Un noeud peut rejoindre ou quitter le réseau à tout moment.

Autonomie. Chaque noeud est en charge de l'indexation de ses propres documents. Aucun document brut n'est échangé entre nœuds. Lorsqu'un noeud se retire du réseau, toute trace permettant d'identifier son contenu disparaît avec lui.

Le système proposé s'inscrit clairement dans un cadre décentralisé et asynchrone aussi bien dans un but technique de passage à l'échelle que dans une optique éthique d'équilibrage des responsabilités et de protection des données privées sur les réseaux.

L'objectif individuel de chaque nœuds consiste donc à calculer *localement* des signatures pour ses propres documents tout en s'accordant avec les autres acteurs sur une mesure de similarité *commune* qui offre une précision de recherche maximale sur la *totalité* des contenus du réseau.

Ces critères posés, de nombreuses contraintes pèsent sur un tel système :

- Les signatures doivent être de faible taille, afin de limiter les coûts de stockage et de communication lorsque l'on doit comparer deux documents hébergés par des machines distinctes.

- Le système doit permettre l'encodage de nouveaux documents à la volée. En particulier, le processus d'encodage des signature doit être suffisamment rapide pour tolérer le débit entrant de nouveaux documents.

- L'apparition de nouveaux documents peut provenir soit de la capture de nouveaux contenus par les machines déjà connectées, soit de la connexion de nouvelles machines. Le système doit donc également tolérer la connexion de nouveaux nœuds à tout moment.

- Le système ne doit pas présumer de la nature du réseau de machines, hormis l'hypothèse triviale de connexité forte.

- Le système doit tolérer tout déséquilibre en matière de répartition des contenus, aussi bien quantitatif que qualitatif.

- Le système doit offrir des garanties de fonctionnement en cas de disponibilité seulement ponctuelle de certains liens ou nœuds.

- Les coût de stockage, de calcul et de communication ne doivent pas exploser lorsque le nombre de nœuds (par extension le nombre de documents) augmente.

Dans ce chapitre, nous soutenons qu'un tel fonctionnement est rendu possible par la combinaison d'outils d'apprentissage statistique décentralisés et asynchrones, issus de la méthodologie et des algorithmes que nous avons présentés dans les chapitres précédents.

5.2.1 Architecture générale

Le système d'indexation que nous proposons est une extension décentralisée et asynchrone du paradigme d'indexation par dictionnaire visuel. Chaque noeud implémente une instance identique de ce système et l'exécute indépendamment des autres nœuds (sans coordination). Celui-ci est composée de quatre composants :

Apprentissage du dictionnaire visuel. Ce composant extrait des descripteurs visuels de tous les documents locaux et échange des messages avec les nœuds voisins de sorte à converger vers un dictionnaire visuel commun qui rende compte de la densité des descripteurs de l'ensemble du réseau. Pour ce faire, nous utilisons l'algorithme AGKM présenté dans le chapitre 2.

Encodage des signatures. Ce composant encode chaque document en une signature vectorielle. Plus précisément, le sac de descripteurs extrait de chaque document est projeté sur un modèle VLAT d'ordre 2 :

- Dans un premier temps, le modèle de référence est construit en calculant (5.11) sur l'ensemble des descripteurs du réseau. En tant que moyennes, le calcul des moments de référence μ_k et $\mathbf{\Sigma}_k$ pour chaque cellule k est réalisé par AGAVG (*cf.* chapitre 1).
- Chaque document est encodé localement via (5.12). Autrement dit, la signature d'un document est donnée par le calcul, pour chaque cellule k du dictionnaire visuel, de la déviation de la covariance de ses descripteurs tombant dans k par rapport à la covariance de référence $\mathbf{\Sigma}_k$. Cette opération ne requiert aucune communication entre nœuds.

Compression des signatures. Ce composant se décompose en deux parties indépendantes :

- Une PCA est appliquée à chaque cellule du dictionnaire. Ceci permet de centrer et réduire les descripteurs au sein de leur cellule (blanchiment), et d'en réduire la dimensionnalité.
- La deuxième partie applique une PCA globale aux signatures de façon à réduire leur dimension finale.

Ces deux parties font appel à l'algorithme AGPCA décrit dans le chapitre 3.

Classification supervisée. Ce composant est facultatif. Dans le cas où le noeud dispose d'annotations manuelles spécifiant l'appartenance ou non de certains documents à une ou plusieurs classes, un classificateur est entraîné en échangeant avec les nœuds voisins qui possèdent des annotations pour une ou plusieurs de ces classes. Pour chacune de ces classes, un classificateur SVM distinct est entraîné grâce à l'algorithme AGSVM présenté dans le chapitre 4.

Par la combinaison de ces quatre composants, chaque document du réseau est converti en une signature compacte stockée sur le nœud qui l'héberge. Pour comparer deux documents, on applique alors directement une métrique vectorielle (*e.g.* L_2, produit scalaire) sur leurs signatures.

5.2.2 Scénario incrémental et limitations

Notons que les quatre composants de notre architecture d'indexation doivent être ordonnancés judicieusement :

- L'encodage des signatures doit attendre la stabilisation de l'apprentissage du dictionnaire visuel.
- La compression ne peut se faire que lorsque l'on dispose de toutes les signatures locales. Notons toutefois que le calcul local des matrices de Gram nécessaires pour AGPCA peut être lancé dès que la première signature est disponible.
- La classification peut être lancée dès que la première signature compressée est disponible.

Ceci induit donc une chaîne de dépendances à la fois fonctionnelles et temporelles entre composants. Or, lorsque la distribution des contenus est amenée à changer dans le temps (données non-stationnaires), il peut être nécessaire de ré-entraîner régulièrement le dictionnaire visuel, qui ne reflète plus la densité des descripteurs disponibles. L'invalidité du dictionnaire courant peut être simplement détectée par une mesure régulière de la MQE.

Cependant, dès lors que le modèle est considéré comme invalide, un nouveau modèle doit être entraîné sur les documents disponibles. En utilisant le dictionnaire courant comme initialisation, on peut raisonnablement espérer une convergence rapide de cet apprentissage (*warm start*). Une telle stratégie *warm start* peut également être appliquée à AGPCA et AGSVM. Ceci ouvre l'opportunité d'un fonctionnement incrémental, où le modèle est actualisé à la volée lorsque celui-ci ne représente plus les données d'entrée avec une précision suffisante.

Bien que la plupart des composants du système puissent fonctionner de manière incrémentale et ainsi offrir des capacités d'optimisation en ligne, cet avantage est limité par l'obligation de ré-encoder toutes les signatures à chaque mise à jour du modèle. En effet, contrairement aux diverses étapes d'optimisation, on ne dispose pas de stratégie incrémentale pour l'étape d'encodage. A chaque modification du modèle, il faut donc extraire à nouveau tous les descripteurs et les projeter sur le modèle actualisé. Les étapes de compression doivent alors être complètement relancées. Un tel scénario s'avère coûteux en ressources de calcul et de stockage.

Une perspective intéressante serait d'adopter une approche d'encodage totalement incrémentale. Ceci permettrait de mettre à jour en permanence le modèle et les signatures de façon à s'adapter en ligne aux changement de distribution des descripteurs. On disposerait ainsi d'un système capable de maximiser constamment ses performances en présence de données non-stationnaires.

5.2.3 Protocole expérimental

Ce chapitre présente les résultats et l'analyse de deux campagnes d'évaluation. La première évalue les composants d'apprentissage des dictionnaires visuels et d'encodage des signatures sur une tâche de recherche d'images par similarité. La seconde évalue le système d'indexation complet sur une tâche de classification d'images.

Le système a été évalué à la fois en simulation et en condition réelles jusqu'à 100 nœuds. Son implémentation est basée sur une librairie logicielle de calcul décentralisé en C++ que nous avons baptisée libAGML (*Asynchronous Gossip Machine Learning*), disponible à l'adresse suivante :

FIGURE 5.1 – Images de la base Holidays (Inria)

FIGURE 5.2 – Images de la base PASCAL VOC 2007

`http://perso-etis.ensea.fr/~jerofell/software.html`

Dans toutes nos expériences, les descripteurs utilisés sont de type HoG (Histogrammes d'orientation des Gradients) mono-échelle de dimension $D = 128$. Naturellement, notre système peut utiliser tout type de descripteurs vectoriels. Les documents sont dispersés uniformément sur les nœuds (les nœuds n'ont aucun document en commun).

Nous évaluons tout d'abord le système sur une tâche de recherche d'images par similarité. Nous utilisons la base d'images Inria Holidays [136], constituée de 1 491 images réparties en 500 requêtes (*cf.* figure 5.1). L'objectif associé à cette base est la recherche de quasi-copies (*i.e.* une même scène dans des conditions de prise de vue différente : angle, éclairage, occultations, etc...). Le critère de performance retenu est la précision moyenne (mAP). Pour un document requête donné, la précision moyenne évalue la capacité du système à trier les documents de sorte à ce que les documents pertinents apparaissent en tête. Ainsi, un mAP de 100% signifie que pour tout document requête, le système retourne tous les documents pertinents (et seulement ceux-ci) en tête de liste. Sauf mention spécifique, nous évaluons le mAP en chaque noeud sur l'ensemble des documents du réseau, puis rapportons sa valeur moyenne (et éventuellement son écart type) sur le réseau.

Le système est ensuite évalué sur une tâche de classification d'images. Nous utilisons la base d'images PASCAL VOC 2007 [86], qui comporte 9963 images dont 5011 sont annotés manuellement en 20 classes (*cf.* figure 5.2). Chaque classe correspond à la présence d'un objet particulier (avion, bicyclette, ...) dans l'image. L'objectif est de déterminer la présence ou l'absence de ces 20 objets dans les 4652 images non-annotées. Nous utilisons deux critères d'évaluation : *(i)* la fonction de coût évaluée sur le jeu d'entraînement et *(ii)* la précision moyenne (mAP) pour chaque classe.

5.3 Apprentissage de dictionnaire visuel

Dans un premier temps, nous évalons les composants d'apprentissage de dictionnaire visuel et d'encodage des signatures. Les méthodes d'indexation par déviation à un modèle ont l'avan-

tage de se contenter de petits dictionnaires (de 64 à 1024 cellules, contre plusieurs centaines de milliers pour BoW).

Dans cette campagne, nous avons adopté des signatures VLAT [199] d'ordre 1, analogues à des signatures VLAD [139], dont la dimension est KD où D est la dimension des descripteurs. Chaque noeud i exécute la procédure décrite dans l'algorithme 10.

Algorithme 10 Procédure d'indexation

1: Extraction d'un jeu de descripteurs HoG (échantillonnage dense) pour chaque image locale (environ 2500 descripteurs par image).
2: Exécution d'AGKM sur l'ensemble des descripteurs locaux. AGKM se termine en i lorsque la MQE a convergé et que la distorsion des cellules est équitablement répartie (*cf* chapitre 2).
3: Pour chaque document, son sac de descripteurs est projeté sur le dictionnaire local obtenu. On calcule alors sa signature VLAD :

$$\mathbf{v} = (\mathbf{v}_1, ..., \mathbf{v}_K) \quad \text{avec} \quad \mathbf{v}_k = \sum_{\mathbf{x} \in \mathcal{C}_k} (\mathbf{x} - \boldsymbol{\mu}_k), \tag{5.13}$$

où $\mathcal{C}_k$ est l'ensemble de ses descripteurs assignées à la cellule k du dictionnaire, et $\boldsymbol{\mu}_k$ la moyenne de la cellule k prise sur l'ensemble des images du réseau. Précisons que $\boldsymbol{\mu}_k$ est directement donné par le k-ième prototype du dictionnaire visuel en sorti d'AGKM. Les signatures sont ensuite normalisées ($\|\mathbf{v}\|_2 = 1$).

Une fois l'encodage réalisé, chaque couple d'images est comparé localement en calculant le produit scalaire de leurs signatures.

Nous comparons alors les performances en terme de mAP par rapport à un dictionnaire de référence de 64 cellules calculé de manière centralisée (K-means + VLAD), qui réalise sur cette tâche un mAP d'environ 55 %.

On observe que le système proposé offre des résultats d'indexation équivalents voire supérieurs à ceux de la méthode centralisée de référence, tel que le montre la figure 5.3. De telles performances signifient qu'un nœud qui ne dispose localement que d'informations parcellaires sur les contenus manipulés est capable de modéliser l'ensemble des données grâce à notre méthode décentralisée d'apprentissage de dictionnaires.

Notamment, chaque noeud modélise des concepts visuels qu'il n'héberge pas localement, là où des systèmes centralisés entraînés isolément sur chaque nœud montrent une mauvaise généralisation comme l'atteste la figure 5.4.

Les expériences menées pour différents nombres de nœuds N et tailles de dictionnaire K, dont les résultats sont rassemblés en figure 5.5, montrent que le système décentralisé maintient des performances supérieures à la méthode de référence et présente une mAP très correcte pour de petits dictionnaires.

Dans le but d'optimiser les coûts de communication, la figure 5.6 montre que les dictionnaires produits en utilisant de faibles valeurs de M (nombre de messages envoyés entre deux étapes de partitionnement, *cf* chapitre 2) obtiennent une mAP aussi élevée que lorsque M est grand. Les simulations menées sur des réseaux de 50 nœuds avec $K = 64$ révèlent que bien qu'un nombre plus important d'itérations soit nécessaire pour atteindre les meilleurs résultats, les coûts totaux de communication sont fortement réduits. Le système global maintient malgré

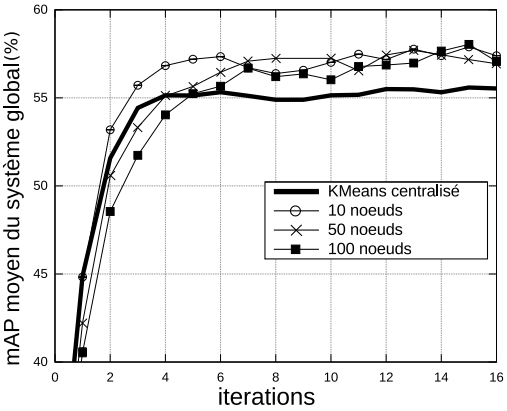

FIGURE 5.3 – Le dictionnaire commun construit par notre système fournit des résultats de recherche (mAP) équivalents voire meilleurs qu'une solution centralisée

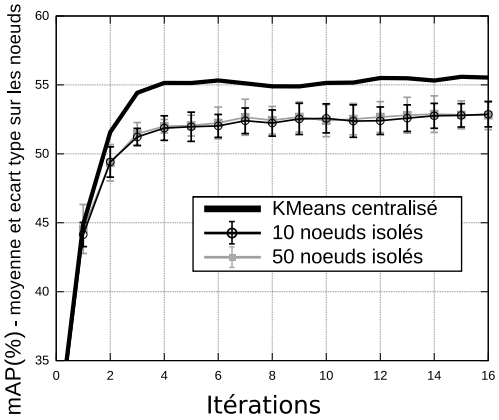

FIGURE 5.4 – Des dictionnaires locaux entraînés isolément sont moins performants qu'une approche centralisé

	K = 4	**K = 8**	**K = 16**	**K = 32**	**K = 64**
référence	42,7	46,5	49,8	53,3	55,1
10 nœuds	44,2	47,9	51,6	54,2	57,0
50 nœuds	42,7	45,9	50,9	55,7	57,2
100 nœuds	44,7	47,3	51,8	53,4	56,0

FIGURE 5.5 – Performances (mAP en %) du système pour différentes tailles de dictionnaire K et nombre de nœuds N

tout des performances de recherche asymptotiques équivalentes, même lorsque chaque nœud n'envoie que 2 messages par itération. Notons cependant que la légère dégradation de la cohérence des dictionnaires naturellement provoquée par la réduction de M se traduit par une variance plus élevée des résultats de recherche entre nœuds, visible dans la figure 5.7.

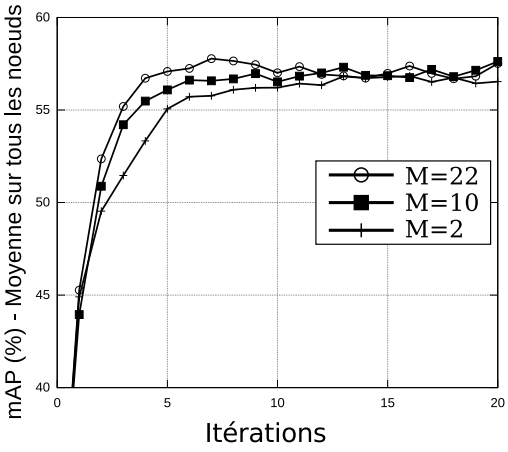

FIGURE 5.6 – Influence de la réduction des coûts de communication sur la précision moyenne

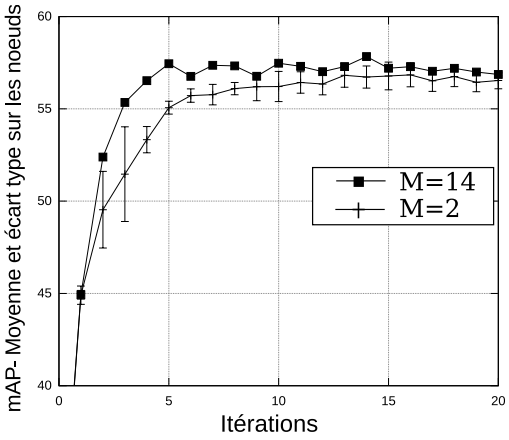

FIGURE 5.7 – Influence de la réduction des coûts de communication sur l'écart type de la précision moyenne

5.4 Classification d'images

Pour finir, nous évaluons les performances du système proposé dans une tâche de classification d'images sur la base VOC 2007. Pour ce faire, nous ajoutons à la procédure précédente une dernière étape d'entraînement d'une SVM linéaire. Celle-ci est confiée à AGSVM, chacune des 20 classes étant traitée par un classificateur distinct (*one versus all*).

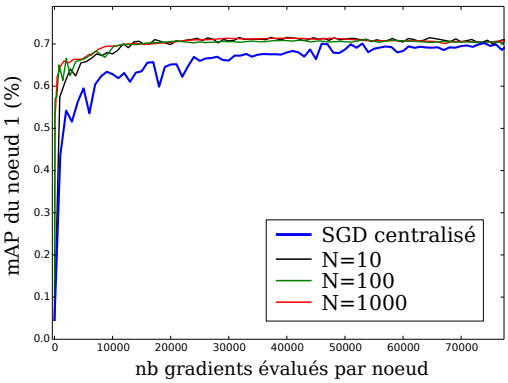

FIGURE 5.8 – Performances du système en classification pour différentes tailles de réseau N (ici, $L = 1$ et $M = 10$).

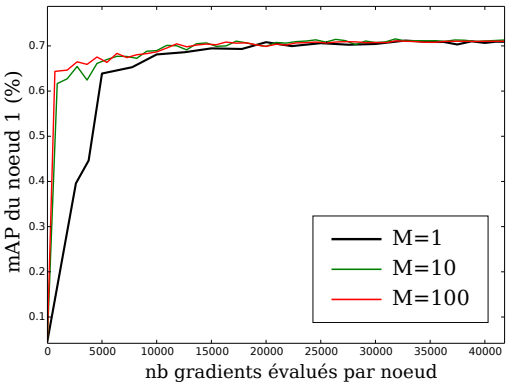

FIGURE 5.9 – Influence du paramètre M sur la convergence du système.

Comme dans la section précédente, les signatures sont compressées en fixant $d = 64$ et $q = 9963$. Notons que puisque q est égal au nombre total de documents, nous sommes assurés que l'étape de compression finale est sans effet sur les performances de recherche. En effet, la matrice de Gram ayant un rang forcément inférieur ou égal au nombre de documents, elle est parfaitement conservée par cette compression. AGSVM étant un solveur primal, cette compression accélère grandement l'entraînement des SVM.

Comme observé dans le chapitre précédent, le moyennage des gradients n'apporte pas d'ac-

célération particulière et peut nuire à la stabilité d'AGSVM. Nous fixons donc $L = 1$ et seuls les hyperplans sont moyennés entre nœuds. Cette variante d'AGSTAG peut être interprétée à une SGD décentralisée et asynchrone.

La figure 5.8 montre que plus le réseau comporte de nœuds, plus la convergence vers un mAP optimal est rapide. Dans tous les cas, le système converge vers un mAP d'environ 71%, identique à un système centralisé. Il est intéressant de noter que l'accélération obtenue tend à s'amenuiser pour les très grand réseau, en tendant vers une asymptote où l'augmentation de N n'apporte plus de rapidité supplémentaire. Par ailleurs, plus N est grand, plus l'évolution du mAP est régulière et stable, ce qui est appréciable dans le cas où l'on dispose d'un budget de temps limité pour entraîner le classificateur (*cf* chapitre 4).

L'influence du paramètre M (nombre de message envoyé entre deux calculs de gradient, *cf* chapitre 4) est illustrée par la figure 5.9, qui compare la convergence pour $M = \{1, 10, 100\}$ avec $N = 10$. Bien qu'une valeur faible de M limite les coûts de communication, on observe que le temps de convergence est impacté lorsque M est très faible. Néanmoins, dès que $M \geq 10$, ce ralentissement est négligeable. A la marge, $M = 100$ confère une stabilité légèrement accrue du mAP. D'après nos expériences réalisées pour diverses tailles de réseau, nous concluons que $M = 10$ est une valeur acceptable quelle que soit la valeur de N.

5.5 Conclusion

Dans ce chapitre, nous avons proposé un système complet d'indexation multimédia décentralisé et asynchrone qui permet l'indexation conjointe de multiples collections de documents multimédias répartis sur de larges réseaux. Chaque machine du réseau est capable de produire des signatures pour ses propres documents qui offrent une précision d'indexation équivalente aux méthodes centralisées usuelles. Toutes les machines exécutent un processus local identique qui n'implique qu'une communication réduite et asynchrone avec les machines voisines.

Le système proposé s'appuie sur les algorithmes d'apprentissage décentralisés et asynchrones introduits dans les chapitres 2 à 4 : AGKM, AGPCA et AGSVM. Il bénéficie ainsi de toutes les propriétés avantageuses liées à la décentralisation et l'asynchronisme exposés en introduction du présent manuscrit. Il offre par conséquent une réponse *systémique* à trois problèmes posés aux outils modernes d'indexation par le contenu :

- L'indexation de milliards de documents devient théoriquement abordable, sous réserve que ces documents soient répartis sur un nombre suffisant d'unités de traitement connectées en réseau.
- L'indexation du contenu "à la source" devient techniquement réalisable et même computationnellement avantageuse.
- La centralisation de grandes masses d'information potentiellement privées n'est plus nécessaire. L'absence d'entité centrale et l'interchangeabilité des rôles des acteurs garantit une répartition équitable des responsabilités et des usages de chacun.

La principale limitation du système d'indexation proposé tient à la nature non-incrémentale de l'étape d'encodage des signatures. Celle-ci empêche une utilisation en ligne efficiente. L'étude de solutions d'encodage incrémentales constitue donc une perspective importante de cette thèse, dans la mesure où toutes les autres étapes présentent un caractère incrémental. L'intégration d'un

encodeur incrémental permettrait d'envisager une étude comportementale plus poussée du système, dans un contexte dynamique où les contenus sont non-stationnaires (dérive conceptuelle, plasticité du réseau, *etc*).

Conclusion

Dans cette thèse, nous avons montré l'intérêt d'utiliser une approche décentralisée et asynchrone pour la mise en œuvre de techniques d'apprentissage statistique sur de larges réseaux. L'approche proposée s'appuie sur des protocoles Gossip asynchrones. Dans ce paradigme, chaque nœud du réseau exécute une procédure identique, et les échanges d'information n'impliquent aucune attente entre nœuds. Cette propriété d'asynchronisme permet une exploitation optimale des ressources disponibles, en supprimant le temps d'inactivité qu'induit l'utilisation de communications synchrones dans la plupart des algorithmes actuels. Si cet avantage apporte des gains de performance modestes sur de petits réseaux, les protocoles Gossip asynchrones expriment leur plein potentiel sur les réseaux très larges, où il surpassent les méthodes à coordination centralisée d'un ordre de grandeur.

Par ailleurs, nous avons mis en évidence les propriétés bénéfiques de la décentralisation sur le respect de la confidentialité des données et de leur nature intrinsèquement distribuée. En effet, les algorithmes d'apprentissage présentées dans cette thèse ne transmettent jamais les échantillons eux-mêmes sur le réseau. Seuls les paramètres du modèle optimisé sont échangés, garantissant ainsi que chaque nœud participant au calcul conserve ses données privées pour son seul usage. Ceci assure également un usage parcimonieux du débit de communication disponible.

D'un point de vue pratique, nous avons formulé des algorithmes Gossip asynchrones pour la catégorisation non-supervisée, l'estimation de densité, la réduction de dimension, la classification supervisée et l'optimisation convexe en général. Ces algorithmes égalent leurs homologues centralisés en termes de précision, tout en offrant une accélération significative par la répartition du calcul sur de multiples unités. Cette accélération se fait au prix d'un léger surcoût de communication qui devient négligeable lorsque le nombre de nœuds est suffisamment grand.

Chaque algorithme proposé a fait l'objet d'une analyse théorique qui fournit une prédiction non-asymptotique du coût de communication et de calcul en fonction de la taille du réseau et de la confiance désirée sur le résultat. Cette analyse a également révélé que ces algorithmes tolèrent (dans une certaine mesure) les changements de topologie de réseau et de distribution des données en cours de fonctionnement.

Au-delà de présenter un ensemble d'algorithmes dédiés chacun à une tâche d'apprentissage particulière, nous avons finalement proposé un formalisme générique à la fois théorique et pratique pour la conception d'algorithmes d'apprentissage massivement parallèles. Le champ d'application de ce formalisme ne se réduit donc pas aux seuls algorithmes proposés dans ce manuscrit. Les concepts généraux que nous avons développés, tels que les protocoles Gossip perturbés, l'agrégation Gossip de sous-espaces ou le moyennage de gradients asynchrone à court terme peuvent être à loisir combinés, étendus à d'autres problèmes, retranscrits dans d'autres

types d'espaces ou appliqués à d'autres formes de contenus numériques.

Par ailleurs, les travaux présentés dans ce manuscrit s'inscrivent dans un contexte de mutation progressive des concepts sous-jacents aux architectures numériques. L'apparition d'architectures électroniques reconfigurables (*e.g.*, FPGA) et de réseaux sur puce performants ouvre de nouvelles possibilités applicatives enthousiasmantes, mais requièrent la définition de nouveaux paradigmes algorithmiques qui exploitent ces potentiels inédits. Nous pensons que le formalisme défini dans cette thèse représente une base solide pour de tels paradigmes, dans la mesure où il offre un cadre complet de conception, d'analyse et d'implémentation pour ces algorithmes.

Dans la continuité des travaux canoniques sur l'apprentissage statistique, cette thèse s'est concentrée sur des problèmes d'estimation ergodiques et stationnaires, principalement incarnés sous la forme de fonctions de coût à optimiser sur un ensemble d'échantillons statique. Toutefois, certaines disciplines relevant de l'apprentissage artificiel ne se basent pas sur ces deux hypothèses. C'est le cas par exemple du contrôle temps-réel en automatique, de l'apprentissage par renforcement, de l'apprentissage développemental, de la prédiction de tendances en finance, ou la détection de communautés sur les réseaux sociaux. Dans ces domaines, les données ne suivent pas une distribution stationnaire, notamment car le phénomène observé évolue dans le temps. La structure même du modèle peut être amenée à changer pour s'adapter à un changement d'objectif, d'interface entre le système étudié et son environnement, ou de ressources disponibles. Nous suggérons qu'une extension non-stationnaire et non-ergodique (*e.g.*, chaotique) des protocoles Gossip sur des réseaux dynamiques peut apporter des solutions élégantes dans ces champs disciplinaires émergents.

Les applications exposées dans le dernier chapitre ont principalement une valeur illustrative. Nous n'avions pas pour objectif initial d'augmenter les performances des systèmes de reconnaissance et d'inférence existants, mais plutôt d'en proposer une implémentation décentralisée. Nous avons toutefois observé des gains de performance appréciables, que l'on peut imputer à la nature intrinsèquement stochastique, décentralisée et asynchrone des schémas algorithmiques considérés. Par exemple, l'initialisation aléatoire des modèles locaux à chaque nœud augmente l'effet de "brassage" des paramètres, que l'on ne peut obtenir avec un algorithme séquentiel. Lorsque le choix des paramètres initiaux conditionne grandement la qualité finale du modèle à convergence (*e.g.*, en catégorisation), ce brassage diminue la probabilité de débuter avec une "mauvaise" initialisation, augmentant par conséquent la qualité et la fiabilité du modèle final. L'envergure précise de ces avantages reste à déterminer et fait partie des perspectives importantes de cette thèse.

Une autre perspective intéressante concerne l'expérimentation du paradigme sur une large échelle et à des niveaux de granularité hétérogènes. Par exemple, il est théoriquement valide d'exploiter simultanément le même algorithme Gossip au niveau le plus bas (*e.g.*, le réseau d'unités de calcul d'un processeur manycore) jusqu'au niveau le plus haut (*e.g.*, le réseau mondial des téléphones mobiles) en passant par des niveaux intermédiaires (*e.g.*, un cluster de calcul scientifique national), ceci en allouant les ressources de calcul et de stockage selon les besoins de l'application. Cette perspective invite à une myriade d'applications pratiques dans des disciplines scientifiques et industrielles aussi variées que le cloud computing pour la physique expérimentale, productrice d'immenses flots de données, la coordination de systèmes déportés en contrôle multi-agent, l'analyse embarquée dans les réseaux de capteurs, l'indexation de collections artistiques sujettes à des droits patrimoniaux restrictifs, *etc*.

L'implémentation des algorithmes proposés dans cette thèse est aisée. Afin d'en favoriser

la diffusion, nous avons mise en ligne une librairie C/C++ nommée libAGML, qui implémente les algorithmes présentés et offre une API unifiée pour la conception d'algorithmes inspirés du formalisme Gossip asynchrone. Cette librairie est disponible à l'adresse
http://perso-etis.ensea.fr/~jerofell/software.html

Annexe A : Démonstrations

A. Théorème 2 : Convergence des algorithmes Gossip push-pull

Soit un protocole Gossip push-pull définit par $\mathbf{x}(t+1)^\top = \mathbf{x}(t)^\top \mathbf{K}(t+1)$ où les $\mathbf{K}(t)$ sont iid et telles que

$$\mathbf{1}^\top \mathbf{K}(t) = \mathbf{1}^\top, \qquad \mathbf{K}(t)\mathbf{1} = \mathbf{1} \qquad \text{et} \qquad \mathbb{E}\mathbf{K} \text{ est primitive} \tag{5.14}$$

L'erreur relative par rapport au consensus est alors bornée comme suit :

$$\forall 0 < \delta < 1,\ \forall \varepsilon > 0, \quad \mathbb{P}\left[\frac{\|\mathbf{x}(t) - \bar{x}\mathbf{1}\|_2^2}{\|\mathbf{x}(0)\|_2^2} \geq \varepsilon\right] \leq \delta \tag{5.15}$$

$$\text{si} \quad t \geq T \equiv \frac{\ln \delta + \ln \varepsilon}{2 \ln \lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top])} \tag{5.16}$$

Démonstration. Notre schéma de preuve s'inspire de [39] et [130]. Notons $\mathbf{P}(t) = \mathbf{K}(1)\dots\mathbf{K}(t)$.

On a ainsi $\mathbf{x}(t)^\top = \mathbf{x}(0)^\top \mathbf{P}(t)$. Puisque les $\mathbf{K}(t)$ sont iid, on peut noter sans ambiguïté $\mathbb{E}\mathbf{K} \equiv \mathbb{E}[\mathbf{K}(t)]$, et on a $\mathbb{E}[\mathbf{P}(t)] = \mathbb{E}\mathbf{K}^t$. Par conséquent,

$$\mathbb{E}\mathbf{x}(t) = \mathbf{x}(0)^\top \mathbb{E}\mathbf{K}^t \tag{5.17}$$

Par hypothèse, $\mathbb{E}\mathbf{K}$ est primitive et $\mathbf{1}^\top \mathbb{E}\mathbf{K} = \mathbf{1}^\top$. D'après le théorème de Perron-Frobenius, on sait donc que

$$\lim_{t\to\infty} \mathbb{E}\mathbf{K}^t = \frac{\mathbf{1}\mathbf{1}^\top}{N} \qquad \text{d'où} \qquad \lim_{t\to\infty} \mathbb{E}\mathbf{x}(t) = \frac{\mathbf{1}\mathbf{1}^\top}{N}\mathbf{x}(0) = \bar{x}\mathbf{1} \tag{5.18}$$

Les estimées convergent donc en espérance vers le consensus.

Observons à présent que puisque $\mathbf{1}^\top = \mathbf{1}^\top \mathbf{K}(t)$, on a

$$\mathbf{x}(t)^\top - \bar{x}\mathbf{1}^\top = \mathbf{x}(0)^\top \mathbf{P}(t) - \mathbf{x}(0)^\top \frac{\mathbf{1}\mathbf{1}^\top}{N}\mathbf{P}(t) \tag{5.19}$$

$$= \mathbf{x}(0)\mathbf{J}\mathbf{P}(t) \qquad \text{où} \quad \mathbf{J} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\top \tag{5.20}$$

Notons que $\mathbf{J}$ est la matrice de projection sur le sous-espace orthogonal à $\mathbf{1}$ (on a clairement $\mathbf{J}\mathbf{1} = \mathbf{0}$). Toute matrice stochastique $\mathbf{A}$ satisfait alors

$$\mathbf{J}\mathbf{A}\mathbf{J} = \mathbf{J}\mathbf{A} - \frac{1}{N}\mathbf{J}\mathbf{A}\mathbf{1}\mathbf{1}^\top = \mathbf{J}\mathbf{A} - \frac{1}{N}\mathbf{J}\mathbf{1}\mathbf{1}^\top = \mathbf{J}\mathbf{A}. \tag{5.21}$$

En utilisant la sous-multiplicativité de la norme spectrale $\|\|_2$ définie par $\|\mathbf{A}\|_2 \equiv \rho(\mathbf{A}\mathbf{A}^\top) \equiv \rho(\mathbf{A}^\top\mathbf{A})$ on obtient la borne suivante sur l'erreur relative à l'instant t :

$$\frac{\|\mathbf{x}(t) - \bar{x}\mathbf{1}\|_2^2}{\|\mathbf{x}(0)\|_2^2} \leq \|\mathbf{JP}(t)\|_2^2 \tag{5.22}$$

Bornons $\|\mathbf{JP}(t)\|_2$ en exploitant (5.21) :

$$\begin{aligned}
\|\mathbf{JP}(t+1)\|_2 &= \|\mathbf{JP}(t)\mathbf{K}(t+1)\|_2 && (5.23)\\
&= \|\mathbf{JP}(t)\mathbf{JK}(t+1)\|_2 && (5.24)\\
&\overset{(5.21)}{\leq} \|\mathbf{JK}(t)\|_2\|\mathbf{JP}(t)\|_2 && (5.25)\\
&= \rho(\mathbf{JK}(t)\mathbf{K}(t)^\top)\|\mathbf{JP}(t)\|_2 && (5.26)
\end{aligned}$$

Puisque $\rho(\mathbf{K}(t)) = 1$ et $\mathbf{K}(t)$ $\mathbf{J}$ est orthogonal à $\mathbf{1}$, il est clair que

$$\rho(\mathbf{JK}(t)\mathbf{K}(t)^\top) = \lambda_2(\mathbf{K}(t)\mathbf{K}(t)^\top) \tag{5.27}$$

En rappelant que les $\mathbf{K}(t)$ sont iid on obtient donc

$$\mathbb{E}\|\mathbf{JP}(t+1)\|_2 \leq \lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top])\mathbb{E}\|\mathbf{JP}(t)\|_2 \tag{5.28}$$

En développant la récursion et en observant que $\|\mathbf{J}\|_2 = 1$ on a

$$\mathbb{E}\|\mathbf{JP}(t)\|_2 \leq \lambda_2^t(\mathbb{E}[\mathbf{K}\mathbf{K}^\top]) \tag{5.29}$$

Ainsi, le taux de convergence vers le consensus est régi par la seconde valeur propre de $\mathbb{E}[\mathbf{K}\mathbf{K}^\top]$. Il nous faut maintenant prouver que $\lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top]) < 1$, c'est à dire que $\mathbb{E}[\mathbf{K}\mathbf{K}^\top]$ est primitive. Pour ce faire, nous nous appuyons sur le lemme suivant

Lemme 3. *Soit $\mathbf{K} \in \mathbb{R}^{N\times N}$ une matrice aléatoire (à support fini). Si $\mathbb{E}\mathbf{K}$ est primitive, alors $\mathbb{E}[\mathbf{K}\mathbf{K}^\top]$ est aussi primitive.*

Démonstration. Puisque $\mathbb{E}\mathbf{K}$ est primitive, il existe $m \in \mathbb{N}$ tel que $\mathbb{E}\mathbf{K}^m > \mathbf{0}$. En notant $\mathcal{E}$ le support de $\mathbf{K}$, on a donc

$$\sum_{\mathbf{A}\in\mathcal{E}} \mathbb{P}[\mathbf{K} = \mathbf{A}]\mathbf{A}^m > \mathbf{0} \tag{5.30}$$

Il existe donc une réalisation $\mathbf{A}$ de $\mathbf{K}$ telle que $\mathbf{A}^m > \mathbf{0}$. Par conséquent

$$\begin{aligned}
(\mathbf{A}^m)(\mathbf{A}^m)^\top &> \mathbf{0} && (5.31)\\
\sum_{\mathbf{A}\in\mathcal{E}} \mathbb{P}[\mathbf{K} = \mathbf{A}](\mathbf{A}^m)(\mathbf{A}^m)^\top &> \mathbf{0} && (5.32)\\
\mathbb{E}[\mathbf{K}\mathbf{K}^\top]^m &> \mathbf{0}, && (5.33)
\end{aligned}$$

ce qui prouve que $\mathbb{E}[\mathbf{K}\mathbf{K}^\top]$ est primitive. □

Retournons à la démonstration du théorème 2. Puisque $\mathbb{E}\mathbf{K}$ est primitive, $\mathbb{E}[\mathbf{K}\mathbf{K}^\top]$ est donc

primitive elle aussi. Le théorème de Perron-Frobenius nous dit alors que $\lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top]) < 1$. Appliquons l'inégalité de Markov à (5.22) :

$$\forall \varepsilon > 0, \quad \mathbb{P}\left[\frac{\|\mathbf{x}(t) - \bar{x}\mathbf{1}\|_2^2}{\|\mathbf{x}(0)\|_2^2} \geq \varepsilon\right] \leq \frac{\mathbb{E}\|\mathbf{J}\mathbf{P}(t)\|_2^2}{\varepsilon} \tag{5.34}$$

$$\leq \varepsilon^{-1}\lambda_2^{2t}(\mathbb{E}[\mathbf{K}\mathbf{K}^\top]) \tag{5.35}$$

On obtient le résultat recherché en fixant $\delta \leq \mathbb{P}\left[\frac{\|\mathbf{x}(t)-\bar{x}\mathbf{1}\|_2^2}{\|\mathbf{x}(0)\|_2^2} \geq \varepsilon\right]$:

$$\delta \leq \varepsilon^{-1}\lambda_2^{2t}(\mathbb{E}[\mathbf{K}\mathbf{K}^\top]) \tag{5.36}$$

$$\lambda_2^{2t}(\mathbb{E}[\mathbf{K}\mathbf{K}^\top]) \geq \delta\varepsilon \tag{5.37}$$

$$t \geq \frac{\ln\delta + \ln\varepsilon}{2\ln\lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top])} \tag{5.38}$$

□

B. Théorème 3 : Convergence du protocole Newscast

Pour un réseau totalement connecté de N noeuds, Newscast atteint une erreur inférieure à ε avec une probabilité au moins δ en moins de M messages, où

$$M \leq \frac{N}{2}(\ln 1/\delta + \ln 1/\varepsilon) \tag{5.39}$$

Démonstration. Rappelons que dans le cas de Newscast, on a

$$\mathbf{K}(t) = \mathbf{I} - \frac{1}{2}(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\top, \quad \text{où} \quad (i,j) \sim \mathcal{U}_A \tag{5.40}$$

Le théorème 2 nous assure alors que nombre de messages M à échanger pour converger à une précision ε avec une probabilité au moins $1-\delta$ est borné comme suit :

$$M \leq \frac{\ln\delta + \ln\varepsilon}{2\ln\lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top])} \tag{5.41}$$

où l'on a utilisé le fait qu'à chaque instant t exactement 2 messages sont échangés.

Calculons $\mathbb{E}[\mathbf{K}\mathbf{K}^\top]$:

$$\mathbf{K}(t)\mathbf{K}(t)^\top = \left(\mathbf{I} - \frac{1}{2}(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\top\right)^2 = \mathbf{I} - \frac{1}{2}(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\top = \mathbf{K}(t) \tag{5.42}$$

$$\mathbb{E}[\mathbf{K}\mathbf{K}^\top] = \mathbb{E}\mathbf{K} = \left(1 - \frac{1}{N-1}\right)\mathbf{I} + \frac{1}{N(N-1)}\mathbf{1}\mathbf{1}^\top \tag{5.43}$$

La simplicité de l'expression (5.43) de $\mathbb{E}\mathbf{K}$ permet alors de déterminer explicitement ses valeurs propres via le lemme suivant :

Lemme 4. *Soit la matrice réelle* $\mathbf{A} = \alpha\mathbf{I} + \beta\mathbf{1}\mathbf{1}^\top$. *Ses valeurs propres sont*

$$\lambda_1(\mathbf{A}) = \alpha + \beta \qquad \textit{et} \qquad \lambda_2(\mathbf{A}) = \alpha \tag{5.44}$$

de multiplicités respectives 1 et $N - 1$.

Démonstration.

$$\det(\mathbf{A} - \lambda\mathbf{I}) = \det\Big((\alpha - \lambda)\mathbf{I} + \beta\mathbf{1}\mathbf{1}^\top\Big) \tag{5.45}$$

$$= (\alpha - \lambda)^N \det(\mathbf{I} + \frac{\beta}{\alpha - \lambda}\mathbf{1}\mathbf{1}^\top) \tag{5.46}$$

$$= (\alpha - \lambda)^N\Big(1 - \frac{\beta}{\lambda - \alpha}\Big) \tag{5.47}$$

où (5.47) utilise l'identité de Sylvester $\det(\mathbf{I} + \mathbf{A}\mathbf{B}) = \det(\mathbf{I} + \mathbf{B}\mathbf{A})$. Par conséquent, $\mathbf{A}$ a une valeur propre simple $\lambda_1(\mathbf{A}) = \alpha + \beta$ et une valeur propre $\lambda_2(\mathbf{A}) = \alpha$ de multiplicité $N - 1$.

On a donc $\lambda_2(\mathbb{E}\mathbf{K}) = 1 - \frac{1}{N-1}$. La borne sur M devient alors

$$M \leq \frac{\ln\delta + \ln\varepsilon}{2\ln(1 - \frac{1}{N-1})} \tag{5.48}$$

□

Cette borne peut être simplifiée à l'aide du lemme suivant :

Lemme 5.

$$\forall x > 0, \quad \frac{1}{\ln(1 - \frac{1}{x})} \leq -x \tag{5.49}$$

Démonstration. On a

$$-x\Big/\frac{1}{\ln(1 - \frac{1}{x})} = -x\ln(1 - \frac{1}{x}) = \ln\left[(1 - \frac{1}{x})^{-x}\right] \tag{5.50}$$

Or, pour $y < 0$ on a

$$(1 + \frac{1}{y})^y > e \quad \text{avec} \quad \lim_{y\to-\infty}(1 + \frac{1}{y})^y = e \tag{5.51}$$

En prenant $x = -y$, on obtient

$$\forall x > 0, \quad (1 - \frac{1}{x})^{-x} > e \tag{5.52}$$

$$\ln\left[(1 - \frac{1}{x})^{-x}\right] > 1 \tag{5.53}$$

$$\frac{1}{\ln(1 - \frac{1}{x})} < -x \tag{5.54}$$

□

Le nombre de messages nécessaires pour atteindre une ε-convergence avec une probabilité au moins $1 - \delta$ est donc borné par

$$M \leq \frac{N}{2}\big(\ln 1/\delta + \ln 1/\varepsilon\big) \tag{5.55}$$

□

C. Théorème 4 : Convergence des protocoles Gossip asynchrones par moyennage temporel

Soit $(\mathbf{K}(t))_{t\geq 1}$ une suite de matrices aléatoires iid telles que

$$\forall t,\ \mathbf{K}(t) \text{ non-négative}, \quad \mathbb{E}\mathbf{K} \text{ primitive} \quad \text{et} \quad \mathbb{E}\mathbf{K} \text{ doublement stochastique} \tag{5.56}$$

Pour chaque noeud i, définissons les estimées locales suivantes

$$x_i(t+1) = \sum_{j=1}^{N} \mathbf{K}_{ji}(t) x_j(t) \quad \text{et} \quad y_i(t+1) = \frac{t y_i(t) + x_i(t+1)}{t+1} \tag{5.57}$$

Alors les moyennes temporelles $y_i(t)$ convergent quasi-sûrement vers la moyenne $\bar{x} \equiv \frac{1}{N}\sum_{i=1}^{N} x_i(0)$:

$$\forall i,\ \forall \varepsilon > 0, \quad \lim_{t\to\infty} \mathbb{P}\Big[|y_i(t) - \bar{x}| < \varepsilon\Big] = 1 \tag{5.58}$$

Démonstration. Définissons $\mathbf{y}(t) = (y_1(t), \dots, y_N(t))$ et calculons $\mathbb{E}\mathbf{y}(t)$:

$$\mathbb{E}\mathbf{y}(t)^\top = \mathbb{E}\Big[\frac{1}{t}\sum_{\tau=1}^{t} \mathbf{x}(t)^\top\Big] = \frac{1}{t}\sum_{\tau=1}^{t} \mathbb{E}\mathbf{x}(t)^\top = \frac{1}{t}\sum_{\tau=1}^{t} \mathbf{x}(0)^\top \mathbb{E}\Big[\prod_{k=1}^{\tau} \mathbf{K}(k)\Big] \tag{5.59}$$

$$= \frac{1}{t}\sum_{\tau=1}^{t} \mathbf{x}(0)^\top \mathbb{E}\mathbf{K}^\tau \tag{5.60}$$

Or $\mathbb{E}\mathbf{K}$ est primitive et doublement stochastique, c'est à dire $\mathbb{E}\mathbf{K}\mathbf{1} = \mathbf{1}$ et $\mathbf{1}^\top \mathbb{E}\mathbf{K} = \mathbf{1}^\top$. Le théorème de Perron-Frobenius nous assure alors que $\rho(\mathbb{E}\mathbf{K}) = 1$ et $\lambda_2(\mathbb{E}\mathbf{K}) < 1$. Par conséquent,

$$\mathbb{E}\mathbf{K}^t = \frac{\mathbf{1}\mathbf{1}^\top}{N} + \lambda_2^t(\mathbb{E}\mathbf{K})\mathbf{M}, \quad \text{où} \quad \mathbf{M} \leq \mathbb{E}\mathbf{K} \tag{5.61}$$

En integrant (5.61) à (5.60), on obtient

$$\mathbb{E}\mathbf{y}(t)^\top = \frac{1}{Nt}\sum_{\tau=1}^{t} \mathbf{x}(0)^\top \mathbf{1}\mathbf{1}^\top + \frac{1}{t}\mathbf{x}(0)^\top \mathbf{M} \sum_{\tau=1}^{t} \lambda_2^\tau(\mathbb{E}\mathbf{K}) \tag{5.62}$$

$$= \frac{\mathbf{x}(0)^\top \mathbf{1}}{N}\mathbf{1}^\top + \frac{1}{t}\mathbf{x}(0)^\top \mathbf{M} \underbrace{\sum_{\tau=1}^{t} \lambda_2^\tau(\mathbb{E}\mathbf{K})}_{S_t} \tag{5.63}$$

Analysons le comportement de la série S_t. En tant que série géométrique, et sachant que $\lambda_2(\mathbb{E}\mathbf{K}) < 1$, on a

$$\lim_{t\to\infty} S_t = \frac{\lambda_2(\mathbb{E}\mathbf{K})}{1 - \lambda_2(\mathbb{E}\mathbf{K})} \tag{5.64}$$

Par conséquent, le second terme de (5.63) tend vers $\mathbf{0}$, ce qui entraîne

$$\lim_{t\to\infty} \mathbb{E}\mathbf{y}(t) = \bar{x}\mathbf{1}^\top \tag{5.65}$$

Rappelons que $\mathbf{y}(t) \equiv \frac{1}{t}\sum_{\tau=1}^{t} \mathbf{x}(\tau)$. En utilisant l'inégalité de Hoeffding [121], on obtient une borne probabiliste sur l'écart maximal entre les $y_i(t)$ et leur espérance :

$$\forall \varepsilon > 0, \quad \mathbb{P}\Big[\|\mathbf{y}(t) - \mathbb{E}\mathbf{y}(t)\|_\infty \geq \varepsilon\Big] \leq 2\exp(-2t\varepsilon^2) \tag{5.66}$$

On a donc asymptotiquement

$$\lim_{t\to\infty} \mathbb{P}\Big[\|\mathbf{y}(t) - \mathbb{E}\mathbf{y}(t)\|_\infty \geq \varepsilon\Big] = \lim_{t\to\infty} \mathbb{P}\Big[\|\mathbf{y}(t) - \bar{x}\mathbf{1}^\top\|_\infty \geq \varepsilon\Big] = 0 \tag{5.67}$$

D'où

$$\lim_{t\to\infty} \mathbb{P}\Big[\|\mathbf{y}(t) - \bar{x}\mathbf{1}^\top\|_\infty \leq \varepsilon\Big] = 1, \tag{5.68}$$

ce qui conclue la preuve. □

D. Convergence des protocoles asynchrones de type Sum-Weight

Dans cette section, nous rappelons que $(\mathbf{K}(t))_{t\geq 1}$ est une suite de matrices aléatoires stochastiques et iid telles que $\mathbb{E}\mathbf{K}$ est primitive. Nous exploitons également le produit $\mathbf{P}(t) = \prod_{\tau=1}^{t} \mathbf{K}(\tau)$. Enfin,

$$\mu_S(\mathbf{A}) \equiv \frac{1}{2}\max_{ij} \sum_{k=1}^{N} |\mathbf{A}_{ik} - \mathbf{A}_{jk}| \tag{5.69}$$

désigne le coefficient d'ergodicité de Dobrushin de la matrice $\mathbf{A}$.

D1. Lemme 1 : Borne sur l'erreur d'estimation de la moyenne

Soient les estimées $\mathbf{x}(t) = (x_1(t), \ldots, x_N(t))$ et $\mathbf{w}(t) = (w_1(t), \ldots, w_N(t))$ définies par

$$\mathbf{x}(t+1)^\top = \mathbf{x}(t)^\top \mathbf{K}(t+1) \quad \text{et} \quad \mathbf{w}(t+1)^\top = \mathbf{w}(t)^\top \mathbf{K}(t+1)$$

En notant $\bar{x} = \frac{\sum_{i=1}^{N} x_i(0)}{\sum_{i=1}^{N} w_i(0)}$, on a

$$\left\| \frac{\mathbf{x}(t)}{\mathbf{w}(t)} - \bar{x}\mathbf{1} \right\|_2 \leq \frac{2\mu_S(\mathbf{P}(t))}{\min_{ij} \mathbf{P}_{ji}(t)} \|\bar{x}\mathbf{1}\|_2 \tag{5.70}$$

où $\mu_S(\mathbf{A})$ est le coefficient d'ergodicité de Dobrushin de la matrice $\mathbf{A}$

Démonstration.

$$\left|\frac{x_i(t)}{w_i(t)} - \bar{x}\right| = \left|\frac{\sum_j^N x_j(0)\mathbf{P}_{ji}(t)}{\sum_j^N w_j(0)\mathbf{P}_{ji}(t)} - \frac{\sum_j^N x_j(0)}{\sum_j^N w_j(0)}\right| \tag{5.71}$$

$$\leq \left|\frac{\max_j \mathbf{P}_{ji}(t)\sum_j^N x_j(0)}{\min_j \mathbf{P}_{ji}(t)\sum_j^N w_j(0)} - \frac{\sum_j^N x_j(0)}{\sum_j^N w_j(0)}\right| \tag{5.72}$$

$$= \left|\frac{\max_j \mathbf{P}_{ji}(t)}{\min_j \mathbf{P}_{ji}(t)} - 1\right| |\bar{x}| \tag{5.73}$$

$$= \frac{\max_{jk} |\mathbf{P}_{ji}(t) - \mathbf{P}_{ki}(t)|}{\min_j \mathbf{P}_{ji}(t)} |\bar{x}| \tag{5.74}$$

$$\left\|\frac{\mathbf{x}(t)}{\mathbf{w}(t)} - \bar{x}\mathbf{1}\right\|_2^2 \leq \frac{\sum_{i=1}^N \max_{jk}(\mathbf{P}_{ji}(t) - \mathbf{P}_{ki}(t))^2}{\min_{ij} \mathbf{P}_{ji}(t)^2} \|\bar{x}\mathbf{1}\|_2^2 \tag{5.75}$$

$$\leq \frac{\frac{1}{2}\sum_{ijk}^N (\mathbf{P}_{ji}(t) - \mathbf{P}_{ki}(t))^2}{\min_{ij} \mathbf{P}_{ji}(t)^2} \|\bar{x}\mathbf{1}\|_2^2 \tag{5.76}$$

$$\leq \frac{\sum_{ij}^N (\mathbf{P}_{ij}(t) - \frac{1}{N}\sum_{k=1}^N \mathbf{P}_{kj}(t))^2}{\min_{ij} \mathbf{P}_{ji}(t)^2} \|\bar{x}\mathbf{1}\|_2^2 \tag{5.77}$$

$$= \frac{\|\mathbf{JP}(t)\|_F^2}{\min_{ij} \mathbf{P}_{ji}(t)^2} \|\bar{x}\mathbf{1}\|_2^2 \tag{5.78}$$

Remarquons alors que pour toute matrice stochastique $\mathbf{A}$,

$$\|\mathbf{JA}\|_F^2 \leq \sum_{ij}^N \left(\mathbf{A}_{ij} - \frac{1}{N}\sum_{k=1}^N \mathbf{A}_{kj}\right)^2 \leq \frac{1}{N^2}\sum_{ij}^N \left(\sum_{k=1}^N |\mathbf{A}_{ij} - \mathbf{A}_{kj}|\right)^2 \tag{5.79}$$

$$\leq \left(\max_{ij} \sum_{k=1}^N |\mathbf{A}_{ij} - \mathbf{A}_{kj}|\right)^2 = 4\mu_S^2(\mathbf{A}) \tag{5.80}$$

En combinant (5.78) et (5.80) on obtient le résultat recherché :

$$\left\|\frac{\mathbf{x}(t)}{\mathbf{w}(t)} - \bar{x}\mathbf{1}\right\|_2 \leq \frac{2\mu_S(\mathbf{P}(t))}{\min_{ij} \mathbf{P}_{ji}(t)} \|\bar{x}\mathbf{1}\|_2 \tag{5.81}$$

□

D2. Théorème 6 : ε-convergence non-asymptotique des protocoles Sum-Weight

Étant données une probabilité δ et une erreur $\varepsilon > 0$, pour tout t tel que

$$\max\left\{\frac{3\ln N - \ln\varepsilon - 2C\ln\chi - \ln\frac{\delta+1}{2}}{-\ln\lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top])}, C\right\} \quad \text{avec} \quad C = \frac{16N(\ln N + 1)}{(1-\delta)(1-\lambda_2(\mathcal{G}))}$$

on a

$$\mathbb{P}\left[\frac{\left\|\frac{\mathbf{x}(t)}{\mathbf{w}(t)} - \bar{x}\mathbf{1}\right\|_2^2}{\|\bar{x}\mathbf{1}\|_2^2} \leq \varepsilon\right] \geq \delta,$$

où χ est la plus petite entrée non-nulle de toute réalisation de $\mathbf{K}(t)$ et $\lambda_2(\mathcal{G})$ est la 2ème plus petite valeur propre du Laplacien du graphe $\mathcal{G}$.

Démonstration.
Notre démonstration comporte quatre parties. Nous bornons tout d'abord l'erreur relative d'estimation de la moyenne par un quotient $P(t)/Q(t)$ où le numérateur $P(t)$ capture l'ergodicité faible du protocole et le dénominateur $Q(t)$ capture le poids minimal à travers le réseau. Par la suite, nous proposons une borne exponentielle sur $P(t)$, puis une borne inférieure constante sur $Q(t)$ qui est valide à partir d'un temps en $\mathcal{O}(N \ln N)$. Nous concluons la démonstration en combinant les deux bornes pour obtenir un temps t en $\mathcal{O}(N \ln N)$ suffisant pour obtenir l'ε-convergence, qui prend en compte la structure du réseau et des matrices aléatoires $\mathbf{K}(t)$.

1. Borne explicite sur l'erreur d'estimation de la moyenne

$$\left|\frac{x_i(t)}{w_i(t)} - \bar{x}\right| = \left|\frac{\sum_j \mathbf{P}_{ji}(t)x_j(0)}{w_i(t)} - \frac{\sum_j x_j(0)}{\sum_j w_j(0)}\right| \tag{5.82}$$

$$\leq \left|\frac{\sum_j x_j(0)\max_j \mathbf{P}_{ji}(t)}{\sum_j w_j(0)w_i(t)/W} - \frac{\sum_j x_j(0)}{\sum_j w_j(0)}\right| \tag{5.83}$$

$$= |\bar{x}|\left|\frac{\max_j \mathbf{P}_{ji}(t)}{w_i(t)/W} - 1\right| \tag{5.84}$$

$$= |\bar{x}|\left|\frac{\max_j \mathbf{P}_{ji}(t) - \frac{1}{W}\sum_j \mathbf{P}_{ji}(t)w_j(0)}{w_i(t)/W}\right| \tag{5.85}$$

$$\leq |\bar{x}|\left|\frac{\max_j \mathbf{P}_{ji}(t) - \min_j \mathbf{P}_{ji}(t)}{w_i(t)/W}\right| \tag{5.86}$$

$$= |\bar{x}|\frac{\max_{jk}\left|\mathbf{P}_{ji}(t) - \mathbf{P}_{ki}(t)\right|}{w_i(t)/W} \tag{5.87}$$

En élevant au carré et en sommant, on obtient

$$\left\|\frac{\mathbf{x}(t)}{\mathbf{w}(t)} - \bar{x}\mathbf{1}\right\|_2^2 \leq \frac{\sum_{i=1}^N \max_{jk}(\mathbf{P}_{ji}(t) - \mathbf{P}_{ki}(t))^2}{\min_i\left(w_i(t)/W\right)^2}\|\bar{x}\mathbf{1}\|_2^2 \tag{5.88}$$

$$\leq \frac{\frac{1}{2}\sum_{ijk}^N(\mathbf{P}_{ji}(t) - \mathbf{P}_{ki}(t))^2}{\min_i\left(w_i(t)/W\right)^2}\|\bar{x}\mathbf{1}\|_2^2 \tag{5.89}$$

$$\leq \frac{\sum_{ij}^N(\mathbf{P}_{ij}(t) - \frac{1}{N}\sum_{k=1}^N \mathbf{P}_{kj}(t))^2}{\min_i\left(w_i(t)/W\right)^2}\|\bar{x}\mathbf{1}\|_2^2 \tag{5.90}$$

$$= \frac{\|\mathbf{JP}(t)\|_F^2}{\min_i\left(w_i(t)/W\right)^2}\|\bar{x}\mathbf{1}\|_2^2 \tag{5.91}$$

$$= \frac{P(t)}{Q(t)}\|\bar{x}\mathbf{1}\|_2^2 \tag{5.92}$$

où nous avons défini

$$P(t) = \|\mathbf{JP}(t)\|_F^2 \qquad \text{et} \qquad Q(t) = \min_i \big(w_i(t)/W\big)^2 \tag{5.93}$$

$$\text{avec} \qquad \mathbf{J} = \mathbf{I} - \frac{\mathbf{1}\mathbf{1}^\top}{N} \tag{5.94}$$

2. Borne supérieure sur le numerateur $P(t)$

Dans cette partie, nous exploitons le lemme spectral suivant :

Lemme 6. *Pour toutes matrices* $\mathbf{A}$ *et* $\mathbf{B}$*, on a* $\|\mathbf{AB}\|_F^2 \leq \|\mathbf{B}\|_2^2 \|\mathbf{A}\|_F^2$*, où* $\|\cdot\|_2$ *désigne la norme spectrale.*

Démonstration.

$$\|\mathbf{AB}\|_F^2 = \text{Trace}(\mathbf{B}^\top \mathbf{B}\mathbf{A}\mathbf{A}^\top) \tag{5.95}$$

$$\leq \sum_i^N \sigma_i(\mathbf{B}^\top\mathbf{B})\sigma_i(\mathbf{A}\mathbf{A}^\top) \qquad \textit{(Inégalité de Von Neuman)} \tag{5.96}$$

$$\leq \sigma_{\max}(\mathbf{B}^\top\mathbf{B}) \sum_i \sigma_i(\mathbf{A}\mathbf{A}^\top) = \sigma_{\max}^2(\mathbf{B}) \sum_i \sigma_i^2(\mathbf{A}) = \|\mathbf{B}\|_2^2 \|\mathbf{A}\|_F^2 \tag{5.97}$$

□

Appliquons ce lemme à l'expression de $P(t+1)$:

$$P(t+1) = \|\mathbf{JP}(t)\mathbf{K}(t+1)\|_F^2 \tag{5.98}$$

$$= \|\mathbf{JP}(t)\mathbf{JK}(t+1)\|_F^2 \tag{5.99}$$

$$\leq \|\mathbf{JK}(t+1)\|_2^2 P(t) \tag{5.100}$$

Les $\mathbf{K}(t)$ étant iid, on a

$$\mathbb{E}P(t+1) \leq \mathbb{E}\Big[\|\mathbf{JK}\|_2^2\Big]\mathbb{E}P(t) \tag{5.101}$$

En rappelant que $\|\mathbf{A}\|_2 = \sqrt{\rho(\mathbf{A}\mathbf{A}^\top)}$ et $\mathbf{J} \perp \mathbf{1}$ on trouve

$$\mathbb{E}\Big[\|\mathbf{JK}\|_2^2\Big] = \mathbb{E}\Big[\rho(\mathbf{JKK}^\top\mathbf{J})\Big] = \lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top]) \tag{5.102}$$

Par conséquent

$$\mathbb{E}P(t) \leq P(0)\lambda_2^t(\mathbb{E}[\mathbf{K}\mathbf{K}^\top]) \tag{5.103}$$

L'inégalité de Markov nous donne alors, pour tout $\varepsilon > 0$,

$$\mathbb{P}[P(t) \geq \varepsilon] \leq \frac{P(0)}{\varepsilon}\lambda_2^t(\mathbb{E}[\mathbf{K}\mathbf{K}^\top]) \tag{5.104}$$

Etant donné une probabilité $0 < \delta < 1$, on cherche

$$T : \forall t \geq T,\ \mathbb{P}[P(t) \geq \varepsilon] \leq \delta \tag{5.105}$$

une condition suffisante est alors donnée par

$$\frac{P(0)}{\varepsilon}\lambda_2^t(\mathbb{E}[\mathbf{K}\mathbf{K}^\top]) \leq \delta \tag{5.106}$$

$$\lambda_2^t(\mathbb{E}[\mathbf{K}\mathbf{K}^\top]) \leq \frac{\varepsilon}{P(0)}\delta \tag{5.107}$$

$$t \ln \lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top]) \leq \ln \frac{\varepsilon}{P(0)} + \ln \delta \tag{5.108}$$

$$t \geq \frac{\ln \varepsilon - \ln P(0) + \ln \delta}{\ln \lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top])} \tag{5.109}$$

$$t \geq \frac{\ln \varepsilon - \ln N + \ln \delta}{\ln \lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top])} \tag{5.110}$$

où la dernière ligne utilise le fait que $P(0) = \|\mathbf{J}\|_F^2 = N - 1$.

3. Borne inférieure sur le dénominateur $Q(t)$

Observons que puisque pour tout t, $\mathbf{P}(t)$ est stochastique (ses lignes somment à 1), on a

$$\sum_i w_i(t) = \sum_i w_0(t) = W \tag{5.111}$$

À tout instant t, il existe donc au moins un noeud m pour lequel

$$w_m(t) \geq \frac{W}{N} \tag{5.112}$$

Notons $\mathbf{P}(t, \Delta t)$ le produit $\mathbf{K}(t) \cdots \mathbf{K}(t + \Delta t)$.

On a alors pour tout $\Delta t > 0$,

$$\forall i,\ w_i(t + \Delta t) = \sum_j \mathbf{P}_{ji}(t, \Delta t) w_j(t) \tag{5.113}$$

$$\geq \mathbf{P}_{mi}(t, \Delta t) w_m(t) \tag{5.114}$$

$$\geq \frac{W}{N}\mathbf{P}_{mi}(t, \Delta t) \tag{5.115}$$

Notons χ la plus petite entrée non nulle parmi toutes les realisations possibles de $\mathbf{K}(t)$. Rappelons également que les $\mathbf{K}(t)$ ont une diagonale strictement positive. On a alors pour tout $\Delta t, i$ et j,

$$\mathbf{P}_{ij}(t, \Delta t) > 0 \quad \Leftrightarrow \quad \mathbf{P}_{ij}(t, \Delta t) \geq \chi^{\Delta t} \tag{5.116}$$

Par conséquent,

$$\forall i, \mathbf{P}_{mi}(t, \Delta t) > 0 \qquad \Rightarrow \qquad \min_i w_i(t + \Delta t) \geq \frac{W}{N}\chi^{\Delta t} \qquad \Rightarrow \qquad Q(t + \Delta t) \geq \left(\frac{\chi^{\Delta t}}{N}\right)^2 \tag{5.117}$$

Remarquons que puisque les $\mathbf{K}(t)$ sont i.i.d., la distribution de $\mathbf{P}(t, \Delta t)$ ne dépend que de Δt et pas de t. Autrement dit,

$$\forall t, \Delta t, \quad \mathbb{P}\Big[\forall i, \mathbf{P}_{mi}(t, \Delta t) > 0\Big] = \mathbb{P}\Big[\forall i, \mathbf{P}_{mi}(t) > 0\Big] \tag{5.118}$$

Choisissons une ligne arbitraire m et posons

$$T = \inf\Big\{t : \forall i, \mathbf{P}_{mi}(t) > 0\Big\} \tag{5.119}$$

On a donc pour tout t

$$T < t \quad \Rightarrow \quad \forall i, \mathbf{P}_{mi}(t) > 0 \tag{5.120}$$

$$\mathbb{P}\Big[T < t\Big] \leq \mathbb{P}\Big[\forall i, \mathbf{P}_{mi}(t) > 0\Big] \tag{5.121}$$

$$\mathbb{P}\Big[\forall i, \mathbf{P}_{mi}(t) > 0\Big] \geq 1 - \mathbb{P}\Big[T \geq t\Big] \tag{5.122}$$

En utilisant l'inégalité de Markov, on obtient

$$\mathbb{P}\Big[\forall i, \mathbf{P}_{mi}(t) > 0\Big] \geq 1 - \frac{\mathbb{E}T}{t} \tag{5.123}$$

Si l'on souhaite que $\mathbb{P}\Big[\forall i, \mathbf{P}_{mi}(t) > 0\Big] \geq \delta$ avec δ une probabilité donnée, il suffit donc que

$$1 - \frac{\mathbb{E}T}{t} \geq \delta \qquad \text{c'est à dire} \qquad t \geq \frac{\mathbb{E}T}{1-\delta} \tag{5.124}$$

En combinant avec (5.117-5.118), on obtient

$$\forall \delta \in]0,1[, \qquad \forall t \geq \frac{\mathbb{E}T}{1-\delta}, \quad \mathbb{P}\left[Q(t) \geq \frac{1}{N^2} \exp\left\{\frac{2\mathbb{E}T \cdot \ln \chi}{1-\delta}\right\}\right] \geq \delta \tag{5.125}$$

Il nous reste à calculer $\mathbb{E}T$. Pour ce faire, comptons le nombre $a(t)$ d'entrées non-nulles sur la ligne m de $\mathbf{P}(t)$:

$$a(t) = \Big|\mathcal{A}(t)\Big| = \Big|\Big\{j : \mathbf{P}_{mj}(t) > 0\Big\}\Big| \tag{5.126}$$

où $\mathcal{A}(t)$ désigne l'ensemble des entrées non-nulles sur la ligne m de $\mathbf{P}(t)$. Les $\mathbf{K}(t)$ étant à diagonale positive, $a(t)$ ne peut que croitre. Par définition, $a(0) = 1$ car $\mathbf{P}(0) = \mathbf{I}$. On sait de plus que comme

$$\mathbf{P}_{mj}(t+1) = \sum_k \mathbf{P}_{mk}(t)\mathbf{K}_{kj}(t+1), \tag{5.127}$$

on a

$$\mathbf{P}_{mj}(t+1) > 0 \qquad \Leftrightarrow \qquad \exists k : \mathbf{P}_{mk}(t) > 0 \text{ et } \mathbf{K}_{kj}(t+1) > 0 \tag{5.128}$$

ou de manières équivalentes

$$\mathcal{A}(t+1) = \mathcal{A}(t) \cup \Big\{j : \exists k \in \mathcal{A}(t),\ \mathbf{K}_{kj}(t+1) > 0\Big\} \tag{5.129}$$

$$a(t+1) - a(t) = \Big|\Big\{j \notin \mathcal{A}(t) : \exists k \in \mathcal{A}(t),\ \mathbf{K}_{kj}(t+1) > 0\Big\}\Big| \tag{5.130}$$

$a(t+1)$ augmente donc du nombre de noeuds n'appartenant pas à $\mathcal{A}(t)$ contactés par un (ou plusieurs) noeud de $\mathcal{A}(t)$. Ainsi définie par (5.130), $a(t)$ forme une chaîne de Markov (homogène) correspondant à un processus de croissance de population (problème du type "collectionneur de vignettes" ou *coupon-collector problem*).

Rappelons qu'il y a au plus un échange de message d'un noeud s à noeud r à chaque pas de temps t. Ainsi,

$$\forall i, \quad a_i(t+1) = \begin{cases} a_i(t)+1 & \text{si} \quad s \in A_i(t) \text{ et } r \notin A_i(t) \\ a_i(t) & \text{sinon} \end{cases} \tag{5.131}$$

Définissons

$$\alpha_k = \min_{S \subset \mathbf{V}, |S|=k} \mathbb{P}[s \in S, r \notin S] \tag{5.132}$$

On a donc

$$\forall i, \mathbb{P}[a_i(t+1) = a+1 | a(t) = a] \geq \alpha_k \tag{5.133}$$

L'étude de la croissance de chaque a_i se résume donc à l'étude de la chaîne de Markov $a(t)$ définie comme suit :

$$\mathbb{P}[a(t+1) = k+1 \mid a(t) = k] = \alpha_k \tag{5.134}$$

$$\mathbb{P}[a(t+1) = k \mid a(t) = k] = 1 - \alpha_k \tag{5.135}$$

Une borne suffisante sur la valeur T recherchée est donc donnée par le temps necessaire pour passer de $a(0) = 1$ à $a(t) = N$:

$$T = \inf\{t : a(t) = N \mid a(0) = 1\} \tag{5.136}$$

On peut ecrire

$$T = \sum_{k=1}^{N-1} T_k \tag{5.137}$$

où T_k dénote le temps passé dans l'état k. $a(t)$ étant homogène, les T_k suivent une loi géométrique de paramètre α_k. Par conséquent

$$\mathbb{E}T = \sum_{k=1}^{N-1} \mathbb{E}T_k \tag{5.138}$$

$$= \sum_{k=1}^{N-1} \frac{1}{\alpha_k} \tag{5.139}$$

Il nous reste à determiner les α_k en question.

Rappelons la définition de la constante de Cheeger $\Phi_C(\mathcal{G})$

$$\Phi_C(\mathcal{G}) = \min_{S \subset V} \frac{|\partial S|}{\min\{d(S), d(V \setminus S)\}} \tag{5.140}$$

Les arêtes étant selectionnées uniformément, on peut assimiler $|\partial S|$ à $\mathbb{P}[s \in S, r \notin S]$ et $d(S)$ à $\mathbb{P}[s \in S]$, ce qui implique

$$\forall S \subset V, \quad \frac{\alpha_k}{\min\{\mathbb{P}[s \in S], \mathbb{P}[s \notin S]\}} \geq \Phi_C(\mathcal{G}) \tag{5.141}$$

Puisque $\alpha_k = \mathbb{P}[s \in A_k, r \notin A_k]$ on obtient

$$\begin{aligned}
\alpha_k &\geq \Phi_C(\mathcal{G}) \min_{|S|=k} \min\{\mathbb{P}[s \in S], \mathbb{P}[s \notin S]\} && (5.142)\\
&= \Phi_C(\mathcal{G}) \min\left\{\frac{k}{N}, \frac{N-k}{N-1}\right\} && (5.143)\\
&\geq \Phi_C(\mathcal{G}) \frac{k(N-k)}{N(N-1)} && (5.144)\\
&\geq \frac{1-\lambda_2(\mathcal{G})}{2} \cdot \frac{k(N-k)}{N(N-1)} && (5.145)
\end{aligned}$$

où la dernière ligne exploite l'inégalité de Cheeger $2\Phi_C(\mathcal{G}) \geq 1 - \lambda_2(\mathcal{G})$.

Combinons (5.139) et (5.145) :

$$\begin{aligned}
\mathbb{E}T &\leq \frac{2}{1-\lambda_2(\mathcal{G})} \sum_{k=1}^{N-1} \frac{N(N-1)}{k(N-k)} && (5.146)\\
&= \frac{2}{1-\lambda_2(\mathcal{G})} (N-1) \sum_{k=1}^{N-1} \frac{N-k+k}{k(N-k)} && (5.147)\\
&= \frac{4(N-1)}{1-\lambda_2(\mathcal{G})} \sum_{k=1}^{N-1} \frac{1}{k} && (5.148)
\end{aligned}$$

On reconnait la serie harmonique $\sum_{k=1}^{n} \frac{1}{k}$ qui est bornée par $1 + \ln n$. On en déduit

$$\begin{aligned}
\mathbb{E}T &\leq \frac{4(N-1)(\ln(N-1)+1)}{1-\lambda_2(\mathcal{G})} && (5.149)\\
&\leq \frac{4N(\ln N+1)}{1-\lambda_2(\mathcal{G})} && (5.150)
\end{aligned}$$

En intégrant cette borne dans (5.125), on obtient une borne inférieure sur notre dénominateur

$$\forall t \geq \frac{4N(\ln N+1)}{(1-\delta)(1-\lambda_2(\mathcal{G}))}, \quad \mathbb{P}\left[Q(t) \geq \frac{1}{N^2} \exp\left\{\frac{8N(\ln N+1)\ln\chi}{(1-\delta)(1-\lambda_2(\mathcal{G}))}\right\}\right] \geq \delta \tag{5.151}$$

4. Combinaison des deux bornes

Rappelons que nous cherchons la plus petite valeur de t pour laquelle

$$\mathbb{P}\left[\frac{P(t)}{Q(t)} \leq \varepsilon\right] \geq \delta \tag{5.152}$$

avec δ une probabilité et ε une erreur arbitraires. Remarquons que pour cela il suffit qu'il existe $q \in \mathbb{R}$ tel que

$$\mathbb{P}\big[P(t) \leq \varepsilon q\big] \geq \frac{\delta+1}{2} \qquad \text{et} \qquad \mathbb{P}\big[Q(t) \geq q\big] \geq \frac{\delta+1}{2} \tag{5.153}$$

En effet, sous ces conditions

$$\begin{aligned}
\mathbb{P}\left[\frac{P(t)}{Q(t)} \leq \varepsilon\right] &\geq \mathbb{P}\big[P(t) \leq \varepsilon q \wedge Q(t) \geq q\big] && (5.154)\\
&= \mathbb{P}\big[P(t) \leq \varepsilon q\big] + \mathbb{P}\big[Q(t) \geq q\big] - \mathbb{P}\big[P(t) \leq \varepsilon q \vee Q(t) \geq q\big] && (5.155)\\
&\geq \mathbb{P}\big[P(t) \leq \varepsilon q\big] + \mathbb{P}\big[Q(t) \geq q\big] - 1 && (5.156)\\
&= \frac{\delta+1}{2} + \frac{\delta+1}{2} - 1 = \delta && (5.157)
\end{aligned}$$

Pour que $\mathbb{P}\big[P(t) \leq \varepsilon q\big] \geq \frac{\delta+1}{2}$, nous savons d'après (5.110) qu'il suffit que

$$t \geq \frac{\ln \varepsilon q - \ln N + \ln \frac{\delta+1}{2}}{\ln \lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top])} = \frac{\ln N - \ln \varepsilon - \ln q - \ln \frac{\delta+1}{2}}{-\ln \lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top])} \tag{5.158}$$

Nous savons également d'après (5.151) que

$$\forall t \geq C, \quad \mathbb{P}\left[Q(t) \geq q\right] \geq \frac{\delta+1}{2} \tag{5.159}$$

$$\text{avec} \quad C = \frac{16N(\ln N + 1)}{(1-\delta)(1-\lambda_2(\mathcal{G}))} \qquad \text{et} \qquad q = \frac{1}{N^2}\exp(2C \ln \chi) \tag{5.160}$$

On en déduit que pour que $\mathbb{P}\left[\frac{P(t)}{Q(t)} \leq \varepsilon\right] \geq \delta$, il suffit que

$$\begin{aligned}
t &\geq \max\left\{\frac{\ln N - \ln \varepsilon - \ln\left[\frac{1}{N^2}\exp(2C\ln\chi)\right] - \ln\frac{\delta+1}{2}}{-\ln \lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top])}, C\right\} && (5.161)\\
&= \max\left\{\frac{3\ln N - \ln\varepsilon - 2C\ln\chi - \ln\frac{\delta+1}{2}}{-\ln \lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top])}, C\right\}, && (5.162)
\end{aligned}$$

ce qui conclue notre démonstration. □

E. Theoreme 7 : Analyse de convergence d'AGAVG

Dans cette section, nous considérons l'algorithme AGAVG défini sur un graphe complet, dont la suite de matrices iid $(\mathbf{K}(t))_{t\geq 1}$ est définie par

$$\mathbf{K}(t) = \mathbf{I} - \frac{1}{2}\mathbf{e}_{i(t)}(\mathbf{e}_{i(t)} - \mathbf{e}_{j(t)})^\top, \quad \text{où} \quad (i(t), j(t)) \sim \mathcal{U}_{\{1,\dots,N\}^2} \tag{5.163}$$

Calculons $\lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top])$. On a

$$\mathbb{E}[\mathbf{K}\mathbf{K}^\top] = \frac{1}{N^2}\sum_{ij}\left(\mathbf{I} - \frac{1}{2}\mathbf{e}_i(\mathbf{e}_i - \mathbf{e}_j)^\top\right) \tag{5.164}$$

$$\mathbb{E}[\mathbf{K}\mathbf{K}^\top] = \left(1 - \frac{1}{2N}\right)\mathbf{I} + \frac{1}{2N^2}\mathbf{1}\mathbf{1}^\top \tag{5.165}$$

En utilisant le lemme 4, on peut alors déterminer la seconde valeur propre de $\mathbb{E}[\mathbf{K}\mathbf{K}^\top]$:

$$\lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top]) = 1 - \frac{1}{2N} \tag{5.166}$$

$$\frac{1}{\ln \lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top])} = \frac{1}{\ln(1 - \frac{1}{2N})} \tag{5.167}$$

$$\frac{1}{\ln \lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top])} \leq -2N \qquad \textit{(Lemme 5)} \tag{5.168}$$

$$\frac{1}{-\ln \lambda_2(\mathbb{E}[\mathbf{K}\mathbf{K}^\top])} \geq 2N \tag{5.169}$$

Dans le cas particulier du graphe complet, on peut trouver une expression simplifiée de C. En effet, les α_k définis par l'équation (5.132) ont la forme explicite suivante

$$\alpha_k = \mathbb{P}[i(t) \in \mathcal{A}(t), j(t) \notin \mathcal{A}(t) \mid a(t) = k] = \frac{k(N-k)}{N(N-1)} \tag{5.170}$$

On a alors

$$\mathbb{E}T = \sum_{k=1}^{N-1}\frac{1}{\alpha_k} = \sum_{k=1}^{N-1}\frac{N(N-1)}{k(N-k)} \tag{5.171}$$

$$= (N-1)\sum_{k=1}^{N-1}\frac{N-k+k}{k(N-k)} \tag{5.172}$$

$$= 2(N-1)\sum_{k=1}^{N-1}\frac{1}{k} \tag{5.173}$$

$$\leq 2N(\ln N + 1) \tag{5.174}$$

En intégrant ce resultat dans (5.125), et en observant que $\chi = 1/2$, on obtient

$$\forall \delta \in]0,1[, \quad \forall t \geq \frac{4N(\ln N + 1)}{1-\delta}, \quad \mathbb{P}\left[Q(t) \geq \frac{1}{N^2}\exp\left\{\frac{8N(\ln N + 1)\ln 2}{\delta - 1}\right\}\right] \geq \frac{\delta + 1}{2} \tag{5.175}$$

En reprenant (5.110), on obtient une borne suffisante sur t :

$$t \geq \max\left\{2N\left(3\ln N - \ln\varepsilon + \frac{8\ln 2 \cdot N(\ln N + 1)}{1-\delta} + \ln\frac{2}{1-\delta}\right), \frac{4N(\ln N + 1)}{1-\delta}\right\} \tag{5.176}$$

Comme $4 \leq 8\ln 2 \leq 6$, le premier terme domine toujours le second. La borne suffisante devient ainsi

$$t \geq 2N\left(3\ln N + \frac{6N(\ln N + 1)}{1-\delta} - \ln\varepsilon + \ln\frac{2}{1-\delta}\right) \tag{5.177}$$

F. Théorème 8 : Comportement des protocoles Sum-Weight en présence d'un sous graphe absorbant

Soit un graphe connexe $\mathcal{G}$ possédant un unique sous-graphe absorbant fortement connexe $\mathcal{A}$, sur lequel on exécute un protocole Gossip Sum-Weight. Tout noeud i de $\mathcal{G}$ satisfait

$$\begin{cases} \lim_{t\to\infty} \frac{x_i(t)}{w_i(t)} = \frac{\sum_{i\in\mathcal{G}} x_i(0)}{\sum_{i\in\mathcal{G}} w_i(0)} \equiv \bar{x} & \text{si } i \text{ est dans } \mathcal{A} \\ \lim_{t\to\infty} x_i(t) = \lim_{t\to\infty} w_i(t) = 0 & \text{sinon} \end{cases} \tag{5.178}$$

Démonstration. Puisque $\mathcal{G}$ n'est pas fortement connexe, $\mathbb{E}\mathbf{K}$ est réductible. Elle peut donc se mettre sous la forme bloc-triangulaire supérieure suivante (dite forme normale) :

$$\mathbb{E}\mathbf{K} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{C}_{1,2} & \mathbf{C}_{1,3} & \dots & \mathbf{C}_{1,n} \\ \mathbf{0} & \mathbf{A}_2 & \mathbf{C}_{2,3} & \dots & \mathbf{C}_{2,n} \\ \mathbf{0} & \dots & \ddots & \dots & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{A}_i & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{A}_n \end{bmatrix} \tag{5.179}$$

où tous les blocs $\mathbf{A}_i$ sont irréductibles. Rappelons que $\mathbb{E}\mathbf{K}$ est non-négative et à diagonale strictement positive, donc tous ses blocs le sont également. $\mathcal{G}$ étant connexe, pour tout $1 \leq i < n$, il existe au moins un bloc $\mathbf{C}_{i,j} \neq \mathbf{0}$. Ainsi, toutes les $\mathbf{A}_i$ sont sous-stochastiques (leurs lignes somment au plus à 1, avec au moins une ligne dont la somme est strictement inférieure à 1), sauf $\mathbf{A}_n$ qui est stochastique et qui correspond à l'ensemble absorbant $\mathcal{A}$. Or, on sait que le rayon spectral de toute matrice sous-stochastique est strictement inférieur à 1. Donc

$$\forall 1 \leq i < n, \quad \rho(\mathbf{A}_i) < 1 \quad \text{et} \quad \rho(\mathbf{A}_n) = 1 \tag{5.180}$$

De plus, puisque $\mathbf{A}_n$ est irréductible et à diagonale strictement positive, elle est primitive. On a alors

$$\lambda_2(\mathbf{A}_n) < 1 \quad \text{d'où} \quad \exists \mathbf{a} : \lim_{t\to\infty} \mathbf{A}_n^t = \mathbf{1}\mathbf{a}^\top \quad \text{avec} \quad \mathbf{a} > \mathbf{0} \tag{5.181}$$

La forme bloc-triangulaire de $\mathbb{E}\mathbf{K}$ implique que $\mathbb{E}\mathbf{K}^t$ prend la forme suivante :

$$\mathbb{E}\mathbf{K}^t = \begin{bmatrix} \mathbf{A}_1^t & * & * & \dots & * \\ \mathbf{0} & \mathbf{A}_2^t & * & \dots & * \\ \mathbf{0} & \dots & \ddots & \dots & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{A}_i^t & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{A}_n^t \end{bmatrix} \tag{5.182}$$

On sait également que le spectre d'une matrice bloc triangulaire est l'union des spectres de ses blocs diagonaux. On en déduit que

$$\rho(\mathbb{E}\mathbf{K}) = 1 \quad \text{et} \quad \lambda_2(\mathbb{E}\mathbf{K}) < 1 \tag{5.183}$$

$\mathbb{E}\mathbf{K}^t$ converge donc vers une matrice de rang 1, c'est à dire dont toutes les lignes sont égales (en particulier, toutes les lignes sont égales à la dernière ligne dont la forme est donnée par (5.182)) :

$$\lim_{t\to\infty} \mathbb{E}\mathbf{K}^t - \mathbf{1}\mathbf{v}^\top = \lim_{t\to\infty} \mathbb{E}\Big[\prod_{\tau=1}^{t} \mathbf{K}(\tau) - \mathbf{1}\mathbf{v}^\top\Big] = \mathbf{0}, \quad \text{où} \quad \mathbf{v} \equiv [0,\dots,0,\mathbf{a}] \tag{5.184}$$

On a ainsi

$$\forall i,\ \lim_{t\to\infty} \mathbf{x}(t) = \lim_{t\to\infty} \mathbf{x}(0)^\top \prod_{\tau=1}^{t} \mathbf{K}(\tau) = \mathbf{x}(0)^\top \mathbf{1}\mathbf{v}^\top = \sum_{i=1}^{N} x_i(0)\mathbf{v}^\top \tag{5.185}$$

$$\forall i,\ \lim_{t\to\infty} \mathbf{w}(t) = \lim_{t\to\infty} \mathbf{w}(0)^\top \prod_{\tau=1}^{t} \mathbf{K}(\tau) = \mathbf{w}(0)^\top \mathbf{1}\mathbf{v}^\top = \sum_{i=1}^{N} w_i(0)\mathbf{v}^\top \tag{5.186}$$

La convergence vers le consensus est ainsi garantie pour tous les noeuds i tels que $v_i \neq 0$:

$$\forall i \notin \mathcal{A}, \quad \lim_{t\to\infty} x_i(t) = \lim_{t\to\infty} w_i(t) = 0 \tag{5.187}$$

$$\forall i \in \mathcal{A}, \quad \lim_{t\to\infty} \frac{x_i(t)}{w_i(t)} = \frac{v_i \sum_{j=1}^{N} x_j(0)}{v_i \sum_{j=1}^{N} w_j(0)} = \bar{x} \tag{5.188}$$

□

G. AGKM et AGEM

G1. Lemme 1 : Conservation du dictionnaire consensus

A tout instant t et pour toute catégorie k, l'égalité suivante est vérifiée :

$$\frac{\sum_i^N \mathbf{s}_k^{(i)}(t)}{\sum_i^N w_k^{(i)}(t)} = \boldsymbol{\mu}_k^\star(t) \tag{5.189}$$

Démonstration. L'initialisation d'AGKM (lignes 1-2) définit une relation explicite entre les estimées locales $(\mathbf{s}_k, w_k)$ et le dictionnaire consensus $\mathcal{M}^\star$. En effet, pour toute catégorie k,

$$\sum_j^N \mathbf{s}_k^{(j)}(0) = \sum_j^N \sum \{\mathbf{x} \in \mathcal{C}_k^{(j)}(0)\} = \sum \{\mathbf{x} \in \mathcal{C}_k^\star(0)\} \tag{5.190}$$

$$\text{et} \quad \sum_j^N w_k^{(j)}(0) = \sum_j^N |\mathcal{C}_k^{(j)}(0)| = |\mathcal{C}_k^\star(0)| \tag{5.191}$$

d'où

$$\frac{\sum_j^N \mathbf{s}_k^{(j)}(0)}{\sum_j^N w_k^{(j)}(0)} = \frac{1}{|\mathcal{C}_k^\star(0)|} \sum \{\mathbf{x} \in \mathcal{C}_k^\star(0)\} = \boldsymbol{\mu}_k^\star(0) \tag{5.192}$$

Rappelons que $\mathcal{M}^\star$ n'est modifié que lors d'événements de partitionnement. Lors du p-ième événement de partitionnement, un certain nœud i modifie à la fois sa partition locale $\mathcal{C}^{(i)}$ et ses estimées $(\mathbf{s}_k^{(i)}, w_k^{(i)})_k^K$, tandis que les autres nœuds restent inchangés. Supposons alors que (5.190) et (5.191) soient vraies au $(p-1)$-ième partitionnement, c'est à dire

$$\sum_j^N \mathbf{s}_k^{(j)}(p-1) = \sum\{\mathbf{x} \in \mathcal{C}_k^\star(p-1)\} \quad \text{et} \quad \sum_j^N w_k^{(j)}(p-1) = |\mathcal{C}_k^\star(p-1)| \tag{5.193}$$

En rappelant la définition du mécanisme d'autocorrection par (2.22) et (2.23) on obtient

$$\sum_j^N \mathbf{s}_k^{(j)}(p) = \sum_j^N \mathbf{s}_k^{(j)}(p-1) + \sum\{\mathbf{x} \in \mathcal{C}_k^{(i)}(p)\} - \mathbf{s}_k^{(i)}(p-1) \tag{5.194}$$

$$= \sum_{j\neq i}^N \sum\{\mathbf{x} \in \mathcal{C}_k^{(j)}(p-1)\} + \sum\{\mathbf{x} \in \mathcal{C}_k^{(i)}(p)\} \tag{5.195}$$

$$= \sum_j^N \sum\{\mathbf{x} \in \mathcal{C}_k^{(j)}(p)\} = \sum\{\mathbf{x} \in \mathcal{C}_k^\star(p)\} \tag{5.196}$$

et

$$\sum_j^N w_k^{(j)}(p) = \sum_j^N w_k^{(j)}(p-1) + |\mathcal{C}_k^{(i)}(p)| - w_k^{(i)}(p-1) \tag{5.197}$$

$$= \sum_{j\neq i}^N |\mathcal{C}_k^{(j)}(p-1)| + |\mathcal{C}_k^{(i)}(p)| = \sum_j^N |\mathcal{C}_k^{(j)}(p)| = |\mathcal{C}_k^\star(p)| \tag{5.198}$$

Par récurrence sur p et sachant que (5.193) est vraie pour $p=0$, (1) est vérifiée pour tout événement de partitionnement p. Ce résultat se généralise sans difficulté à tout instant t en rappelant que $\mathcal{M}^\star(t)$ n'est pas modifié par les événements de mise à jour de dictionnaire, et en remarquant qu'ils conservent la sommes des estimées $(\mathbf{s}_k, w_k)$. En notant (s,r) le couple émetteur-récepteur aléatoire sélectionné lors d'une mise à jour à un instant t, on a en effet

$$\frac{\sum_i^N \mathbf{s}_k^{(i)}(t)}{\sum_i^N w_k^{(i)}(t)} = \frac{\sum_{i\neq s}^N \mathbf{s}_k^{(i)}(t-1) + \frac{1}{2}\mathbf{s}_k^{(s)}(t-1) + \frac{1}{2}\mathbf{s}_k^{(s)}(t-1)}{\sum_{i\neq s}^N w_k^{(i)}(t) + \frac{1}{2}w_k^{(s)}(t-1) + \frac{1}{2}w_k^{(s)}(t-1)} = \frac{\sum_i^N \mathbf{s}_k^{(i)}(t-1)}{\sum_i^N w_k^{(i)}(t-1)}, \tag{5.199}$$

ce qui conclue la preuve. □

G2. Lemme 2.36 : Condition suffisante pour la décroissance de J_1

A tout instant t et pour tout nœud i, la partition de Voronoi de $\mathbf{X}^{(i)}$ associée à $\mathcal{M}^{(i)}(t)$ est identique à celle associée à $\mathcal{M}^\star(t)$ si

$$J_2(t) \leq \xi_i(t) \equiv \frac{1}{2} \min_{\mathbf{x}\in\mathbf{X}^{(i)}} \left(\|\mathbf{x} - 1_{\text{NN}}(\mathbf{x}, \mathcal{M}^{(i)}(t))\|_2^2 - \|\mathbf{x} - 2_{\text{NN}}(\mathbf{x}, \mathcal{M}^{(i)}(t))\|_2^2 \right),$$

où $k_{\text{NN}}(\mathbf{x}, \mathcal{A})$ désigne le k-plus proche voisin de $\mathbf{x}$ dans l'ensemble $\mathcal{A}$.

Démonstration. Pour tout échantillon local $\mathbf{x} \in \mathbf{X}^{(i)}$, soient a et b les indices respectifs du plus proche voisin et du second plus proche voisin de $\mathbf{x}$ dans $\mathcal{M}^{(i)}(t)$. Supposons que

$$J_2(t) \leq \frac{1}{2}(\|\mathbf{x} - \boldsymbol{\mu}_b^{(i)}(t)\|_2^2 - \|\mathbf{x} - \boldsymbol{\mu}_a^{(i)}(t)\|_2^2)$$

Alors $\forall k \neq a,\ \|\mathbf{x} - \boldsymbol{\mu}_a^{(i)}(t)\|_2^2 + J_2(t) \leq \|\mathbf{x} - \boldsymbol{\mu}_k^{(i)}(t)\|_2^2 - J_2(t)$. Par définition de J_2, on a trivialement $\forall l, J_2(t) \geq \|\boldsymbol{\mu}_l^{(i)}(t) - \boldsymbol{\mu}_l^{\star}(t)\|_2^2$. En utilisant les inégalités triangulaires on obtient

$$\begin{aligned}
\|\mathbf{x} - \boldsymbol{\mu}_a^{(i)}(t)\| + \epsilon &\geq \|\mathbf{x} - \boldsymbol{\mu}_a^{(i)}(t)\|_2^2 + \|\boldsymbol{\mu}_a^{(i)}(t) - \boldsymbol{\mu}_a^{\star}(t)\|_2^2 \\
&\geq \|\mathbf{x} - \boldsymbol{\mu}_a^{\star}(t)\|_2^2 \\
\|\mathbf{x} - \boldsymbol{\mu}_k^{(i)}(t)\| - \epsilon &\leq \|\mathbf{x} - \boldsymbol{\mu}_k^{(i)}(t)\|_2^2 - \|\boldsymbol{\mu}_k^{(i)}(t) - \boldsymbol{\mu}_k^{\star}(t)\|_2^2 \\
&\leq \|\mathbf{x} - \boldsymbol{\mu}_k^{\star}(t)\|_2^2
\end{aligned}$$

Par conséquent, $\|\mathbf{x} - \boldsymbol{\mu}_a^{\star}(t)\|_2^2 \leq \|\mathbf{x} - \boldsymbol{\mu}_k^{\star}(t)\|_2^2$, ce qui implique que les partitions de Voronoi de $\mathbf{X}^{(i)}$ associées à $\mathcal{M}^{(i)}(t)$ et à $\mathcal{M}^{\star}(t)$ sont identiques. □

Bibliographie

[1] Abbasi, A. A. and Younis, M. (2007). A survey on clustering algorithms for wireless sensor networks. *Computer communications*, 30(14) :2826–2841. 69

[2] Agarwal, A., Chapelle, O., Dudík, M., and Langford, J. (2014). A reliable effective terascale linear learning system. *The Journal of Machine Learning Research*, 15(1) :1111–1133. 25

[3] Agarwal, A. and Duchi, J. C. (2011). Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 873–881. 24, 140

[4] Agarwal, A., Wainwright, M. J., and Duchi, J. C. (2010). Distributed dual averaging in networks. In *Advances in Neural Information Processing Systems*, pages 550–558. 135, 140

[5] Agrawal, R. and Srikant, R. (2000). Privacy-preserving data mining. In *ACM Sigmod Record*, volume 29, pages 439–450. ACM. 151

[6] Ailon, N., Jaiswal, R., and Monteleoni, C. (2009). Streaming k-means approximation. *Advances in Neural Information Processing Systems*, 22 :10–18. 64

[7] Aizerman, M., Braverman, E. M., and Rozonoer, L. (1964). Theoretical foundations of the potential function method in pattern recognition. *Avtomat. i Telemekh*, 25(6) :917–936. 128

[8] Akelbek, M. and Kirkland, S. (2009). Coefficients of ergodicity and the scrambling index. *Linear Algebra and its Applications*, 430(4) :1111–1130. 31

[9] Almendral, J. A. and Díaz-Guilera, A. (2007). Dynamical and spectral properties of complex networks. *New Journal of Physics*, 9(6) :187. 30

[10] Amari, S.-i. (1977). Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological cybernetics*, 27(2) :77–87. 67

[11] Amari, S.-I., Park, H., and Fukumizu, K. (2000). Adaptive method of realizing natural gradient learning for multilayer perceptrons. *Neural Computation*, 12(6) :1399–1409. 130

[12] Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American mathematical society*, pages 337–404. 128

[13] Arthur, D. and Vassilvitskii, S. (2007). k-means++ : The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics. 67

[14] Avila, S., Thome, N., Cord, M., Valle, E., et al. (2011). Bossa : Extended bow formalism for image classification. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 2909–2912. IEEE. 154

[15] Bandyopadhyay, S., Giannella, C., Maulik, U., Kargupta, H., Liu, K., and Datta, S. (2006). Clustering distributed data streams in peer-to-peer environments. *Inf. Sci.*, 176(14) :1952–1985. 71, 72

[16] Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *science*, 286(5439) :509–512. 33, 89, 122

[17] Belkin, M. and Niyogi, P. (2001). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, volume 14, pages 585–591. 99

[18] Benezit, F., Blondel, V., Thiran, P., Tsitsiklis, J., and Vetterli, M. (2010). Weighted gossip : Distributed averaging using non-doubly stochastic matrices. In *IEEE International Symposium on Information Theory*, pages 1753–1757. IEEE. 45, 111

[19] Benezit, F., Thiran, P., and Vetterli, M. (2009). Interval consensus : from quantized gossip to voting. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 3661–3664. IEEE. 49

[20] Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1) :1–127. 156

[21] Bengio, Y. and Monperrus, M. (2005). Non-local manifold tangent learning. *Advances in Neural Information Processing Systems*, 17 :129–136. 102

[22] Bennett, K. P. and Bredensteiner, E. J. (2000). Duality and geometry in svm classifiers. In *ICML*, pages 57–64. 133

[23] Berkhin, P. (2006). A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer. 64

[24] Bertrand, A. and Moonen, M. (2014). Distributed adaptive estimation of covariance matrix eigenvectors in wireless sensor networks with application to distributed pca. *Signal Processing*, 104 :120–135. 104, 105

[25] Bianchi, P., Fort, G., and Hachem, W. (2013). Performance of a distributed stochastic approximation algorithm. *Information Theory, IEEE Transactions on*, 59(11) :7405–7418. 135

[26] Bianchi, P., Hachem, W., and Iutzeler, F. (2014). A stochastic coordinate descent primal-dual algorithm and applications. In *Machine Learning for Signal Processing (MLSP), 2014 IEEE International Workshop on*, pages 1–6. IEEE. 132

[27] Birkhoff, G. (1957). Extensions of jentzsch's theorem. *Transactions of the American Mathematical Society*, pages 219–227. 53

[28] Bishop, C. M. et al. (2006). *Pattern recognition and machine learning*, volume 4. springer New York. 95

[29] Blatt, D., Hero, A. O., and Gauchman, H. (2007). A convergent incremental gradient method with a constant step size. *SIAM Journal on Optimization*, 18(1) :29–51. 131

[30] Bordes, A., Bottou, L., and Gallinari, P. (2009). Sgd-qn : Careful quasi-newton stochastic gradient descent. *The Journal of Machine Learning Research*, 10 :1737–1754. 130

[31] Bordes, A., Ertekin, S., Weston, J., and Bottou, L. (2005). Fast kernel classifiers with online and active learning. *The Journal of Machine Learning Research*, 6 :1579–1619. 128, 129, 134

[32] Bottou, L. (2010a). Large-scale machine learning with stochastic gradient descent. In *International Conference on Computational Statistics*. 129, 137

[33] Bottou, L. (2010b). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer. 140, 143

[34] Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural Networks : Tricks of the Trade*, pages 421–436. Springer. 132, 136, 142

[35] Bottou, L. and Bousquet, O. (2008). The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems*, volume 20, pages 161–168. 130

[36] Bourgain, J. and Yehudayoff, A. (2012). Monotone expansion. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1061–1078. ACM. 32

[37] Boyd, S., Diaconis, P., and Xiao, L. (2004). Fastest mixing markov chain on a graph. *SIAM review*, 46(4) :667–689. 32

[38] Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D. (2005). Gossip algorithms : Design, analysis and applications. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1653–1664. IEEE. 49

[39] Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D. (2006). Randomized gossip algorithms. *Information Theory, IEEE Transactions on*, 52(6) :2508–2530. 20, 34, 37, 171

[40] Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1) :1–122. 134

[41] Brand, M. (2002). Charting a manifold. In *Advances in neural information processing systems*, pages 961–968. 99, 100

[42] Brauer, F. (2008). Compartmental models in epidemiology. In *Mathematical epidemiology*, pages 19–79. Springer. 34

[43] Bruna, J. and Mallat, S. (2013). Invariant scattering convolution networks. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8) :1872–1886. 157

[44] Bruneau, P., Gelgon, M., and Picarougne, F. (2010a). Aggregation of probabilistic PCA mixtures with a variational-bayes technique over parameters. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 702–705. IEEE. 71, 105

[45] Bruneau, P., Gelgon, M., and Picarougne, F. (2010b). Aggregation of probabilistic PCA mixtures with a variational-bayes technique over parameters. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 702–705. IEEE. 92, 123

[46] Buhl, J., Sumpter, D. J., Couzin, I. D., Hale, J. J., Despland, E., Miller, E., and Simpson, S. J. (2006). From disorder to order in marching locusts. *Science*, 312(5778) :1402–1406. 34

[47] Bunte, K., Haase, S., Biehl, M., and Villmann, T. (2012). Stochastic neighbor embedding (sne) for dimension reduction and visualization using arbitrary divergences. *Neurocomputing*, 90 :23–45. 98

[48] Carpenter, G. A., Grossberg, S., and Rosen, D. (1991). Art 2-a : An adaptive resonance algorithm for rapid category learning and recognition. In *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, volume 2, pages 151–156. IEEE. 68

[49] Carroll, J. E. (2004). Birkhoffś contraction coefficient. *Linear algebra and its applications*, 389 :227–234. 53

[50] Casey, M., Veltkamp, R., Goto, M., Leman, M., Rhodes, C., Slaney, M., et al. (2008). Content-based music information retrieval : Current directions and future challenges. *Proceedings of the IEEE*, 96(4) :668–696. 149

[51] Cauchy, A. (1847). Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847) :536–538. 129

[52] Chang, C.-C. and Lin, C.-J. (2001). Libsvm : A library for support vector machines. $http : //www.csie.ntu.edu.tw/sim_cjlin/libsvm$. 128

[53] Chang, E. Y., Bai, H., Zhu, K., Wang, H., Li, J., and Qiu, Z. (2012). Psvm : Parallel support vector machines with incomplete cholesky factorization. *Scaling up Machine Learning : Parallel and Distributed Approaches*. 134

[54] Chang, K. C., Pearson, K., Zhang, T., et al. (2008). Perron-frobenius theorem for nonnegative tensors. *Commun. Math. Sci*, 6(2) :507–520. 55

[55] Chapelle, O. (2007). Training a support vector machine in the primal. *Neural Computation*, 19(5) :1155–1178. 129, 130

[56] Cheeger, J. (1970). A lower bound for the smallest eigenvalue of the laplacian. *Problems in analysis*, 625 :195–199. 31

[57] Chen, J.-J., Hu, C.-J., and Su, C.-R. (2008). Scalable retrieval and mining with optimal peer-to-peer configuration. *Multimedia, IEEE Transactions on*, 10(2) :209–220. 151

[58] Chung, F. (2005). Laplacians and the cheeger inequality for directed graphs. *Annals of Combinatorics*, 9(1) :1–19. 31

[59] Cohen, J. E. and Sellers, P. H. (1982). Sets of nonnegative matrices with positive inhomogeneous products. *Linear Algebra and its Applications*, 47 :185–192. 29, 34

[60] Collobert, R. and Bengio, S. (2001). Svmtorch : Support vector machines for large-scale regression problems. *The Journal of Machine Learning Research*, 1 :143–160. 128

[61] Cortes, C. and Vapnik, V. (1995a). Support-vector networks. In *Machine Learning*, pages 273–297. 95

[62] Cortes, C. and Vapnik, V. (1995b). Support-vector networks. *Machine learning*, 20(3) :273–297. 128

[63] Cox, T. F. and Cox, M. A. (2000). *Multidimensional scaling*. CRC Press. 95

[64] Datta, S., Bhaduri, K., Giannella, C., Wolff, R., and Kargupta, H. (2006a). Distributed data mining in peer-to-peer networks. *Internet Computing, IEEE*, 10(4) :18–26. 69

[65] Datta, S., Giannella, C., and Kargupta, H. (2006b). K-means clustering over a large, dynamic network. In *SDM*. 71, 72

[66] Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., Le, Q. V., et al. (2012). Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, pages 1223–1231. 24, 156

[67] Defazio, A., Bach, F., and Lacoste-Julien, S. (2014). Saga : A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654. 132, 137

[68] Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D., and Terry, D. (1987). Epidemic algorithms for replicated database maintenance. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1–12. ACM. 34

[69] DeMers, D., Cottrell, G., et al. (1993). Non-linear dimensionality reduction. *Advances in neural information processing systems*, pages 580–580. 98

[70] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38. 62, 65, 99

[71] Diaconis, P. (1988). Group representations in probability and statistics. *Lecture Notes-Monograph Series*, pages i–192. 31

[72] Diaconis, P. and Stroock, D. (1991). Geometric bounds for eigenvalues of markov chains. *The Annals of Applied Probability*, pages 36–61. 31

[73] Dimakis, A. G., Kar, S., Moura, J. M., Rabbat, M. G., and Scaglione, A. (2010). Gossip algorithms for distributed signal processing. *Proceedings of the IEEE*, 98(11) :1847–1864. 49

[74] Dobrushin, R. (1956). Central limit theorem for nonstationary markov chains. ii. *Theory of Probability & Its Applications*, 1(4) :329–383. 35

[75] Dong, J.-x., Krzyżak, A., and Suen, C. (2003). A fast parallel optimization for training support vector machine. In *Machine Learning and Data Mining in Pattern Recognition*, pages 96–105. Springer. 134

[76] Donoho, D. L. and Grimes, C. (2003). Hessian eigenmaps : Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 100(10) :5591–5596. 98

[77] Duchi, J. C., Agarwal, A., and Wainwright, M. J. (2012). Dual averaging for distributed optimization : convergence analysis and network scaling. *Automatic Control, IEEE Transactions on*, 57(3) :592–606. 131, 135, 140

[78] Durut, M., Patra, B., and Rossi, F. (2012). A discussion on parallelization schemes for stochastic vector quantization algorithms. *CoRR*, abs/1205.2282. 70, 72, 73, 75

[79] Eckart, C. and Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3) :211–218. 95

[80] Eisenhardt, M., Muller, W., and Henrich, A. (2003). Classifying documents by distributed p2p clustering. *GI Jahrestagung*, pages 286–291. 70, 71, 72

[81] Elhashash, A. and Szyld, D. B. (2008). Generalizations of m-matrices which may not have a nonnegative inverse. *Linear Algebra and its Applications*, 429(10) :2435–2450. 55

[82] Erdős, P. and Rényi, A. (1959). On random graphs. *Publicationes Mathematicae Debrecen*, 6 :290–297. 32

[83] Erosheva, E. A. and Fienberg, S. E. (2005). Bayesian mixed membership models for soft clustering and classification. In *Classification—The Ubiquitous Challenge*, pages 11–26. Springer. 67

[84] Eugster, P. T., Guerraoui, R., Kermarrec, A.-M., and Massoulié, L. (2004). Epidemic information dissemination in distributed systems. *Computer*, 37(5) :60–67. 34

[85] Everett, B. (2013). *An introduction to latent variable models*. Springer Science & Business Media. 95

[86] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2007). The PASCAL Visual Object Classes Challenge 2007 Results. http ://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. 161

[87] Fagnani, F. and Zampieri, S. (2008). Asymmetric randomized gossip algorithms for consensus. In *Proc. IFAC world congress*, pages 9051–9056. 45

[88] Farabet, C., LeCun, Y., Kavukcuoglu, K., Culurciello, E., Martini, B., Akselrod, P., and Talay, S. (2011). Large-scale fpga-based convolutional networks. *Machine Learning on Very Large Data Sets*, 1. 156

[89] Fatta, G. D., Blasa, F., Cafiero, S., and Fortino, G. (2011). Epidemic k-means clustering. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 151–158. IEEE. Print ISBN : 9781467300056 Issue Date : 11-11 Dec. 2011 On page(s) : 151 - 158. 72, 89

[90] Fatta, G. D., Blasa, F., Cafiero, S., and Fortino, G. (2012). Fault tolerant decentralised k-

means clustering for asynchronous large-scale networks. *Journal of Parallel and Distributed Computing*. 72, 74, 75, 76

[91] Fellus, J., Picard, D., and Gosselin, P.-H. (2013a). Calcul décentralisé de dictionnaires visuels pour l'indexation multimédia dans les bases de données réparties sur les réseaux. In *ORASIS : Orasis, Congrès des jeunes chercheurs en vision par ordinateur*. 73

[92] Fellus, J., Picard, D., and Gosselin, P.-H. (2013b). Decentralized k-means using randomized gossip protocols for clustering large datasets. In *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*, pages 599–606. IEEE. 73, 78

[93] Fellus, J., Picard, D., and Gosselin, P.-H. (2015a). Asynchronous decentralized convex optimization through short-term gradient averaging. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. 107, 136

[94] Fellus, J., Picard, D., and Gosselin, P.-H. (2015b). Indexation multimédia par dictionnaires visuels en environnement décentralisé : Une approche par protocoles gossip. *Traitement du Signal*. 73, 78

[95] Fellus, J., Picard, D., Gosselin, P.-H., et al. (2013c). Dimensionality reduction in decentralized networks by gossip aggregation of principal components analyzers. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 171–176. 107

[96] Fine, S. and Scheinberg, K. (2002). Efficient svm training using low-rank kernel representations. *The Journal of Machine Learning Research*, 2 :243–264. 134

[97] Flouri, K., Beferull-Lozano, B., and Tsakalides, P. (2008). Distributed consensus algorithms for svm training in wireless sensor networks. In *Signal Processing Conference, 2008 16th European*, pages 1–5. IEEE. 133

[98] Forero, P. A., Cano, A., and Giannakis, G. B. (2010). Consensus-based distributed support vector machines. *The Journal of Machine Learning Research*, 11 :1663–1707. 134

[99] Forero, P. A., Cano, A., and Giannakis, G. B. (2011). Distributed clustering using wireless sensor networks. *Selected Topics in Signal Processing, IEEE Journal of*, 5(4) :707–724. 71, 72

[100] Francis, J. G. (1961). The qr transformation a unitary analogue to the lr transformation—part 1. *The Computer Journal*, 4(3) :265–271. 102

[101] Gabay, D. (1983). Chapter ix applications of the method of multipliers to variational inequalities. *Studies in mathematics and its applications*, 15 :299–331. 134

[102] Gelfand, I. (1941). Normierte ringe. *Matematicheskii Sbornik*, 9(1) :3–24. 30

[103] Ghahramani, Z., Beal, M. J., et al. (1999). Variational inference for bayesian mixtures of factor analysers. In *NIPS*, pages 449–455. 66

[104] Ghahramani, Z., Hinton, G. E., et al. (1996). The em algorithm for mixtures of factor

analyzers. Technical report, Technical Report CRG-TR-96-1, University of Toronto. 66, 92, 99

[105] Gionis, A., Indyk, P., Motwani, R., et al. (1999). Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529. 153

[106] Glowinski, R. and Marroco, A. (1975). Sur l'approximation, par elements finis d'ordre un, et la resolution, par penalisation-dualite d'une classe de problemes de dirichlet non lineaires. *ESAIM : Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, 9(R2) :41–76. 134

[107] Goh, A. and Vidal, R. (2008). Clustering and dimensionality reduction on riemannian manifolds. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–7. IEEE. 64

[108] Goldreich, O. (2011). Basic facts about expander graphs. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 451–464. Springer. 32

[109] Golub, G. H. and Van Loan, C. F. (2012). *Matrix computations*, volume 3. JHU Press. 101

[110] Gosselin, P.-H., Murray, N., Jégou, H., and Perronnin, F. (2014). Revisiting the fisher vector for fine-grained classification. *Pattern Recognition Letters*, 49 :92–98. 156

[111] Graf, H. P., Cosatto, E., Bottou, L., Dourdanovic, I., and Vapnik, V. (2004). Parallel support vector machines : The cascade svm. In *Advances in neural information processing systems*, pages 521–528. 134

[112] Greenberg, A., Hamilton, J., Maltz, D. A., and Patel, P. (2008). The cost of a cloud : research problems in data center networks. *ACM SIGCOMM computer communication review*, 39(1) :68–73. 151

[113] Grimmett, G. (1999). *What is Percolation ?* Springer. 34

[114] Gunnemann, S. and Faloutsos, C. (2013). Mixed membership subspace clustering. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 221–230. IEEE. 67, 92

[115] Guyon, I., Boser, B., and Vapnik, V. (1993). Automatic capacity tuning of very large vc-dimension classifiers. *Advances in neural information processing systems*, pages 147–147. 128

[116] Hajnal, J. (1976). On products of non-negative matrices. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 79, pages 521–530. Cambridge Univ Press. 34

[117] Hearst, M. A., Dumais, S. T., Osman, E., Platt, J., and Scholkopf, B. (1998). Support vector machines. *Intelligent Systems and their Applications, IEEE*, 13(4) :18–28. 95

[118] Hensel, C. and Dutta, H. (2009). Gadget svm : a gossip-based sub-gradient svm solver. In *International Conference on Machine Learning (ICML), Numerical Mathematics in Machine Learning Workshop*. 135

[119] Hinton, G. E. and Roweis, S. T. (2002). Stochastic neighbor embedding. In *Advances in neural information processing systems*, pages 833–840. 97

[120] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786) :504–507. 98

[121] Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301) :13–30. 176

[122] Hofmann, T., Schölkopf, B., and Smola, A. J. (2008). Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220. 95

[123] Hong, M. and Luo, Z.-Q. (2012). On the linear convergence of the alternating direction method of multipliers. *arXiv preprint arXiv :1208.3922*. 135

[124] Hopf, E. (1963). An inequality for positive integral linear operators. *Journal of Mathematics & Mechanics*, 12(5) :683–692. 53

[125] Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6) :417. 95

[126] Hu, W., Xie, N., Li, L., Zeng, X., and Maybank, S. (2011). A survey on visual content-based video indexing and retrieval. *Systems, Man, and Cybernetics, Part C. Applications and Reviews, IEEE Transactions on*, 41(6) :797–819. 149

[127] Indyk, P. and Motwani, R. (1998). Approximate nearest neighbors : Towards removing the curse of dimensionality. In *ACM Symposium on Theory of Computing*, pages 604–613. 64

[128] Iutzeler, F., Bianchi, P., Ciblat, P., and Hachem, W. (2013). Asynchronous distributed optimization using a randomized alternating direction method of multipliers. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 3671–3676. IEEE. 134

[129] Iutzeler, F., Bianchi, P., Ciblat, P., and Hachem, W. (2014). Linear convergence rate for distributed optimization with the alternating direction method of multipliers. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 5046–5051. IEEE. 135

[130] Iutzeler, F., Ciblat, P., and Hachem, W. (2012). Analysis of sum-weight-like algorithms for averaging in wireless sensor networks. *CoRR*, abs/1209.5912. 37, 43, 45, 171

[131] Iutzeler, F. and Hendrickx, J. (2015). Online relaxation method for improving linear convergence rates of the admm. In *34 th Benelux Meeting on Systems and Control*, page 20. 135

[132] Ivanciuc, O. (2007). Applications of support vector machines in chemistry. *Reviews in computational chemistry*, 23 :291. 128

[133] Jaakkola, T., Haussler, D., et al. (1999). Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems*, pages 487–493. 155

[134] Jain, A. K. (2010). Data clustering : 50 years beyond k-means. *Pattern recognition letters*, 31(8) :651–666. 64

[135] Januzaj, E., Kriegel, H.-P., and Pfeifle, M. (2004). Dbdc : Density based distributed clustering. In *Advances in Database Technology-EDBT 2004*, pages 88–105. Springer. 70, 72

[136] Jegou, H., Douze, M., and Schmid, C. (2008). Hamming embedding and weak geometric consistency for large scale image search. In *European Conference on Computer Vision*, pages 304–317. Springer. 84, 86, 161

[137] Jégou, H., Douze, M., Schmid, C., and Pérez, P. (2010). Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3304–3311. IEEE. 129

[138] Jégou, H., Perronnin, F., Douze, M., Sanchez, J., Perez, P., and Schmid, C. (2012a). Aggregating local image descriptors into compact codes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(9) :1704–1716. 156

[139] Jégou, H., Perronnin, F., Douze, M., Sánchez, J., Pérez, P., and Schmid, C. (2012b). Aggregating local image descriptors into compact codes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1 :3304–3311. QUAERO. 162

[140] Jelasity, M., Kowalczyk, W., and Van Steen, M. (2003). Newscast computing. Technical report, Technical Report IR-CS-006, Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam, The Netherlands. 37, 71, 89

[141] Jelasity, M., Montresor, A., and Babaoglu, O. (2005). Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(3) :219–252. 37

[142] Jerrum, M. and Sinclair, A. (1989). Approximating the permanent. *SIAM journal on computing*, 18(6) :1149–1178. 30

[143] Joachims, T. (1999). Making large scale svm learning practical. Technical report, Universität Dortmund. 128, 134

[144] Karp, R., Schindelhauer, C., Shenker, S., and Vocking, B. (2000). Randomized rumor spreading. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 565–574. IEEE. 34

[145] Kempe, D., Dobra, A., and Gehrke, J. (2003). Gossip-based computation of aggregate information. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '03, pages 482–, Washington, DC, USA. IEEE Computer Society. 34, 43, 45, 72, 104, 135

[146] Kempe, D. and McSherry, F. (2008). A decentralized algorithm for spectral analysis. *Journal of Computer and System Sciences*, 74(1) :70–83. 104, 105, 109

[147] Kermack, W. O. and McKendrick, A. G. (1927). A contribution to the mathematical theory of epidemics. In *Proceedings of the Royal Society of London A : Mathematical, Physical and Engineering Sciences*, volume 115, pages 700–721. The Royal Society. 34

[148] King, I., Ng, C. H., and Sia, K. C. (2004). Distributed content-based visual information retrieval system on peer-to-peer networks. *ACM Transactions on Information Systems (TOIS)*, 22(3) :477–501. 151

[149] Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, 78(9) :1464–1480. 67

[150] Korada, S. B., Montanari, A., and Oh, S. (2011). Gossip PCA. In *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pages 209–220. ACM. 104, 105

[151] Kowalczyk, W. and Vlassis, N. A. (2004). Newscast em. In *Advances in neural information processing systems*, pages 713–720. 71, 72, 73, 74, 75

[152] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105. 156

[153] Kruskal, J. B. (1964). Nonmetric multidimensional scaling : a numerical method. *Psychometrika*, 29(2) :115–129. 96

[154] Kublanovskaya, V. N. (1962). On some algorithms for the solution of the complete eigenvalue problem. *USSR Computational Mathematics and Mathematical Physics*, 1(3) :637–657. 102

[155] Lanczos, C. (1950). *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office. 102

[156] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324. 84, 86, 116, 143

[157] LeCun, Y., Kavukcuoglu, K., and Farabet, C. (2010). Convolutional networks and applications in vision. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 253–256. IEEE. 156

[158] Lee, H., Battle, A., Raina, R., and Ng, A. Y. (2006). Efficient sparse coding algorithms. In *Advances in neural information processing systems*, pages 801–808. 154

[159] Lee, J. A., Peluffo-Ordonez, D. H., and Verleysen, M. (2014). Multiscale stochastic neighbor embedding : Towards parameter-free dimensionality reduction. In *Proceedings of 22st European Symposium on Artificial Neural Networks, Computational Intelligence And Machine Learning (ESANN)*. 98

[160] Lee, J. A., Renard, E., Bernard, G., Dupont, P., and Verleysen, M. (2013). Type 1 and 2 mixtures of kullback–leibler divergences as cost functions in dimensionality reduction based on similarity preservation. *Neurocomputing*, 112 :92–108. 98

[161] Lejsek, H., Jónsson, B. Þ., and Amsaleg, L. (2011). Nv-tree : nearest neighbors at the billion scale. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, page 54. ACM. 153

[162] Li, H., Teng, L., Chen, W., and Shen, I.-F. (2005). Supervised learning on local tangent space. In *Advances in Neural Networks–ISNN 2005*, pages 546–551. Springer. 102

[163] Linde, Y., Buzo, A., and Gray, R. (1980). An algorithm for vector quantizer design. *IEEE Transaction on Communication*, 28 :84–94. 62

[164] Liu, Y., Zhang, D., Lu, G., and Ma, W.-Y. (2007). A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1) :262–282. 149

[165] Lloyd, S. P. (1982). Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28 :129–137. 62

[166] Lovász, L. (1993). Random walks on graphs : A survey. *Combinatorics, Paul erdos is eighty*, 2(1) :1–46. 30, 31, 32

[167] Lyu, S. (2005). Mercer kernels for object recognition with local features. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 223–229. IEEE. 153

[168] Macua, S. V., Belanovic, P., and Zazo, S. (2010). Consensus-based distributed principal component analysis in wireless sensor networks. In *Signal Processing Advances in Wireless Communications (SPAWC), 2010 IEEE Eleventh International Workshop on*, pages 1–5. IEEE. 105, 106

[169] Mahajan, M., Nimbhorkar, P., and Varadarajan, K. (2009). The planar k-means problem is np-hard. In *Proceedings of the 3rd International Workshop on Algorithms and Computation*, WALCOM '09, pages 274–285, Berlin, Heidelberg. Springer-Verlag. 62

[170] Mairal, J., Koniusz, P., Harchaoui, Z., and Schmid, C. (2014). Convolutional kernel networks. In *Advances in Neural Information Processing Systems*, pages 2627–2635. 157

[171] Marée, R., Denis, P., Wehenkel, L., and Geurts, P. (2010). Incremental indexing and distributed image search using shared randomized vocabularies. In *Proceedings of the international conference on Multimedia information retrieval*, pages 91–100. ACM. 151

[172] Markov, A. A. (1906). Extension of the law of large numbers to dependent quantities. *Izv. Fiz.-Matem. Obsch. Kazan Univ.(2nd Ser)*, 15 :135–156. 27

[173] Martinetz, T. M., Berkovich, S. G., and Schulten, K. J. (1993). Neural-gas' network for vector quantization and its application to time-series prediction. *Neural Networks, IEEE Transactions on*, 4(4) :558–569. 68

[174] Mehrotra, S. (1992). On the implementation of a primal-dual interior point method. *SIAM Journal on optimization*, 2(4) :575–601. 128

[175] Meng, Z., Wiesel, A., and Hero, A. O. (2012). Distributed principal component analy-

sis on networks via directed graphical models. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 2877–2880. IEEE. 104

[176] Mercer, J. (1909). Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, pages 415–446. 97, 128

[177] Merton, R. K. (1968). The matthew effect in science. *Science*, 159(3810) :56–63. 91

[178] Meyer, C. D. (2000). *Matrix analysis and applied linear algebra*. Siam. 18, 29

[179] Minsky, M. and Seymour, P. (1969). Perceptrons. 127

[180] müller, W. T., eisenhardt, M., and henrich, A. (2003). Efficient content-based P2P image retrieval using peer content descriptions. In *Internet Imaging V. Edited by Santini, Simone ; Schettini, Raimondo. Proceedings of the SPIE, Volume 5304, pp. 57-68 (2003).*, pages 57–68. 25, 151

[181] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814. 156

[182] Negrel, R., Picard, D., and Gosselin, P.-H. (2013). Web scale image retrieval using compact tensor aggregation of visual descriptors. *IEEE Multimedia*, 99(1) :1. 129

[183] Negrel, R., Picard, D., and Gosselin, P.-H. (9999). Web scale image retrieval using compact tensor aggregation of visual descriptors. *IEEE Transactions on Multimedia*, page En révision. 150

[184] Nesterov, Y. (2009). Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1) :221–259. 125, 131, 135

[185] Newman, M. E., Moore, C., and Watts, D. J. (2000). Mean-field solution of the small-world network model. *Physical Review Letters*, 84(14) :3201. 32

[186] Nguyen, A., Yosinski, J., and Clune, J. (2015). Deep neural networks are easily fooled : High confidence predictions for unrecognizable images. 157

[187] Nikseresht, A. (2008). *Estimation de modèles de mélange probabilistes : une proposition pour un fonctionnement réparti et décentralisé*. PhD thesis, Université de Nantes. 105

[188] Nikseresht, A. and Gelgon, M. (2008). Gossip-based computation of a gaussian mixture model for distributed multimedia indexing. *Multimedia, IEEE Transactions on*, 10(3) :385–392. 71, 72, 105

[189] Noutsos, D. (2006). On perron–frobenius property of matrices having some negative entries. *Linear Algebra and its Applications*, 412(2) :132–153. 55

[190] Ordonez, C., Mohanam, N., and Garcia-Alvarado, C. (2014). Pca for large data sets with parallel data summarization. *Distributed and Parallel Databases*, 32(3) :377–403. 104, 105, 106

[191] Ortega, J. M. and Kaiser, H. F. (1963). The llt and qr methods for symmetric tridiagonal matrices. *The Computer Journal*, 6(1) :99–101. 102

[192] Parsons, L., Haque, E., and Liu, H. (2004). Subspace clustering for high dimensional data : a review. *ACM SIGKDD Explorations Newsletter*, 6(1) :90–105. 67

[193] Patanè, G. and Russo, M. (2001). The enhanced LBG algorithm. *Neural Networks*, 14(9) :1219–1237. 67, 78

[194] Patanè, G. and Russo, M. (2000). Elbg implementation. *International Journal of Knowledge based Intelligent Engineering Systems*, 2 :2–4. 67

[195] Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11) :559–572. 95

[196] Perron, O. (1907). Zur theorie der matrices. *Mathematische Annalen*, 64(2) :248–263. 27

[197] Perronnin, F. and Dance, C. (2007). Fisher kernels on visual vocabularies for image categorization. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE. 129, 155

[198] Perronnin, F., Sánchez, J., and Mensink, T. (2010). Improving the fisher kernel for large-scale image classification. In *Computer Vision–ECCV 2010*, pages 143–156. Springer. 156

[199] Picard, D. and Gosselin, P. (9999). Efficient image signatures and similarities using tensor products of local descriptors. *Computer Vision and Image Understanding*, page En révision. 162

[200] Picard, D. and Gosselin, P.-H. (2013). Efficient image signatures and similarities using tensor products of local descriptors. *Computer Vision and Image Understanding*, 117(6) :680–687. 129, 156

[201] Picard, D., Revel, A., and Cord, M. (2012). An application of swarm intelligence to distributed image retrieval. *Information Sciences*, 192 :71–81. 151, 152

[202] Pittel, B. (1987). On spreading a rumor. *SIAM Journal on Applied Mathematics*, 47(1) :213–223. 34

[203] Platt, J. et al. (1998). Sequential minimal optimization : A fast algorithm for training support vector machines. 128

[204] Pollett, P. and van Doorn, E. (2008). Quasi-stationary distributions for reducible absorbing markov chains in discrete time. 55

[205] Radovanović, M., Nanopoulos, A., and Ivanović, M. (2010). Hubs in space : Popular nearest neighbors in high-dimensional data. *The Journal of Machine Learning Research*, 11 :2487–2531. 64, 102

[206] Rosenblatt, F. (1958). The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6) :386. 127

[207] Roussopoulos, N., Kelley, S., and Vincent, F. (1995). Nearest neighbor queries. In *ACM sigmod record*, volume 24, pages 71–79. ACM. 153

[208] Roux, N. L., Schmidt, M., and Bach, F. R. (2012). A stochastic gradient method with an exponential convergence rate for finite training sets. *NIPS*, pages 2663–2671. 125, 132, 136, 137

[209] Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500) :2323–2326. 98

[210] Sánchez, J., Perronnin, F., Mensink, T., and Verbeek, J. (2013). Image classification with the fisher vector : Theory and practice. *International journal of computer vision*, 105(3) :222–245. 155

[211] Sarwate, A. D. and Dimakis, A. G. (2012). The impact of mobility on gossip algorithms. *Information Theory, IEEE Transactions on*, 58(3) :1731–1742. 49

[212] Schizas, I. D., Giannakis, G. B., Roumeliotis, S., Ribeiro, A., et al. (2008). Consensus in ad hoc wsns with noisy links—part ii : Distributed estimation and smoothing of random signals. *Signal Processing, IEEE Transactions on*, 56(4) :1650–1666. 134

[213] Schmidt, M., Roux, N. L., and Bach, F. (2013). Minimizing finite sums with the stochastic average gradient. *arXiv preprint arXiv :1309.2388*. 132, 137

[214] Schnitzer, D. and Flexer, A. (2014). Choosing the metric in high–dimensional spaces based on hub analysis. In *Proc. 22nd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*. 64

[215] Schölkopf, B., Smola, A., and Müller, K.-R. (1997). Kernel principal component analysis. In *Artificial Neural Networks—ICANN'97*, pages 583–588. Springer. 97

[216] Schraudolph, N. N., Yu, J., and Günter, S. (2007). A stochastic quasi-newton method for online convex optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 436–443. 130

[217] Seneta, E. (1979). Coefficients of ergodicity : structure and applications. *Advances in applied probability*, pages 576–590. 53

[218] Seneta, E. (2006). *Non-negative matrices and Markov chains*. Springer. 27, 28, 34, 35, 53

[219] Shah, D. (2009). *Gossip algorithms*. Now Publishers Inc. 34, 49

[220] Shalev-Shwartz, S., Singer, Y., Srebro, N., and Cotter, A. (2011). Pegasos : Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1) :3–30. 130, 135

[221] Shalev-Shwartz, S. and Tewari, A. (2011). Stochastic methods for l 1-regularized loss minimization. *The Journal of Machine Learning Research*, 12 :1865–1892. 129

[222] Shalev-Shwartz, S. and Zhang, T. (2013). Stochastic dual coordinate ascent methods for regularized loss. *The Journal of Machine Learning Research*, 14(1) :567–599. 128, 129, 131, 132, 135, 136, 142

[223] Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge university press. 153

[224] Sinclair, A. (1992). Improved bounds for mixing rates of markov chains and multicommodity flow. *Combinatorics, probability and Computing*, 1(04) :351–370. 30, 31

[225] Sivic, J. and Zisserman, A. (2003). Video google : A text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1470–1477. IEEE. 154

[226] Suzuki, T. (2013). Dual averaging and proximal gradient descent for online alternating direction multiplier method. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 392–400. 131

[227] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv :1312.6199*. 157

[228] Tahbaz-Salehi, A. and Jadbabaie, A. (2007). Small world phenomenon, rapidly mixing markov chains, and average consensus algorithms. In *Decision and Control, 2007 46th IEEE Conference on*, pages 276–281. IEEE. 32

[229] Tahbaz-Salehi, A. and Jadbabaie, A. (2008). A necessary and sufficient condition for consensus over random networks. *Automatic Control, IEEE Transactions on*, 53(3) :791–795. 55

[230] Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). Deepface : Closing the gap to human-level performance in face verification. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1701–1708. IEEE. 156

[231] Tanenbaum, A. S. and Van Steen, M. (2007). *Distributed systems*. Prentice-Hall. 49

[232] Tangelder, J. W. and Veltkamp, R. C. (2008). A survey of content based 3d shape retrieval methods. *Multimedia tools and applications*, 39(3) :441–471. 149

[233] Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500) :2319–2323. 100

[234] Teng, L., Li, H., Fu, X., Chen, W., Shen, I., et al. (2005). Dimension reduction of microarray data based on local tangent space alignment. In *Cognitive Informatics, 2005.(ICCI 2005). Fourth IEEE Conference on*, pages 154–159. IEEE. 96, 97

[235] Tipping, M. E. and Bishop, C. M. (1999). Mixtures of probabilistic principal component analyzers. *Neural computation*, 11(2) :443–482. 66, 92, 99, 105

[236] Tsianos, K. I., Lawlor, S., and Rabbat, M. G. (2012). Push-sum distributed dual averaging for convex optimization. In *CDC*, pages 5453–5458. 135, 140, 141

[237] Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605) :85. 98

[238] Van der Maaten, L., Postma, E., and Van Den Herik, H. (2009). Dimensionality reduction : A comparative review. *Journal of Machine Learning Research*, 10 :1–41. 94

[239] van Gemert, J. C., Veenman, C. J., Smeulders, A. W. M., and Geusebroek, J. M. (2010). Visual word ambiguity. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(7) :1271–1283. 154

[240] Vapnik, V. (1963). Pattern recognition using generalized portrait method. *Automation and remote control*, 24 :774–780. 127

[241] Vapnik, V. (1998a). *Statistical Learning Theory*. Wiley-Interscience, New York. 18

[242] Vapnik, V. (1998b). *Statistical Learning Theory*. Wiley-Interscience, New York. 126

[243] Vapnik, V. N. and Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2) :264–280. 127

[244] Verhulst, P. (1845). La loi d'accroissement de la population. *Nouv. Mem. Acad. Roy. Soc. Belle-lettr. Bruxelles*, 18 :1. 34

[245] Verykios, V. S., Bertino, E., Fovino, I. N., Provenza, L. P., Saygin, Y., and Theodoridis, Y. (2004). State-of-the-art in privacy preserving data mining. *ACM Sigmod Record*, 33(1) :50–57. 151

[246] Vicsek, T., Czirók, A., Ben-Jacob, E., Cohen, I., and Shochet, O. (1995). Novel type of phase transition in a system of self-driven particles. *Physical review letters*, 75(6) :1226. 34

[247] Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., and Gong, Y. (2010). Locality-constrained linear coding for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3360–3367. IEEE. 154

[248] Watanabe, T. and Masuda, N. (2010). Enhancing the spectral gap of networks by node removal. *Physical Review E*, 82(4) :046102. 30

[249] Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of 'small-world'networks. *nature*, 393(6684) :440–442. 32

[250] Weiss, Y. (1999). Segmentation using eigenvectors : a unifying view. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 975–982. IEEE. 97

[251] Wiesel, A. and Hero, A. O. (2009). Decomposable principal component analysis. *Signal Processing, IEEE Transactions on*, 57(11) :4369–4377. 104

[252] Williams, D. R. G. H. R. and Hinton, G. (1986). Learning representations by back-propagating errors. *Nature*, pages 323–533. 156

[253] Wolfowitz, J. (1963). Products of indecomposable, aperiodic, stochastic matrices. *Proceedings of the American Mathematical Society*, 14(5) :733–737. 55

[254] Xiao, L. (2009). Dual averaging method for regularized stochastic learning and online optimization. In *Advances in Neural Information Processing Systems*, pages 2116–2124. 131, 138, 142

[255] Xiao, L. and Zhang, T. (2014). A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4) :2057–2075. 132

[256] Yang, J., Yu, K., Gong, Y., and Huang, T. (2009). Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1794–1801. IEEE. 154

[257] Yang, T. (2013). Trading computation for communication : Distributed stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 629–637. 135

[258] Zanghirati, G. and Zanni, L. (2003). A parallel solver for large quadratic programs in training support vector machines. *Parallel computing*, 29(4) :535–551. 134

[259] Zanni, L., Serafini, T., and Zanghirati, G. (2006). Parallel software for training large scale support vector machines on multiprocessor systems. *The Journal of Machine Learning Research*, 7 :1467–1492. 134

[260] Zezula, P., Amato, G., Dohnal, V., and Batko, M. (2006). *Similarity search : the metric space approach*, volume 32. Springer Science & Business Media. 152

[261] Zhang, D. and Pal, S. K. (2002). *Neural networks and systolic array design*. World Scientific. 25

[262] Zhang, Z. and Zha, H. (2002). Principal manifolds and nonlinear dimension reduction via local tangent space alignment. In *SIAM Journal of Scientific Computing*. 98

[263] Zukerman, M. (2013). Introduction to queueing theory and stochastic teletraffic models. *arXiv preprint arXiv :1307.2968*. 21

Résumé . Avec l'avènement de « l'ère des données », les besoins des systèmes de traitement de l'information en ressources de calcul ont explosé, dépassant largement les évolutions technologiques des processeurs modernes. Dans le domaine de l'apprentissage statistique en particulier, les paradigmes de calcul massivement distribués représentent la seule alternative praticable.

L'algorithmique distribuée emprunte la plupart de ses concepts à l'algorithmique classique, centralisée et séquentielle, dans laquelle le comportement du système est décrit comme une suite d'instructions exécutées l'une après l'autre. L'importance de la communication entre unités de calcul y est généralement négligée et reléguée aux détails d'implémentation. Or, lorsque le nombre d'unités impliquées augmente, le poids des opérations locales s'efface devant les effets émergents propres aux larges réseaux d'unités. Pour conserver les propriétés désirables de stabilité, de prédictibilité et de programmabilité offertes par l'algorithmique centralisée, les paradigmes de calcul distribué doivent dès lors intégrer cette dimension qui relève de la théorie des graphes.

Cette thèse propose un cadre algorithmique pour l'apprentissage statistique large échelle, qui prévient deux défaut majeurs des méthodes classiques : la centralisation et la synchronisation. Nous présentons ainsi plusieurs algorithmes basés sur des protocoles Gossip décentralisés et asynchrones, applicables aux problèmes de catégorisation, estimation de densité, réduction de dimension, classification et optimisation convexe. Ces algorithmes produisent des solutions identiques à leurs homologues centralisés, tout en offrant une accélération appréciable sur de larges réseaux pour un coût de communication très réduit. Ces qualités pratiques sont démontrées mathématiquement par une analyse de convergence détaillée. Nous illustrons finalement la pertinence des méthodes proposées sur des tâches d'indexation multimédia et de classification d'images.

Abstract . With the advent of the "data era", the amount of computational resources required by information processing systems has exploded, largely exceeding the technological evolutions of modern processors. Specifically, contemporary machine learning applications necessarily resort to massively distributed computation.

Distributed algorithmics borrows most of its concepts from classical centralized and sequential algorithmics, where the system's behavior is defined as a sequence of instructions, executed one after the other. The importance of communication between computation units is generally neglected and pushed back to implementation details. Yet, as the number of units grows, the impact of local operations vanishes behind the emergent effects related to the large network of units. To preserve the desirable properties of centralized algorithmics such as stability, predictability and programmability, distributed computational paradigms must encompass this graph-theoretical dimension.

This thesis proposes an algorithmic framework for large scale machine learning, which prevent two major drawbacks of classical methods, namely *centralization* and *synchronization*. We therefore introduce several new algorithms based on decentralized and asynchronous Gossip protocols, for solving clustering, density estimation, dimension reduction, classification and general convex optimization problems, while offering an appreciable speed-up on large networks with a very low communication cost. These practical advantages are mathematically supported by a theoretical convergence analysis. We finally illustrate the relevance of proposed methods on multimedia indexing applications and real image classification tasks.