Reconstruction de réseaux de neurones parcimonieux par échantillonnage

Valérie Castin

Rapport de stage de M1, encadré par Rémi Gribonval.
ENS de Lyon, du 1er février au 1er juillet 2022.

J'ai fait mon stage du 1er février au 1er juillet 2022 à l'ENS de Lyon dans l'équipe DANTE, dont les thèmes de recherche sont l'apprentissage statistique, en particulier les questions de parcimonie dans l'apprentissage, et le traitement du signal. C'est une équipe nombreuse et très dynamique, avec une majorité de jeunes (doctorants et post-doctorants) et cinq membres permanents. Je me suis rapidement intégrée grâce aux nombreux moments d'échange organisés chaque semaine : séminaires hebdomadaires, réunions d'équipe mensuelles où chacun peut donner des nouvelles de ses travaux, groupes de travail sur différents sujets, mais aussi déjeuner collectif chaque jour au restaurant universitaire, souvent en compagnie d'autres équipes, et pause café ou thé après le repas. En somme, l'ambiance du labo était à la fois très agréable et très propice à la discussion et au travail de recherche. J'ai eu la chance d'arriver vers la fin des restrictions dues au covid, et donc de pouvoir être en présentiel tout au long de mon stage.

J'ai eu de très bons rapports avec mon encadrant, Rémi Gribonval, qui était toujours disponible pour répondre à mes questions malgré un emploi du temps chargé, et avec qui j'avais rendez-vous toutes les semaines environ. Il m'a permis de participer à différents ateliers et de présenter mes travaux tout au long du stage : j'ai pu assister à distance au workshop Mathematics of deep learning, donner un exposé devant toute l'équipe et un autre au séminaire des doctorants et post-doc, et présenter un poster résumant les principaux résultats de mon stage dans le cadre de l'atelier Sparsity in Neural Networks. J'aurai également l'occasion d'assister à une conférence sur le traitement du signal et des images (GRETSI) à Nancy en septembre.

Je remercie donc très chaleureusement Rémi Gribonval pour son accueil et sa disponibilité, aussi bien en phase de recherche que pendant l'écriture du rapport, ainsi que tous les autres membres de l'équipe DANTE, notamment Mathurin Massias, dont l'aide et les conseils m'ont été précieux. Ce stage à Lyon était mon premier contact avec la recherche, et j'en ai gardé une très bonne impression et l'envie de continuer dans cette voie.

Enfin, j'ai pu profiter pendant ces cinq mois de l'offre culturelle (et gastronomique) de Lyon, qui est une ville très agréable et vivante, et rejoindre le choeur de l'ENS Lyon, ce qui fut une très belle expérience. Nous avons en particulier donné en concert le Requiem de Verdi dans une église au cœur de Lyon, aux côtés de solistes et d'un orchestre professionnels.

Toutes les démonstrations sont regroupées en annexe, de façon à réduire autant que possible la longueur du corps du rapport.

1 Introduction

Les réseaux de neurones sont des algorithmes qui permettent de réaliser une grande variété de tâches, de la reconnaissance faciale à la traduction de texte, en passant par le jeu d'échecs et la génération d'images. Introduits au milieu du vingtième siècle selon un modèle très simplifié de l'organisation en couches des neurones humains, et initialement peu performants, ils ont connu un essor remarquable à partir de 2012 [3] avec le développement des réseaux de neurones profonds, c'est-à-dire comportant de nombreuses couches successives de neurones. Le succès de ce type de réseaux a été permis à la fois par une augmentation significative de la puissance de calcul, et par l'émergence des données de masse. Une très grande quantité de données est en effet nécessaire pour permettre à un réseau profond d'apprendre à effectuer correctement une tâche complexe.

D'un point de vue plus mathématique, tout réseau de neurones encode une fonction, appelée sa réalisation, qui dépend à la fois de l'architecture du réseau, c'est-à-dire de sa structure, de ses paramètres, et du choix d'une fonction réelle non linéaire, l'activation du réseau, qui apporte de la non-linéarité à la réalisation. Pour définir un réseau dévolu à une tâche donnée, il faut donc choisir son activation, ce qui est le plus souvent fait empiriquement, et son architecture. Les paramètres sont ensuite déterminés par une méthode de type descente de gradient, à partir d'un ensemble de données d'entraînement. On voit que le choix de l'architecture est central, et doit avant tout répondre à une exigence d'expressivité, c'est-à-dire que les fonctions représentables par les réseaux de l'architecture sélectionnée doivent être suffisamment complexes pour effectuer la tâche considérée. Cependant, depuis une dizaine d'années, un deuxième critère émerge pour développer de nouvelles architectures, dans une logique d'optimisation et d'économie des moyens de calcul : la parcimonie. Une architecture est dite parcimonieuse si les neurones de cette architecture ont peu de liens entre eux, comparativement au cas saturé où chaque neurone est relié à tous ceux de la couche précédente et de la couche suivante. Nous nous intéresserons dans ce rapport à des réseaux parcimonieux particuliers, à travers la question suivante : étant donné un réseau parcimonieux de paramètres inconnus, mais d'architecture connue, et dont on pourrait échantillonner à loisir la réalisation et sa jacobienne, est-il possible d'en reconstruire les paramètres, ou tout au moins de trouver des paramètres conduisant à la même réalisation ? Et si oui, avec quelle complexité d'échantillonnage, i.e. combien d'échantillons de la réalisation et de sa jacobienne ?

L'idée qui est à l'origine de ce sujet de stage est un résultat sur la factorisation matricielle à supports fixés, présenté dans [4]. Étant donnés une matrice $\mathbf{X}$ et $\mathbf{S}_1, \dots, \mathbf{S}_J$ des supports de matrices (vus comme des matrices binaires), il s'agit de trouver des matrices $\mathbf{X}_1, \dots, \mathbf{X}_J$ telles que pour tout $j \in [\![J]\!]$, le support de $\mathbf{X}_j$ est inclus dans $\mathbf{S}_j$, et :

$$\mathbf{X} \approx \mathbf{X}_J \dots \mathbf{X}_1, \tag{1}$$

où $\dots$ désigne le produit des matrices $\mathbf{X}_1$ à $\mathbf{X}_J$, et $\approx$ est une approximation pour la norme de Frobenius. L'article [4] propose un algorithme pour réaliser une telle factorisation sous certaines contraintes sur les supports des matrices $\mathbf{X}_j$, de sorte que ceux-ci deviennent parcimonieux (i.e. contiennent beaucoup de zéros), et en fournissant des garanties sur la factorisation obtenue quand la matrice $\mathbf{X}$ admet une factorisation exacte de la forme de (1). Ainsi, dans la perspective de reconstruire les paramètres d'un réseau inconnu, en supposant que les matrices de poids de celui-ci ont des supports permettant la factorisation de leur produit, ce résultat est exploitable à condition de pouvoir retrouver tout ou partie des coefficients du produit des matrices de poids du réseau. Lorsque l'activation choisie pour le réseau est la fonction `ReLU`: $x \mapsto \max(0, x)$, ce qui est très classique depuis [2], une façon naturelle d'obtenir des informations sur ce produit est d'échantillonner la jacobienne de la réalisation, comme on le verra dans la section suivante.

À la lumière de ces remarques, nous pouvons énoncer plus précisément le fil directeur du stage : étant donné un réseau ReLU dont les matrices de poids ont un support parcimonieux permettant

la factorisation de leur produit, d'architecture connue mais de paramètres inconnus, et dont on pourrait échantillonner la réalisation et sa jacobienne à loisir, est-il possible d'en reconstruire les paramètres, ou tout au moins de trouver des paramètres conduisant à la même réalisation ? Avec quelle complexité d'échantillonnage ? Ce problème, lié à des aspects de confidentialité des paramètres d'un réseau, soulève des questions qui, tout en faisant appel à des outils relativement élémentaires, n'en sont pas moins mathématiquement intéressantes.

Table des matières

2 Formalisation du problème, notations et premières définitions

Pour préciser le plan de réflexion énoncé dans la partie précédente, il nous faut avant tout introduire un certain nombre de définitions et notations, dont la plupart sont inspirées de [6]. Nous notons $\odot$ le produit de Hadamard entre matrices. Pour tout entier naturel N non nul, $[\![N]\!]$ désigne l'ensemble $\{1, \dots, N\}$, et $\mathfrak{S}_N$ l'ensemble des permutations de $[\![N]\!]$. Si $a, b \in \mathbb{Z}$, on note $[\![a, b]\!]$ l'ensemble $\{a, a+1, \dots, b\}$. Le support d'une matrice $\mathbf{M} = (m_{i,j})_{i,j} \in \mathbb{R}^{n \times m}$, noté $\mathrm{Supp}(\mathbf{M})$, est l'ensemble $\{(i,j) \in [\![n]\!] \times [\![m]\!] \mid m_{i,j} \neq 0\}$. On identifiera cet ensemble à la matrice binaire $(\mathbf{1}_{(m_{i,j} \neq 0)})_{i,j}$ où $\mathbf{1}$ est une indicatrice. Si $\mathbf{x} \in \mathbb{R}^n$ est un vecteur, $\mathrm{diag}(\mathbf{x})$ désigne la matrice diagonale construite à partir de $\mathbf{x}$. Pour tout entier $d \geq 1$, nous notons $\mathcal{S}^{d-1}$ la sphère unité de $\mathbb{R}^d$. Nous travaillerons exclusivement avec des réseaux de type perceptron multicouche, définis comme suit.

Définition 2.1 (perceptron multicouche, poids, biais, réalisation)**.** *Soient $\rho \colon \mathbb{R} \to \mathbb{R}$ une fonction continue dérivable presque partout, et $J \geq 2$. Un perceptron multicouche d'activation ρ et comportant $J+1$ couches se constitue de matrices de poids $\mathbf{W}_1, \dots, \mathbf{W}_J$, avec $\mathbf{W}_j \in \mathbb{R}^{N_j \times N_{j-1}}$ pour $j \in [\![J]\!]$, et de vecteurs de biais $\mathbf{b}_1, \dots, \mathbf{b}_J$ avec $\mathbf{b}_j \in \mathbb{R}^{N_j}$ pour $j \in [\![J]\!]$. Pour tout indice de couche $j \in [\![0, J]\!]$, chaque $i \in [\![N_j]\!]$ s'identifie à un neurone du réseau. On parle du i-ème neurone de la j-ème couche, en considérant que $[\![N_j]\!] \cap [\![N_{j'}]\!] = \emptyset$ quand $j \neq j'$. Si i appartient à la couche d'entrée $[\![N_0]\!]$, on dit que c'est un neurone d'entrée, si i appartient à la couche de sortie $[\![N_J]\!]$, c'est un neurone de sortie, et si i appartient à une couche cachée $[\![N_j]\!]$ avec $j \in [\![J-1]\!]$, le neurone i est dit caché. Les poids et biais du réseau en constituent les paramètres, que l'on regroupe dans un vecteur $\boldsymbol{\theta}$. Notons $\mathcal{R}$ l'objet réseau de neurones ainsi défini, qui s'identifie à $(\boldsymbol{\theta}, \rho)$. Un tel réseau encode une fonction, appelée réalisation, de la forme :*

$$R_{\boldsymbol{\theta}} \colon \mathbf{x} \in \mathbb{R}^{N_0} \mapsto \mathbf{W}_J \rho(\mathbf{W}_{J-1} \rho(\dots \rho(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) + \dots) + \mathbf{b}_{J-1})) + \mathbf{b}_J, \tag{2}$$

où ρ s'applique coordonnée par coordonnée par abus de notation. On note enfin $J_{\boldsymbol{\theta}}$ la jacobienne de $R_{\boldsymbol{\theta}}$, définie presque partout.

Dans la suite, tous les réseaux considérés sont des réseaux ReLU, c'est-à-dire dont l'activation est $\texttt{ReLU} \colon x \in \mathbb{R} \mapsto \max(0, x)$. L'expression (2) se réexprime en adoptant le formalisme par récurrence suivant.

Définition 2.2 (préactivation, neurone activé, neurone activable)**.** *Pour tout vecteur d'entrée $\mathbf{x}$, soit $y_0(\mathbf{x}) = \mathbf{x}$, puis $z_j(\mathbf{x}) = \mathbf{W}_j y_{j-1}(\mathbf{x}) + \mathbf{b}_j$ pour $j \in [\![J]\!]$ et $y_j(\mathbf{x}) = \texttt{ReLU}(z_j(\mathbf{x}))$ pour $j \in [\![J-1]\!]$. Alors $R_{\boldsymbol{\theta}}(\mathbf{x}) = z_J(\mathbf{x})$. Si ν est le i-ème neurone de la couche j, sa* préactivation *en $\mathbf{x}$ est définie par $(z_j(\mathbf{x}))_i$, et si ν est un neurone caché, on dit que $\mathbf{x}$ active ν si $(z_j(\mathbf{x}))_i > 0$. On considérera que les neurones d'entrée et de sortie sont toujours activés. Enfin, un neurone est dit activable s'il existe un vecteur d'entrée qui l'active.*

Tout réseau peut être vu comme un graphe (voir Figure 1), ce qui rend les raisonnements plus visuels et permet d'alléger la rédaction.

Définition 2.3 (graphe associé à un réseau)**.** *Soit $\mathcal{R}$ un réseau de neurones quelconque. On peut associer à $\mathcal{R}$ un graphe (V, E) défini de la façon suivante. L'ensemble V contient tous les neurones ν du réseau, et peut être partitionné en $(\mathcal{V}_j)_{0 \leq j \leq J}$, où $\mathcal{V}_0$ regroupe les neurones de la couche d'entrée, $\mathcal{V}_J$ ceux de la couche de sortie, et $\mathcal{V}_j$, pour $j \in [\![J-1]\!]$, les neurones de la j-ème couche cachée. Pour tout $j \in [\![0, J]\!]$, on note N_j le cardinal de l'ensemble $\mathcal{V}_j$, et l'on identifie ce dernier ensemble à $[\![N_j]\!]$, conformément à la Définition 2.1. L'ensemble E regroupe toutes les*

arêtes $e = (\mu \to \nu)$, orientées et pondérées par $w_e := (\mathbf{W}_j)_{\nu,\mu}$, où les neurones $\mu \in \mathcal{V}_{j-1}$ et $\nu \in \mathcal{V}_j$ sont situés dans deux couches consécutives. On note enfin $H = \cup_{j=1}^{J-1} \mathcal{V}_j$ l'ensemble des neurones cachés du réseau.

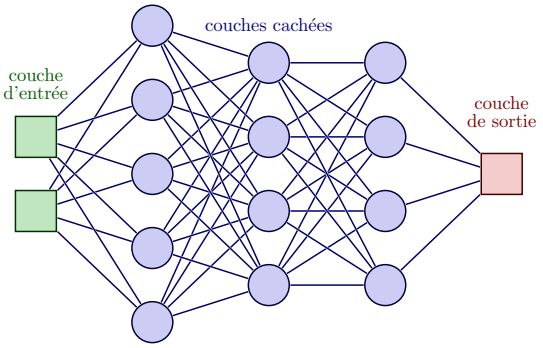


FIGURE 1 – Exemple de graphe d'un réseau à 3 couches cachées.

Le problème général dans lequel s'inscrit le sujet de ce stage est le suivant. On fixe $\mathcal{R}$ un réseau ReLU de paramètres inconnus, dont on note $R_{\boldsymbol{\theta}}$ la réalisation et $J_{\boldsymbol{\theta}}$ la jacobienne. On suppose que l'on a accès à $R_{\boldsymbol{\theta}}(\mathbf{x})$ et $J_{\boldsymbol{\theta}}(\mathbf{x})$ pour tous les $\mathbf{x}$ que l'on souhaite, et il s'agit de reconstruire autant que possible les paramètres de $\mathcal{R}$, en optimisant la complexité d'échantillonnage, c'est-à-dire le nombre de vecteurs d'entrée en lesquels on échantillonne $R_{\boldsymbol{\theta}}$ et $J_{\boldsymbol{\theta}}$.

Tout d'abord, que peut-on raisonnablement espérer quand on parle de reconstruction ? Pour répondre précisément à cette question, il faut étudier quelques propriétés de la fonction $\boldsymbol{\theta} \mapsto R_{\boldsymbol{\theta}}$ qui à un vecteur de paramètres associe la réalisation du réseau ReLU associé.

2.1 Invariances de la réalisation et reconstruction des paramètres

Une première remarque est qu'il est illusoire d'espérer retrouver exactement les paramètres de $\mathcal{R}$. En effet, il est connu [5, 6] que la fonction $\boldsymbol{\theta} \mapsto R_{\boldsymbol{\theta}}$ a d'importants défauts d'injectivité. Il y a notamment invariance de la réalisation par permutation des neurones au sein de chaque couche cachée.

Lemme 2.4 (invariance de la réalisation par permutation)**.** *Soit $\tau = (\tau_1, \dots, \tau_{J-1}) \in \mathfrak{S}_{N_1} \times \cdots \times \mathfrak{S}_{N_{J-1}}$. Un tel $J-1$-uplet de permutations agit naturellement sur un vecteur de paramètres $\boldsymbol{\theta}$ en permutant les lignes de $\mathbf{W}_j$, les colonnes de $\mathbf{W}_{j+1}$ et les coordonnées de $\mathbf{b}_j$ selon τ_j pour tout $j \in [\![J-1]\!]$. Alors, pour tout $\boldsymbol{\theta}$ et tout $\tau \in \mathfrak{S}_{N_1} \times \cdots \times \mathfrak{S}_{N_{J-1}}$, en notant $\boldsymbol{\theta}' = \tau \cdot \boldsymbol{\theta}$ le résultat de l'action de τ sur $\boldsymbol{\theta}$, on a : $R_{\boldsymbol{\theta}} = R_{\boldsymbol{\theta}'}$.*

Ce résultat est vrai pour tout réseau, quelle que soit son architecture, comme le montre bien le point de vue par graphe des réseaux de neurones. En revanche, le lemme suivant, qui énonce une deuxième propriété d'invariance de la réalisation, est spécifique aux réseaux de neurones ayant une activation positivement homogène, ce qui est le cas de `ReLU`.

Lemme 2.5 (invariance de la réalisation par changements d'échelle strictement positifs)**.** *Soient $\mathbf{W}_1, \dots, \mathbf{W}_J$ des matrices de poids et $\mathbf{b}_1, \dots, \mathbf{b}_J$ des vecteurs de biais. Notons $\boldsymbol{\theta}$ le vecteur de paramètres associé, et soient $\mathbf{D}_1, \dots, \mathbf{D}_{J-1}$ des matrices diagonales à coefficients diagonaux strictement positifs, avec $\mathbf{D}_j \in \mathbb{R}^{N_j \times N_j}$. Pour $j \in [\![J]\!]$, posons $\mathbf{W}'_j = \mathbf{D}_j \mathbf{W}_j \mathbf{D}_{j-1}^{-1}$, où $\mathbf{D}_0 = \mathbf{I}_{N_0}$ et $\mathbf{D}_J = \mathbf{I}_{N_J}$. Soit aussi $\mathbf{b}'_j = \mathbf{D}_j \mathbf{b}_j$. Alors le vecteur $\boldsymbol{\theta}'$ correspondant à ces derniers poids et biais vérifie : $R_{\boldsymbol{\theta}'} = R_{\boldsymbol{\theta}}$.*

Ainsi, si deux vecteurs de paramètres se déduisent l'un de l'autre par permutation et changements d'échelle strictement positifs, ils conduisent à la même réalisation. On peut noter que la réciproque n'est pas vraie si l'on n'impose aucune contrainte sur l'espace Θ dans lequel vivent les paramètres (voir [6] pour une analyse plus détaillée). Dès lors, sans plus d'hypothèses sur les réseaux étudiés, on ne peut espérer une reconstruction exacte des paramètres, et il est plus indiqué, étant donné un réseau ReLU d'architecture connue, et de paramètres $\boldsymbol{\theta}$ inconnus, de chercher à construire $\boldsymbol{\theta}'$ tel que $R_{\boldsymbol{\theta}} = R_{\boldsymbol{\theta}'}$. Toutefois, si l'on contraint suffisamment les supports des matrices de poids du réseau inconnu, ce que l'on va faire dans la suite, on pourra souvent retrouver les poids et biais modulo changements d'échelle strictement positifs (au sens du Lemme 2.5), sans permutation des neurones cachés. Ceci nous conduit à distinguer deux notions de reconstruction, l'une faible et l'autre forte, dont les définitions suivantes précisent le sens.

Définition 2.6 (paramètres fonctionnellement équivalents, reconstruction faible)**.** *Deux vecteurs de paramètres $\boldsymbol{\theta}$ et $\boldsymbol{\theta}'$ sont dits fonctionnellement équivalents si $R_{\boldsymbol{\theta}} = R_{\boldsymbol{\theta}'}$. On dira aussi que les réseaux associés sont fonctionnellement équivalents. Reconstruire faiblement un réseau de paramètres inconnus $\boldsymbol{\theta}$, c'est trouver $\boldsymbol{\theta}'$ fonctionnellement équivalent à $\boldsymbol{\theta}$.*

Définition 2.7 (paramètres équivalents modulo changements d'échelle strictement positifs, reconstruction forte)**.** *Des poids et biais $\mathbf{W}'_1, \dots, \mathbf{W}'_J$ et $\mathbf{b}'_1, \dots, \mathbf{b}'_J$ sont dits équivalents à d'autres poids et biais $\mathbf{W}_1, \dots, \mathbf{W}_J$ et $\mathbf{b}_1, \dots, \mathbf{b}_J$ modulo changements d'échelle strictement positifs s'il existe des matrices diagonales à coefficients diagonaux strictement positifs $\mathbf{D}_1, \dots, \mathbf{D}_{J-1}$ telles que $\mathbf{W}'_j = \mathbf{D}_j \mathbf{W}_j \mathbf{D}_{j-1}^{-1}$ pour tout $j \in [\![J]\!]$, où $\mathbf{D}_0 = \mathbf{I}_{N_0}$ et $\mathbf{D}_J = \mathbf{I}_{N_J}$, et $\mathbf{b}'_j = \mathbf{D}_j \mathbf{b}_j$. Reconstruire fortement un réseau de paramètres inconnus $\boldsymbol{\theta}$, c'est trouver $\boldsymbol{\theta}'$ équivalent à $\boldsymbol{\theta}$ modulo changements d'échelle strictement positifs.*

Intéressons-nous à présent à la jacobienne des réseaux ReLU (c'est-à-dire celle de leur réalisation par abus de langage). La sous-section suivante montre qu'échantillonner la jacobienne donne des informations directes sur le produit des matrices de poids de $\mathcal{R}$.

2.2 Forme de la jacobienne pour des réseaux ReLU

Pour étudier la forme que prend la jacobienne d'un perceptron multicouche, on dispose du lemme suivant, qui s'obtient en faisant un développement au premier ordre.

Lemme 2.8 ([6, Lemme 9])**.** *Soient $J \geq 2$ et $\mathcal{R}$ un réseau à $J+1$ couches, d'activation ρ dérivable presque partout. Notons $\mathbf{W}_1, \dots, \mathbf{W}_J$ ses matrices de poids, et $\mathbf{b}_1, \dots, \mathbf{b}_J$ ses biais. Nous reprenons le formalisme par récurrence introduit dans la Définition 2.2. Alors la jacobienne de $\mathcal{R}$ est définie presque partout, et s'exprime de la façon suivante, pour tout vecteur d'entrée $\mathbf{x}$ en lequel l'expression a un sens :*

$$J_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{W}_J \operatorname{diag}(\rho'(z_{J-1}(\mathbf{x})))\mathbf{W}_{J-1} \dots \operatorname{diag}(\rho'(z_1(\mathbf{x})))\mathbf{W}_1. \tag{3}$$

Appliqué aux réseaux ReLU, ce lemme conduit immédiatement au corollaire suivant.

Corollaire 2.9. *Soit $\mathcal{R}$ un réseau ReLU, dont on note $\mathbf{W}_1, \dots, \mathbf{W}_J$ les poids, et $\mathbf{b}_1, \dots, \mathbf{b}_J$ les biais. On reprend les notations du Lemme 2.8 avec $\rho = \texttt{ReLU}$. Pour $j \in [\![J-1]\!]$, soit $\mathbf{D}_j$ la matrice diagonale dont le i-ème coefficient diagonal vaut 1 si le i-ème neurone de la j-ème couche cachée est activé, i.e. si $(z_j(\mathbf{x}))_i > 0$, et 0 sinon. Alors pour tout $\mathbf{x}$ où $R_{\boldsymbol{\theta}}$ est différentiable on a :*

$$J_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{W}_J \mathbf{D}_{J-1} \mathbf{W}_{J-1} \dots \mathbf{D}_1 \mathbf{W}_1. \tag{4}$$

L'observation de $J_{\boldsymbol{\theta}}$ en un point d'entrée donne ainsi des informations partielles sur le produit $\mathbf{W}_J \dots \mathbf{W}_1$. Or, sous certaines conditions sur les supports des $\mathbf{W}_j$, on dispose d'un algorithme de factorisation qui, appliqué à $\mathbf{W}_J \dots \mathbf{W}_1$, ou à une version approchée de ce produit, fournit des matrices de poids $\mathbf{W}''_1, \dots, \mathbf{W}''_J$ compatibles (ou presque) avec le réseau initial. Il s'agit donc, à l'aide d'un échantillonnage judicieux de la jacobienne, de réunir les informations obtenues sur $\mathbf{W}_J \dots \mathbf{W}_1$, puis de factoriser pour reconstruire les poids. La sous-section suivante présente les résultats qui nous seront utiles pour cette étape de factorisation, et qui sont présentés dans [7] et [8].

2.3 Algorithme de factorisation matricielle à supports fixés

Tous les résultats de cette partie sont énoncés dans $\mathbb{R}$, mais s'appliquent plus généralement à des matrices à coefficients complexes. Penchons-nous en premier lieu sur la factorisation de matrices en produit de deux facteurs.

Définition 2.10 (famille des composantes de rang 1)**.** *On définit* $\varphi\colon \mathbb{R}^{N\times d} \times \mathbb{R}^{s\times N} \to (\mathbb{R}^{s\times d})^N$ *l'opérateur associant à* $(\mathbf{M}_1, \mathbf{M}_2)$ *la famille suivante :*

$$\varphi\colon (\mathbf{M}_1, \mathbf{M}_2) \mapsto (C_i(\mathbf{M}_2) L_i(\mathbf{M}_1)^\top)_{1\leq i\leq N}, \tag{5}$$

où L_i *et* C_i *extraient respectivement la* i*-ème ligne et la* i*-ème colonne de la matrice passée en argument, sous forme de vecteurs colonne. Cette famille est appelée famille des composantes de rang 1 du produit* $\mathbf{M}_2\mathbf{M}_1$*.*

Remarque 2.11. *Rappelons que* $\varphi(\mathbf{M}_1, \mathbf{M}_2)$ *est relié au produit* $\mathbf{M}_2\mathbf{M}_1$ *par la formule suivante :*

$$\mathbf{M}_2\mathbf{M}_1 = \sum_{i=1}^{N} C_i(\mathbf{M}_2) L_i(\mathbf{M}_1)^\top. \tag{6}$$

Définition 2.12 (couple admissible de supports)**.** *Soient* $\mathbf{S}_1 \in \{0,1\}^{N\times d}$ *et* $\mathbf{S}_2 \in \{0,1\}^{s\times N}$ *deux supports de matrices. On dira que* $(\mathbf{S}_1, \mathbf{S}_2)$ *est* admissible *si les éléments de* $\varphi(\mathbf{S}_1, \mathbf{S}_2)$ *ont des supports non vides et deux à deux disjoints.*

On a alors le résultat suivant.

Proposition 2.13 ([4],[8])**.** *Soient* $(\mathbf{S}_1 \in \{0,1\}^{N\times d}, \mathbf{S}_2 \in \{0,1\}^{s\times N})$ *un couple admissible de supports, et* $\mathbf{M}_1, \mathbf{M}_2$ *deux matrices de supports inclus respectivement dans* $\mathbf{S}_1$ *et* $\mathbf{S}_2$*. Alors connaissant* $\mathbf{X} = \mathbf{M}_2\mathbf{M}_1$*, on dispose d'un algorithme de factorisation qui renvoie deux matrices* $\mathbf{M}'_1, \mathbf{M}'_2$ *de supports inclus dans* $\mathbf{S}_1$ *et* $\mathbf{S}_2$ *respectivement, et telles que* $\mathbf{M}'_2\mathbf{M}'_1 = \mathbf{X}$*.*

En outre, si $\mathrm{Supp}(\mathbf{M}_1) = \mathbf{S}_1$ *et* $\mathrm{Supp}(\mathbf{M}_2) = \mathbf{S}_2$*, le couple de matrices* $(\mathbf{M}'_1, \mathbf{M}'_2)$ *renvoyé par l'algorithme est équivalent à* $(\mathbf{M}_1, \mathbf{M}_2)$ *modulo changements d'échelle inversibles, au sens où il existe* $\mathbf{D} \in \mathbb{R}^{N\times N}$ *une matrice diagonale inversible vérifiant* $\mathbf{M}'_1 = \mathbf{D}\mathbf{M}_1$ *et* $\mathbf{M}'_2 = \mathbf{M}_2\mathbf{D}^{-1}$*.*

Sans rentrer dans les détails de la démonstration de cette proposition, présentons tout de même le principe de l'algorithme de factorisation à deux facteurs de Le, Zheng, Riccietti et Gribonval [4], qu'il est utile de connaître pour la suite. Notons $(\mathbf{\Sigma}_i)_{1\leq i\leq N} = \varphi(\mathbf{S}_1, \mathbf{S}_2)$. Par hypothèse d'admissibilité des supports, les $\mathbf{\Sigma}_i$ sont non vides et deux à deux disjoints. Pour chaque $i \in [\![N]\!]$, soit I l'ensemble des indices correspondant à une ligne non nulle de $\mathbf{\Sigma}_i$, et J celui des indices correspondant à une colonne non nulle de $\mathbf{\Sigma}_i$. Notons $\mathbf{X}_{I,J}$ la sous-matrice de $\mathbf{X}$ indexée par I sur les lignes et J sur les colonnes. C'est une matrice de rang 1, car extraite de la matrice $\mathbf{X} \odot \mathbf{\Sigma}_i = C_i(\mathbf{M}_2) L_i(\mathbf{M}_1)^\top$, cette dernière égalité provenant de la Remarque 2.11, dans

le cas où les $\mathbf{\Sigma}_i$ sont supposés deux à deux disjoints. En effectuant une décomposition en valeurs singulières, on trouve $\mathbf{U} \in \mathbb{R}^{|I|}$ et $\mathbf{V} \in \mathbb{R}^{|J|}$ tels que $\mathbf{X}_{I,J} = \mathbf{U}\mathbf{V}^\top$. On définit alors $(\mathbf{M}_2)_{I,i} = \mathbf{U}$ et $(\mathbf{M}_1)_{i,J} = \mathbf{V}$, ce qui donne à la fin du processus des matrices $\mathbf{M}_1$ et $\mathbf{M}_2$ convenables. Notons que l'on peut appliquer l'algorithme à n'importe quelle matrice $\mathbf{X}$, même si celle-ci ne s'écrit pas comme un produit $\mathbf{M}_2\mathbf{M}_1$ avec $\mathrm{Supp}(\mathbf{M}_1) \subset \mathbf{S}_1$ et $\mathrm{Supp}(\mathbf{M}_2) \subset \mathbf{S}_2$. Dans ce cas, on approche $\mathbf{X}_{I,J}$ par une matrice de rang 1, et l'on obtient $\mathbf{M}'_1, \mathbf{M}'_2$ de supports inclus respectivement dans $\mathbf{S}_1$ et $\mathbf{S}_2$, et telles que $\mathbf{M}'_2\mathbf{M}'_1$ est une approximation de $\mathbf{X}$ de la forme souhaitée optimale pour la norme de Frobenius.

Remarque 2.14. *Extraire $\mathbf{X}_{I,J}$ avant de réaliser la décomposition en valeurs singulières plutôt que d'effectuer celle-ci sur $\mathbf{X} \odot \mathbf{\Sigma}_i$ permet de réduire le temps de calcul, puisque les SVD portent alors sur des matrices de taille plus petite, pour le même résultat.*

L'application récursive d'un tel algorithme, présentée dans [4], permet une factorisation matricielle à plusieurs facteurs pour un choix adéquat de supports. Soient $\mathbf{S}_1, \dots, \mathbf{S}_J$ des supports de matrices. Étant donné $\mathbf{X}$ s'écrivant exactement comme un produit $\mathbf{M}_J \dots \mathbf{M}_1$ avec $\mathrm{Supp}(\mathbf{M}_j) \subset \mathbf{S}_j$ pour tout j, on veut trouver des matrices $\mathbf{M}'_1, \dots, \mathbf{M}'_J$ telles que $\mathbf{X} = \mathbf{M}'_J \dots \mathbf{M}'_1$ et $\mathrm{Supp}(\mathbf{M}'_j) \subset \mathbf{S}_j$ pour tout j. On raisonne de la façon suivante. Chaque itération de l'algorithme est associée à deux entiers $p < q$ et à un bloc à factoriser $\mathbf{H}$. L'initialisation se fait avec $p = 1$, $q = J$ et $\mathbf{H} = \mathbf{X}$. Tant que $q - p > 1$, on choisit $\ell \in [\![p, q-1]\!]$, et en supposant que les supports $\mathbf{S}^G := (\mathbf{S}_q \dots \mathbf{S}_{\ell+1})$ et $\mathbf{S}^D := (\mathbf{S}_\ell \dots \mathbf{S}_p)$ forment un couple admissible, on factorise $\mathbf{H}$ avec l'algorithme à deux facteurs selon ces supports. On applique ensuite récursivement l'algorithme aux deux blocs obtenus, associés aux entiers p, ℓ et $\ell + 1, q$ respectivement.

On voit que cette méthode s'applique à condition qu'il existe un choix d'ordre pour la factorisation – correspondant à un arbre binaire (voir Figure 2) – suivant lequel à chaque étape, $(\mathbf{S}^G, \mathbf{S}^D)$ est admissible, sinon l'on ne peut pas factoriser avec l'algorithme précédent.

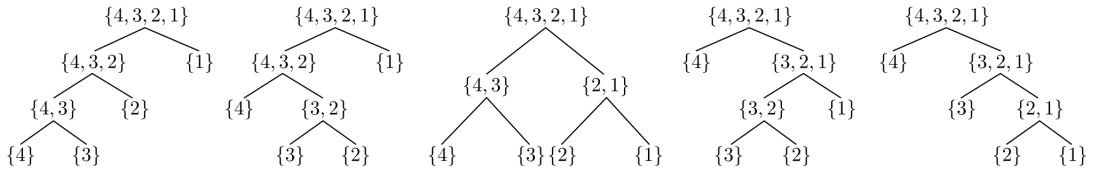


FIGURE 2 – Les cinq arbres de factorisation possibles dans le cas à 4 facteurs, figure tirée de [7]. Les entiers p et q choisis à chaque étape sont respectivement le maximum de l'ensemble fils droit et le minimum de l'ensemble fils gauche associés.

Ceci nous amène à introduire la notion de supports factorisants, en vue d'énoncer un résultat de factorisation multicouche récursive.

Définition 2.15 (supports factorisants)**.** *Soit $J \geq 2$. Une famille de supports $(\mathbf{S}_1, \dots, \mathbf{S}_J)$ est dite* factorisante *s'il existe un arbre binaire déterminant l'ordre des factorisations successives suivant lequel, à chaque étape, $(\mathbf{S}^G, \mathbf{S}^D)$ est admissible.*

Proposition 2.16 ([7, Thm 3.10])**.** *Soit $(\mathbf{S}_j \in \{0,1\}^{N_j \times N_{j-1}})_{1 \leq j \leq J}$ une famille de supports factorisante. Pour tout $j \in [\![J]\!]$, considérons $\mathbf{M}_j$ une matrice dont le support est inclus dans $\mathbf{S}_j$, et notons $\mathbf{X} = \mathbf{M}_J \dots \mathbf{M}_1$. Alors on dispose d'un algorithme de factorisation qui renvoie J matrices $\mathbf{M}'_1, \dots, \mathbf{M}'_J$ de supports inclus dans les $\mathbf{S}_j$, et vérifiant $\mathbf{M}'_J \dots \mathbf{M}'_1 = \mathbf{X}$.*

Si l'on suppose de plus que les supports des $\mathbf{M}_j$ sont égaux aux $\mathbf{S}_j$, alors le J-uplet de matrices $(\mathbf{M}'_1, \dots, \mathbf{M}'_J)$ obtenu est équivalent à $(\mathbf{M}_1, \dots, \mathbf{M}_J)$ modulo changements d'échelle inversibles,

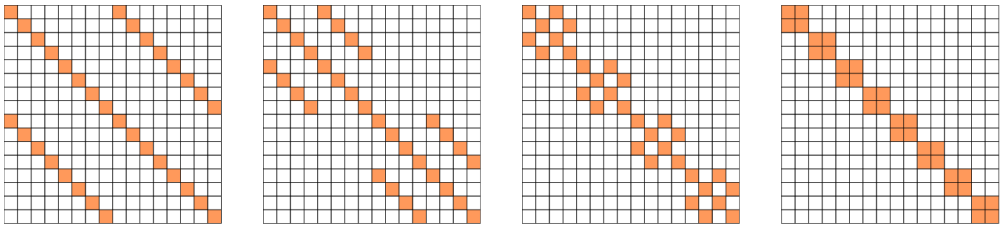
au sens où il existe $\mathbf{D}_1, \dots, \mathbf{D}_{J-1}$ *des matrices diagonales inversibles telles que* $\mathbf{M}'_j = \mathbf{D}_j \mathbf{M}_j \mathbf{D}_{j-1}^{-1}$ *pour tout* $j \in [\![J]\!]$*, avec* $\mathbf{D}_0 = \mathbf{I}_{N_0}$ *et* $\mathbf{D}_J = \mathbf{I}_{N_J}$*.*

En pratique, peut-on exhiber une famille de supports factorisants pour tout J ? L'exemple que nous allons utiliser pour la reconstruction de réseaux multicouche est celui des supports butterfly, dont l'idée apparaît (avant la formulation matricielle) dans un article de 1965 par Cooley et Tukey [1].

Définition 2.17 (matrice butterfly)**.** *Soient* $J \geq 2$ *et* $N = 2^J$*. Pour* $j \in [\![J]\!]$*, on définit le* j*-ème support butterfly en taille* N *par :*

$$\mathbf{S}_j = \mathbf{I}_{N/2^j} \otimes \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \otimes \mathbf{I}_{2^{j-1}} \in \{0,1\}^{N \times N}, \tag{7}$$

où $\mathbf{I}_\ell$ *désigne la matrice identité de taille* ℓ*, et* $\otimes$ *est le produit de Kronecker entre matrices. On dit qu'une matrice* $\mathbf{X} \in \mathbb{R}^{N \times N}$ *admet une structure butterfly si elle peut être factorisée exactement sous la forme* $\mathbf{X}_J \dots \mathbf{X}_1$*, avec* $\mathrm{Supp}(\mathbf{X}_j) \subset \mathbf{S}_j$ *pour tout* $j \in [\![J]\!]$*.*



(a) $\mathbf{S}_4$ (b) $\mathbf{S}_3$ (c) $\mathbf{S}_2$ (d) $\mathbf{S}_1$

FIGURE 3 – Supports butterfly dans le cas à 4 facteurs, figure tirée de [7].

La structure butterfly apparaît dans la version rapide de nombreuses transformées discrètes, comme celles de Fourier ou de Hadamard. Nous développons ce dernier cas dans un exemple.

Exemple 2.18 (factorisation butterfly de la transformée de Hadamard, [4])**.** *Pour* $J \in \mathbb{N}$*, notons* $\mathbf{H}_N$ *la matrice associée à la transformée de Hadamard en taille* $N = 2^J$*, définie par récurrence par* $\mathbf{H}_1 = 1$ *et* $\mathbf{H}_{2N} = \left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right) \otimes \mathbf{H}_N$*. Alors* $\mathbf{H}_N$ *admet la décomposition butterfly suivante :* $\mathbf{H}_N = \mathbf{F}_J \dots \mathbf{F}_1$ *avec* $\mathbf{F}_j = \mathbf{I}_{N/2^j} \otimes \left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right) \otimes \mathbf{I}_{2^{j-1}}$ *pour tout* $j \in [\![J]\!]$*. La multiplication par* $\mathbf{H}_N$ *peut être réalisée de façon efficace par multiplications successives avec les facteurs butterfly, qui sont parcimonieux, ce qui n'est autre qu'une reformulation de la transformée de Hadamard rapide.*

Les supports butterfly sont particulièrement adaptés à l'application récursive de l'algorithme de factorisation à deux couches.

Lemme 2.19 ([4, Lemme 2])**.** *Soient* $J \geq 2$ *et* $N = 2^J$*. Notons* $\mathbf{S}_1, \dots, \mathbf{S}_J$ *les supports butterfly de taille* N*. Alors pour tous* $p \leq \ell < q$ *dans* $[\![J]\!]$*, les supports* $\mathbf{S}^G := (\mathbf{S}^q \dots \mathbf{S}^{\ell+1})$ *et* $\mathbf{S}^D := (\mathbf{S}^\ell \dots \mathbf{S}^p)$ *forment un couple admissible.*

Ainsi, si $\mathbf{X}$ admet une structure butterfly et que ses facteurs $\mathbf{X}_1, \dots, \mathbf{X}_J$ vérifient $\mathrm{Supp}(\mathbf{X}_j) = \mathbf{S}_j$, la Proposition 2.16 s'applique, et l'on peut factoriser $\mathbf{X}$ en $\mathbf{Y}_J \dots \mathbf{Y}_1$ où les $\mathbf{Y}_j$ sont équivalents aux $\mathbf{X}_j$ modulo changements d'échelle inversibles.

Nous n'approfondirons pas outre mesure ces résultats de factorisation, qui sont plutôt des outils pour résoudre notre problème que l'objet même du stage.

2.4 Délimitation du sujet

Deux points de vue émergent alors pour répondre à la question posée en introduction. Le premier est algorithmique : peut-on concevoir un algorithme qui échantillonne la jacobienne et la réalisation en un nombre fini de vecteurs d'entrée et reconstruit – fortement ou faiblement – le réseau initial inconnu, en gardant une complexité d'échantillonnage raisonnable ? Nous donnons au terme *complexité d'échantillonnage* le sens suivant.

Définition 2.20 (complexité d'échantillonnage)**.** *La complexité d'échantillonnage d'un algorithme de reconstruction des paramètres du réseau inconnu est le nombre de vecteurs d'entrée* $\mathbf{x}_k$ *en lesquels l'algorithme évalue* $R_{\boldsymbol{\theta}}(\mathbf{x}_k)$ *ou* $J_{\boldsymbol{\theta}}(\mathbf{x}_k)$*.*

Le deuxième point de vue, plus théorique, consiste à caractériser les familles d'entrée qui permettent la reconstruction, puis à borner leur cardinal à architecture fixée, ce qui fournit une échelle de mesure de la complexité de différentes architectures de réseaux de neurones. Les définitions suivantes permettent de préciser un peu la terminologie.

Définition 2.21 (familles reconstruisantes et fortement reconstruisantes, minimalité)**.** *Étant donné un réseau inconnu* $\mathcal{R}$ *dont les matrices de poids ont des supports inclus dans des supports factorisants, une famille d'entrée* $(\mathbf{x}_k)_{1\leq k\leq K}$ *est dite reconstruisante (resp. fortement reconstruisante) si la connaissance de* $(J_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ *et* $(R_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ *permet la reconstruction faible (resp. forte) des paramètres de* $\mathcal{R}$*. On dira de plus qu'une telle famille est minimale s'il n'existe pas de famille (fortement) reconstruisante pour* $\mathcal{R}$ *de cardinal strictement inférieur à* K*.*

Définition 2.22 (cardinal reconstruisant, fortement reconstruisant)**.** *Le cardinal reconstruisant (resp. fortement reconstruisant) d'un réseau est le cardinal de ses familles reconstruisantes (resp. fortement reconstruisantes) minimales.*

Au cours de notre analyse, nous serons amenés à distinguer le cas dit sans biais du cas avec biais. Se placer dans le cas sans biais signifie considérer uniquement des réseaux inconnus sans biais : l'on sait que le réseau à reconstruire est sans biais. Dans le cas avec biais, on n'a en revanche aucune information sur la nullité ou non des biais du réseau inconnu.

Nous allons commencer par étudier des réseaux à une seule couche cachée, situation plus simple que le cas multicouche mais où interviennent déjà plusieurs idées centrales dans le sujet. Nous nous pencherons ensuite sur le cas de réseaux butterfly à nombre de couches arbitraire.

3 Reconstruction de réseaux à une couche cachée

Dans cette section, $(\mathbf{S}_1, \mathbf{S}_2)$ est un couple admissible de supports au sens de la Définition 2.12, supposé connu, et $\mathcal{R}$ est un réseau ReLU à une couche cachée, dont on note $\mathbf{W}_1 \in \mathbb{R}^{N\times d}$ et $\mathbf{W}_2 \in \mathbb{R}^{s\times N}$ les matrices de poids et $(\mathbf{b}, \mathbf{c})$ les vecteurs de biais. Ainsi, d est la dimension d'entrée, et s celle de sortie. Nous nous plaçons dans un premier temps sous l'hypothèse suivante (H_1).

Hypothèse 3.1 (Hypothèse (H_1))**.** *Soit* $(\mathbf{S}_1, \mathbf{S}_2)$ *un couple admissible de supports, connu. Notons* $\mathbf{W}_1 \in \mathbb{R}^{N\times d}$ *et* $\mathbf{W}_2 \in \mathbb{R}^{s\times N}$ *les matrices de poids et* $(\mathbf{b}, \mathbf{c}) \in \mathbb{R}^N \times \mathbb{R}^s$ *les vecteurs de biais du réseau à reconstruire, dont on connaît l'architecture. On suppose que l'on sait que les supports de* $\mathbf{W}_1$ *et* $\mathbf{W}_2$ *sont respectivement égaux à* $\mathbf{S}_1$ *et* $\mathbf{S}_2$*.*

Nous traiterons ensuite le cas où l'on ne suppose plus l'égalité mais seulement l'inclusion entre supports, et qui fait intervenir des situations dites dégénérées. Nous notons $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_N$ les éléments de $\varphi(\mathbf{S}_1, \mathbf{S}_2)$. Rappelons l'expression de la réalisation et de la jacobienne :

$$R_{\boldsymbol{\theta}} \colon \mathbf{x} \in \mathbb{R}^d \mapsto \mathbf{W}_2\, \mathtt{ReLU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}) + \mathbf{c}, \tag{8}$$

et

$$J_{\boldsymbol{\theta}} \colon \mathbf{x} \in \mathbb{R}^d \mapsto \mathbf{W}_2 \mathbf{D} \mathbf{W}_1, \text{ avec } \mathbf{D} = \operatorname{diag}\left(\mathbf{1}_{\langle L_i(\mathbf{W}_1), \mathbf{x} \rangle + b_i > 0}\right)_{1 \leq i \leq N}. \tag{9}$$

3.1 Caractérisation des familles fortement reconstruisantes et bornes sur leur cardinal

Pour établir ce que doit vérifier une famille fortement reconstruisante, commençons par chercher des conditions sur les familles d'entrée assurant la reconstruction des poids, puis celle des biais connaissant les poids. La reconstruction des matrices de poids se fait essentiellement par factorisation de leur produit, produit que l'on peut retrouver en observant la jacobienne en certains points bien choisis, comme l'indique le lemme suivant, qui est une réécriture des équations (8) et (9).

Lemme 3.2. *Soit $\mathbf{x} \in \mathbb{R}^d$ un vecteur d'entrée. On a :*

$$R_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{c} + \sum_{i=1}^{N} \mathbf{1}_{\langle L_i(\mathbf{W}_1), \mathbf{x} \rangle + b_i > 0} \; C_i(\mathbf{W}_2)(L_i(\mathbf{W}_1)^\top \mathbf{x} + b_i) \tag{10}$$

et

$$J_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{i=1}^{N} \mathbf{1}_{\langle L_i(\mathbf{W}_1), \mathbf{x} \rangle + b_i > 0} \; C_i(\mathbf{W}_2) L_i(\mathbf{W}_1)^\top. \tag{11}$$

Remarquons que si un vecteur d'entrée $\mathbf{x}$ active le neurone caché i (au sens de la Définition 2.2), alors $C_i(\mathbf{W}_2)L_i(\mathbf{W}_1)^\top = J_{\boldsymbol{\theta}}(\mathbf{x}) \odot \boldsymbol{\Sigma}_i$, puisque par ($\mathrm{H}_1$) les éléments de la somme (11) ont des supports deux à deux disjoints inclus dans les $\boldsymbol{\Sigma}_i$. Il s'ensuit que si une famille d'entrée $(\mathbf{x}_k)_{1 \leq k \leq K}$ est telle que pour tout $i \in [\![N]\!]$, il existe un indice $k \in [\![K]\!]$ tel que $\mathbf{x}_k$ active le neurone caché i (quels que soient les autres neurones éventuellement activés par $\mathbf{x}_k$), on peut aisément retrouver le produit $\mathbf{W}_2\mathbf{W}_1$ en exploitant la formule $\mathbf{W}_2\mathbf{W}_1 = \sum_{i=1}^{N} C_i(\mathbf{W}_2)L_i(\mathbf{W}_1)^\top$, puisque $C_i(\mathbf{W}_2)L_i(\mathbf{W}_1)^\top = J_{\boldsymbol{\theta}}(\mathbf{x}_k) \odot \boldsymbol{\Sigma}_i$ pour $\mathbf{x}_k$ activant i, puis factoriser ce produit en un couple de facteurs $(\mathbf{W}_1'', \mathbf{W}_2'')$. Pour arriver à une reconstruction forte, il reste à ajuster les signes de certaines lignes de $\mathbf{W}_1''$ et colonnes de $\mathbf{W}_2''$, mais on verra que cette étape, dite de redressement, ne nécessite pas d'échantillonnage supplémentaire sous de bonnes conditions sur la famille d'entrée. Ceci nous conduit à énoncer un premier résultat.

Proposition 3.3 (condition pour la reconstruction forte des matrices de poids)**.** *On suppose que l'hypothèse (H_1) est satisfaite. Soit $(\mathbf{x}_k)_{1 \leq k \leq K}$ une famille d'entrée. Alors la donnée de $(J_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1 \leq k \leq K}$ et $(R_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1 \leq k \leq K}$ permet de construire des poids équivalents à ceux de $\mathcal{R}$ modulo changements d'échelle strictement positifs (sans pour l'instant s'occuper des biais) si, et seulement si, les deux conditions suivantes sont simultanément satisfaites :*

(i) pour tout $i \in [\![N]\!]$, il existe $k \in [\![K]\!]$ tel que $\mathbf{x}_k$ active le neurone caché i,

(ii) pour tout $i \in [\![N]\!]$, il existe $k \in [\![K]\!]$ tel que $|\langle L_i(\mathbf{W}_1), \mathbf{x}_k \rangle| \geq |b_i|$.

Voir démonstration en annexe.

Remarque 3.4. *En pratique, si les conditions (i) et (ii) de la Proposition 3.3 sont satisfaites, connaître simplement $(J_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1 \leq k \leq K}$ est suffisant pour reconstruire les poids de $\mathcal{R}$ modulo changements d'échelle strictement positifs, sous l'hypothèse (H_1).*

On peut ensuite chercher à caractériser les familles qui permettent, connaissant les poids modulo changements d'échelle strictement positifs, de reconstruire les biais associés. La première méthode que nous avons étudiée repose sur des considérations géométriques s'inspirant de

[5], et fournit une condition suffisante pour la reconstruction. Pour l'énoncer, étudions de plus près la représentation graphique de la réalisation des réseaux ReLU, et introduisons un peu de terminologie.

Définition 3.5 (neurone dégénéré)**.** *Un neurone caché $i \in [\![N]\!]$ est dit dégénéré si au moins l'un des deux vecteurs $L_i(\mathbf{W}_1)$ et $C_i(\mathbf{W}_2)$ est nul. Nous notons $\mathcal{D}$ l'ensemble des neurones cachés dégénérés, et $\overline{\mathcal{D}}$ son complémentaire dans $[\![N]\!]$, qui regroupe les neurones cachés non dégénérés.*

Remarquons que l'hypothèse (H_1) est incompatible avec l'existence de neurones dégénérés, puisque des supports admissibles $\mathbf{S}_1, \mathbf{S}_2$ ne peuvent avoir respectivement une ligne ou une colonne nulle.

Définition 3.6 (frontière associée à un neurone caché non dégénéré, zones affines)**.** *Si $i \in [\![N]\!]$ est un neurone caché non dégénéré, et z sa préactivation, on définit $\mathcal{H}_i = \{\mathbf{x} \in \mathbb{R}^d \mid z(\mathbf{x}) = 0\}$ la frontière associée à i. Les composantes connexes de $\mathbb{R}^d \setminus \mathcal{H}$, où $\mathcal{H} := \cup_{i \in \overline{\mathcal{D}}} \mathcal{H}_i$, sont appelées* zones affines*. La réalisation coïncide avec une application affine sur chaque zone affine et, sous (H_1), n'est différentiable en aucun point de $\mathcal{H}$. Pour tout neurone $i \in \overline{\mathcal{D}}$, on voit que $\mathcal{H}_i$ est un hyperplan affine de $\mathbb{R}^d$, d'équation $\mathcal{H}_i = \left\{\mathbf{z} \in \mathbb{R}^d \ ; \ \langle L_i(\mathbf{W}_1), \mathbf{z}\rangle = -b_i\right\}$.*

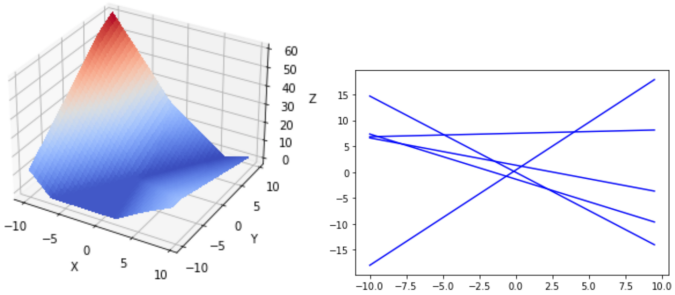


FIGURE 4 – À gauche : représentation graphique de la réalisation d'un réseau ReLU à une couche cachée, celle-ci comportant cinq neurones, tous non dégénérés. À droite : frontières des zones affines associées.

Nous définissons également la notion de i-voisinage, pour $i \in [\![N]\!]$.

Définition 3.7 (points i-voisins)**.** *Soit $i \in \overline{\mathcal{D}}$ un neurone caché non dégénéré. Deux vecteurs d'entrée $\mathbf{x}$ et $\mathbf{y}$ sont dits i-voisins si $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d \setminus \mathcal{H}$, et seul l'état d'activation du i-ème neurone change de $\mathbf{x}$ à $\mathbf{y}$.*

Nous avons alors le résultat suivant, qui découle immédiatement de l'équation (9).

Lemme 3.8. *Si $\mathbf{x}$ et $\mathbf{y}$ sont i-voisins pour $i \in \overline{\mathcal{D}}$, alors il existe $\varepsilon \in \{-1, 1\}$ tel que*

$$J_{\boldsymbol{\theta}}(\mathbf{y}) - J_{\boldsymbol{\theta}}(\mathbf{x}) = \varepsilon \, C_i(\mathbf{W}_2) L_i(\mathbf{W}_1)^\top. \tag{12}$$

Remarque 3.9. *La réciproque n'est pas vraie en général. On peut par exemple considérer un réseau sans biais à trois neurones cachés, de poids $\tilde{\mathbf{W}}_1$ et $\tilde{\mathbf{W}}_2$ tels que $L_1(\tilde{\mathbf{W}}_1) = L_2(\tilde{\mathbf{W}}_1) = L_3(\tilde{\mathbf{W}}_1)$ et $C_1(\tilde{\mathbf{W}}_2) = C_2(\tilde{\mathbf{W}}_2) = -C_3(\tilde{\mathbf{W}}_2)$. Il y a alors deux zones affines, et si $\mathbf{x}$ et $\mathbf{y}$ ne sont pas dans la même zone, on a $J_{\boldsymbol{\theta}}(\mathbf{y}) - J_{\boldsymbol{\theta}}(\mathbf{x}) = \varepsilon \, C_1(\tilde{\mathbf{W}}_2) L_1(\tilde{\mathbf{W}}_1)^\top$ pour un certain $\varepsilon \in \{-1, 1\}$, et pourtant $\mathbf{x}$ et $\mathbf{y}$ ne sont pas 1-voisins. Cependant, sous l'hypothèse (H_1), l'implication énoncée au Lemme 3.8 devient une équivalence, car alors la famille $\varphi(\mathbf{W}_1, \mathbf{W}_2)$ est libre dans $\mathbb{R}^{N \times N}$. Il s'ensuit que, toujours sous cette hypothèse, les points i-voisins sont exactement ceux qui appartiennent à deux zones affines partageant à leur frontière une partie de l'hyperplan $\mathcal{H}_i$.*

Nous avons alors la condition suffisante suivante.

Proposition 3.10 (condition suffisante pour la reconstruction forte des biais, ayant déjà reconstruit les poids)**.** *On suppose que l'hypothèse* (H_1) *est satisfaite. Soit* $(\mathbf{x}_k)_{1\leq k\leq K}$ *une famille d'entrée telle que pour tout neurone* $i \in [\![N]\!]$*, il existe* $k, \ell \in [\![K]\!]$ *tels que* $\mathbf{x}_k$ *et* $\mathbf{x}_\ell$ *sont* i*-voisins. Alors la connaissance de* $(R_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$*,* $(J_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ *et de matrices de poids* $\mathbf{W}'_1, \mathbf{W}'_2$ *équivalentes à celles de* $\mathcal{R}$ *modulo changements d'échelle strictement positifs permet de construire des vecteurs de biais* $\mathbf{b}', \mathbf{c}'$ *induisant une reconstruction forte.*

Voir démonstration en annexe.

On peut remarquer que l'existence de i-voisins pour tout $i \in [\![N]\!]$ implique en particulier que les hyperplans associés aux différents neurones cachés sont deux à deux distincts – sous (H_1), ces deux conditions sont même équivalentes. Cela semble une contrainte raisonnable, puisque la situation où deux hyperplans sont égaux est de mesure nulle. Nous introduisons donc l'hypothèse suivante (H_2), qui sera utile dans la partie 3.2.

Hypothèse 3.11 (Hypothèse (H_2))**.** *Nous notons* (H_2) *l'hypothèse qu'il existe des points* i*-voisins pour tout neurone caché* i *de* $\mathcal{R}$*.*

On dispose toutefois d'une caractérisation qui se traduira algorithmiquement par une complexité d'échantillonnage nettement meilleure dans le cas (plus restrictif) où $\mathbf{W}_2$ a des colonnes linéairement indépendantes, hypothèse que nous notons (H_3).

Hypothèse 3.12 (Hypothèse (H_3))**.** *Nous notons* (H_3) *l'hypothèse que les colonnes de* $\mathbf{W}_2$ *sont linéairement indépendantes.*

Proposition 3.13. *Soit* $\mathcal{X} = (\mathbf{x}_k)_{1\leq k\leq K}$ *une famille d'entrée. On suppose que les hypothèses* (H_1) *et* (H_3) *sont satisfaites. Alors la donnée de* $(R_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ *et* $(J_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ *ainsi que de matrices de poids* $\mathbf{W}'_1, \mathbf{W}'_2$ *équivalentes à* $\mathbf{W}_1, \mathbf{W}_2$ *modulo changements d'échelle strictement positifs permet de reconstruire les biais associés si, et seulement si, pour tout neurone caché* i *de* $\mathcal{R}$*, il existe deux éléments* $\mathbf{x}_k$ *et* $\mathbf{x}_\ell$ *de* $\mathcal{X}$ *tels que* $\mathbf{x}_k$ *active* i *et* $\mathbf{x}_\ell$ *n'active pas* i*.*

Voir démonstration en annexe.

On peut à présent se demander si les différents résultats énoncés nous permettent d'établir une condition nécessaire et suffisante pour qu'une famille d'entrée soit fortement reconstruisante, sous certaines hypothèses sur le réseau à reconstruire. Nous n'avons pas encore trouvé de condition nécessaire et suffisante sous les hypothèses (H_1) et (H_2), car la condition suffisante énoncée dans la Proposition 3.10 n'est pas nécessaire en général. Nous énonçons donc une caractérisation des familles fortement reconstruisantes sous les hypothèses plus spécifiques (H_1) et (H_3), bien que tous les formats de matrice $\mathbf{W}_2$ ne permettent pas que (H_3) soit satisfaite.

Proposition 3.14. *On suppose que les hypothèses* (H_1) *et* (H_3) *sont satisfaites. Soit* $(\mathbf{x}_k)_{1\leq k\leq K}$ *une famille d'entrée. Alors la donnée de* $(J_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ *et* $(R_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ *permet une reconstruction forte si, et seulement si, les deux conditions suivantes sont simultanément satisfaites :*

(i) pour tout neurone caché $i \in [\![N]\!]$*, il existe deux vecteurs d'entrée* $\mathbf{x}_k$ *et* $\mathbf{x}_\ell$ *tels que* $\mathbf{x}_k$ *active* i *et* $\mathbf{x}_\ell$ *n'active pas* i*,*

(ii) pour tout $i \in [\![N]\!]$*, il existe* $k \in [\![K]\!]$ *tel que* $|\langle L_i(\mathbf{W}_1), \mathbf{x}_k\rangle| \geq |b_i|$*.*

Voir démonstration en annexe.

Nous pouvons alors borner uniformément le cardinal fortement reconstruisant (voir Définition 2.22) des réseaux d'une architecture donnée.

Proposition 3.15. *Le plus petit et le plus grand cardinal fortement reconstruisant associés à un réseau vérifiant les hypothèses* (H_1) *et* (H_3) *sont égaux, et valent 2.*

Voir démonstration en annexe.

3.2 Algorithmes de reconstruction

Une fois énoncées les garanties théoriques de la section précédente, on peut se demander comment échantillonner en pratique pour obtenir une famille reconstruisante, puis effectuer la reconstruction numériquement. Nous présentons dans cette partie les algorithmes développés pour mettre en œuvre une reconstruction forte, en distinguant le cas sans biais (plus simple) du cas avec biais.

Dans un souci de factorisation des raisonnements, commençons par définir un algorithme préliminaire qui applique la procédure de redressement, et auquel les algorithmes suivants font appel.

Algorithme 1 : Algorithme de redressement

Entrées : $(\mathbf{W}_1'', \mathbf{W}_2'')$ poids équivalents à $(\mathbf{W}_1, \mathbf{W}_2)$ modulo changements d'échelle inversibles, $\mathbf{x}$ vecteur d'entrée vérifiant $|\langle L_i(\mathbf{W}_1), \mathbf{x}\rangle| > |b_i|$ pour tout neurone caché i, et $J_{\boldsymbol{\theta}}(\mathbf{x})$ et $J_{\boldsymbol{\theta}}(-\mathbf{x})$ jacobiennes.

Sorties : $(\mathbf{W}_1', \mathbf{W}_2')$ poids équivalents à $(\mathbf{W}_1, \mathbf{W}_2)$ modulo changements d'échelle strictement positifs.

$\mathbf{W}_1', \mathbf{W}_2' \leftarrow \mathbf{W}_1'', \mathbf{W}_2''$;
pour $i = 1, \dots, N$ **faire**
  Calculer $\langle L_i(\mathbf{W}_1''), \mathbf{x}\rangle$ et $J_{\boldsymbol{\theta}}(\mathbf{x}) \odot \boldsymbol{\Sigma}_i$;
  si $\langle L_i(\mathbf{W}_1''), \mathbf{x}\rangle > 0$ **alors**
    si $J_{\boldsymbol{\theta}}(\mathbf{x}) \odot \boldsymbol{\Sigma}_i = \mathbf{0}_{\mathbb{R}^{s\times d}}$ **alors**
      multiplier $L_i(\mathbf{W}_1')$ et $C_i(\mathbf{W}_2')$ par -1
  sinon
    si $J_{\boldsymbol{\theta}}(\mathbf{x}) \odot \boldsymbol{\Sigma}_i \neq \mathbf{0}_{\mathbb{R}^{s\times d}}$ **alors**
      multiplier $L_i(\mathbf{W}_1')$ et $C_i(\mathbf{W}_2')$ par -1

Le fonctionnement de l'Algorithme 1 est le suivant. Les matrices $\mathbf{W}_1''$ et $\mathbf{W}_2''$ passées en entrée vérifient $\mathbf{W}_1'' = \mathbf{D}\mathbf{W}_1$ et $\mathbf{W}_2'' = \mathbf{W}_2\mathbf{D}^{-1}$ pour $\mathbf{D} = \mathrm{diag}(\lambda_1, \dots, \lambda_N)$ une certaine matrice diagonale inversible. Pour trouver $\mathbf{W}_1'$ et $\mathbf{W}_2'$ équivalentes à $\mathbf{W}_1$ et $\mathbf{W}_2$ modulo changements d'échelle strictement positifs, il suffit de changer le signe des lignes de $\mathbf{W}_1''$ et des colonnes de $\mathbf{W}_2''$ indexées par les i tels que $\lambda_i < 0$. Pour connaître le signe de chaque λ_i, on s'appuie sur la fonction suivante :

$$\mathcal{N} : \mathbf{x} \in \mathbb{R}^d \mapsto \{i \in [\![N]\!] \ ; \ \mathbf{x} \text{ active } i\}, \tag{13}$$

qui à tout vecteur d'entrée associe l'ensemble des neurones cachés qu'il active. L'ensemble $\mathcal{N}(\mathbf{x})$ peut être déterminé à partir de $J_{\boldsymbol{\theta}}(\mathbf{x})$. En effet, par hypothèse, les $\boldsymbol{\Sigma}_i$ sont deux à deux disjoints, et puisque l'on a supposé que $\mathbf{W}_1$ et $\mathbf{W}_2$ n'ont pas d'éléments nuls au sein de leurs supports respectifs $\mathbf{S}_1$ et $\mathbf{S}_2$, on en déduit que $\mathbf{x}$ active le neurone i si, et seulement si, $\boldsymbol{\Sigma}_i \odot J_{\boldsymbol{\theta}}(\mathbf{x}) \neq \mathbf{0}$. Dès lors, il faut changer le signe de $L_i(\mathbf{W}_1)$ et $C_i(\mathbf{W}_2)$ si, et seulement si, $[\langle L_i(\mathbf{W}_1), \mathbf{x}\rangle < 0$ et $i \in \mathcal{N}(\mathbf{x})]$ ou $[\langle L_i(\mathbf{W}_1), \mathbf{x}\rangle > 0$ et $i \notin \mathcal{N}(\mathbf{x})]$ (puisque par hypothèse $|\langle L_i(\mathbf{W}_1), \mathbf{x}\rangle| > |b_i|$, donc $\langle L_i(\mathbf{W}_1), \mathbf{x}\rangle + b_i$ est du signe de $\langle L_i(\mathbf{W}_1), \mathbf{x}\rangle$), ce qui fournit un algorithme de redressement.

Nous pouvons à présent écrire les algorithmes de reconstruction complets, en commençant par le cas sans biais.

3.2.1 Cas sans biais

La démonstration de la Proposition 3.15 nous fournit un algorithme simple et de complexité d'échantillonnage optimale pour la reconstruction forte d'un réseau qui vérifie l'hypothèse (H_1) et dont on sait qu'il est sans biais.

Algorithme 2 : Reconstruction forte dans le cas sans biais, sous (H_1)

Entrées : Deux applications échantillonnables en tout point : $R_{\boldsymbol{\theta}}$ et $J_{\boldsymbol{\theta}}$
Sorties : $(\mathbf{W}'_1, \mathbf{W}'_2)$ poids équivalents à $(\mathbf{W}_1, \mathbf{W}_2)$ modulo changements d'échelle strictement positifs

1 Tirer un vecteur $\mathbf{x} \in \mathbb{R}^d$ uniformément sur la sphère unité $\mathcal{S}^{d-1}$;
2 Évaluer $J_{\boldsymbol{\theta}}$ en $\mathbf{x}$ et $-\mathbf{x}$;
3 Calculer $\mathbf{W}_2\mathbf{W}_1 = J_{\boldsymbol{\theta}}(\mathbf{x}) + J_{\boldsymbol{\theta}}(-\mathbf{x})$;
4 Factoriser $\mathbf{W}_2\mathbf{W}_1$ avec l'Algorithme 2.13. On note $\mathbf{W}''_1$ et $\mathbf{W}''_2$ les facteurs obtenus ;
5 $\mathbf{W}'_1, \mathbf{W}'_2 \leftarrow$ sortie de l'Algorithme 1 avec $\mathbf{W}''_1, \mathbf{W}''_2, \mathbf{x}, J_{\boldsymbol{\theta}}(\mathbf{x}), J_{\boldsymbol{\theta}}(-\mathbf{x})$ comme paramètres d'entrée ;

Penchons-nous plus en détail sur certaines étapes de l'Algorithme 2.

Ligne 2 : Avec probabilité 1, la famille $(\mathbf{x}, -\mathbf{x})$ vérifie les conditions de la Proposition 3.3, donc est fortement reconstruisante. En effet, si ce n'est pas le cas, il existe $i \in [\![N]\!]$ tel que ni $\mathbf{x}$ ni $-\mathbf{x}$ n'activent i, c'est-à-dire tel que $\langle L_i(\mathbf{W}_1), \mathbf{x}\rangle = 0$. Mais $L_i(\mathbf{W}_1)^\perp \cap \mathcal{S}^{d-1}$ est de mesure nulle, d'où le résultat.

Remarque 3.16. *Une autre méthode d'échantillonnage, déterministe, consiste à prendre comme famille d'entrée* $(\mathbf{u}_1, \dots, \mathbf{u}_d, -\mathbf{u}_1, \dots, -\mathbf{u}_d)$*, où* $(\mathbf{u}_1, \dots, \mathbf{u}_d)$ *est une base quelconque de* $\mathbb{R}^d$*. Une telle famille est bien fortement reconstruisante, mais on voit que son cardinal dépend de la dimension de l'espace d'entrée, ce qui pénalise la complexité d'échantillonnage par rapport à l'Algorithme 2.*

Ligne 3 : La formule $\mathbf{W}_2\mathbf{W}_1 = J_{\boldsymbol{\theta}}(\mathbf{x}) + J_{\boldsymbol{\theta}}(-\mathbf{x})$ découle du Lemme 3.2, en remarquant que puisque l'on est dans le cas sans biais, $\mathbf{x}$ et $-\mathbf{x}$ ne peuvent activer simultanément un même neurone. Ainsi, avec probabilité 1, les neurones cachés activés par $\mathbf{x}$ forment exactement le complémentaire des neurones cachés activés par $-\mathbf{x}$, ce qui permet de conclure.

3.2.2 Cas avec biais

Penchons-nous à présent sur le cas avec biais : on n'impose plus que les biais du réseau inconnu $\mathcal{R}$ soient nuls. Supposons dans un premier temps que $\mathcal{R}$ vérifie les hypothèses (H_1) et (H_3). Sous les hypothèses de la Proposition 3.14, la reconstruction forte peut être effectuée avec l'Algorithme 3, qui est une adaptation directe de l'Algorithme 2 et exploite les idées présentées dans la démonstration de la Proposition 3.13.

La boucle de la ligne 4 est finie, car dès que le vecteur $\mathbf{x}$ est assez grand en module pour que $|\langle L_i(\mathbf{W}_1), \mathbf{x}\rangle| > |b_i|$ (rappelons que $\langle L_i(\mathbf{W}_1), \mathbf{x}\rangle$ est non nul avec probabilité 1), le neurone caché i a un état d'activation différent en $\mathbf{x}$ et en $-\mathbf{x}$. Notons qu'après l'arrêt du **tant que** (ligne 4), on obtient $\mathbf{x}$ tel que $\mathcal{N}(\mathbf{x}) \cup \mathcal{N}(-\mathbf{x}) = [\![N]\!]$ et $\mathcal{N}(\mathbf{x}) \cap \mathcal{N}(-\mathbf{x}) = \emptyset$, où $\mathcal{N}$ est définie dans l'équation (13). Ces deux conditions assurent que $(\mathbf{x}, -\mathbf{x})$ est alors une famille fortement reconstruisante, d'après les propositions 3.3 et 3.13, et justifient également l'égalité $\mathbf{W}_2\mathbf{W}_1 = J_{\boldsymbol{\theta}}(\mathbf{x}) + J_{\boldsymbol{\theta}}(-\mathbf{x})$, ainsi que la méthode de redressement employée dans l'Algorithme 1.

Algorithme 3 : Reconstruction forte dans le cas avec biais, sous (H_1) et (H_3)

Entrées : Deux applications échantillonnables en tout point : $R_{\boldsymbol{\theta}}$ et $J_{\boldsymbol{\theta}}$
Sorties : $(\mathbf{W}_1', \mathbf{W}_2', \mathbf{b}', \mathbf{c}')$ poids et biais équivalents à ceux de $\mathcal{R}$ modulo changements d'échelle strictement positifs

1 Tirer un vecteur $\mathbf{x} \in \mathbb{R}^d$ uniformément sur la sphère unité $\mathcal{S}^{d-1}$;
2 Évaluer $J_{\boldsymbol{\theta}}$ en $\mathbf{x}$ et $-\mathbf{x}$;
3 Déterminer $\mathcal{N}(\mathbf{x})$ et $\mathcal{N}(-\mathbf{x})$ en calculant $J_{\boldsymbol{\theta}}(\mathbf{x}) \odot \boldsymbol{\Sigma}_i$ et $J_{\boldsymbol{\theta}}(-\mathbf{x}) \odot \boldsymbol{\Sigma}_i$ pour tout $i \in [\![N]\!]$ (voir l'équation 13);
4 **tant que** $\mathcal{N}(\mathbf{x}) \cup \mathcal{N}(-\mathbf{x}) \neq [\![N]\!]$ *ou* $\mathcal{N}(\mathbf{x}) \cap \mathcal{N}(-\mathbf{x}) \neq \emptyset$ **faire**
5 | $\mathbf{x} \leftarrow 2\mathbf{x}$
6 Calculer $\mathbf{W}_2\mathbf{W}_1 = J_{\boldsymbol{\theta}}(\mathbf{x}) + J_{\boldsymbol{\theta}}(-\mathbf{x})$;
7 Factoriser $\mathbf{W}_2\mathbf{W}_1$ avec l'Algorithme 2.13. On note $\mathbf{W}_1''$ et $\mathbf{W}_2''$ les facteurs obtenus ;
8 $\mathbf{W}_1', \mathbf{W}_2' \leftarrow$ sortie de l'Algorithme 1 avec $\mathbf{W}_1'', \mathbf{W}_2'', \mathbf{x}, J_{\boldsymbol{\theta}}(\mathbf{x}), J_{\boldsymbol{\theta}}(-\mathbf{x})$ comme paramètres d'entrée ;
9 Calculer $\alpha_1, \dots, \alpha_N \in \mathbb{R}$ tels que $R_{\boldsymbol{\theta}}(\mathbf{x}) - R_{\boldsymbol{\theta}}(-\mathbf{x}) = \sum_{i=1}^{N} \alpha_i C_i(\mathbf{W}_2')$;
10 **pour** $i = 1, \dots, N$ **faire**
11 | **si** $i \in \mathcal{N}(\mathbf{x})$ **alors**
12 | | $b_i' \leftarrow \alpha_i - \langle L_i(\mathbf{W}_1'), \mathbf{x} \rangle$
13 | **sinon**
14 | | $b_i' \leftarrow -\alpha_i + \langle L_i(\mathbf{W}_1'), \mathbf{x} \rangle$
15 $\mathbf{c}' \leftarrow R_{\boldsymbol{\theta}}(\mathbf{x}) - \mathbf{W}_2' \,\mathtt{ReLU}(\mathbf{W}_1'\mathbf{x} + \mathbf{b}')$

Si en revanche l'hypothèse (H_3) ne tient pas, mais que l'on suppose (H_2) vérifiée, on peut retrouver les biais cachés par recherche de i-voisins (voir Définition 3.7), en s'appuyant sur la démonstration de la Proposition 3.10. Nous présentons l'implémentation de cette méthode dans l'Algorithme 4, dont le principe est, une fois que les poids du réseau ont été retrouvés modulo changements d'échelle strictement positifs, de se déplacer à pas fixé dans l'espace d'entrée de façon à s'approcher orthogonalement d'une frontière jusqu'à la dépasser, pour obtenir des couples de i-voisins pour tout neurone caché i. Détaillons certains points de l'Algorithme 4.

Ligne 2 : Les cases du tableau T_{voisins} sont indexées par les neurones cachés du réseau (en commençant à numéroter à zéro). Pour tout neurone caché $i \in [\![N]\!]$, si $T_{\text{voisins}}[i-1][0] = 0$, cela signifie que l'on n'a encore trouvé de couple de i-voisins, auquel cas la valeur de $T_{\text{voisins}}[i-1][1]$ n'a pas d'importance, et si $T_{\text{voisins}}[i-1][0] = 1$, alors $(\mathbf{x}, \mathbf{y}) := T_{\text{voisins}}[i-1][1]$ est un couple de i-voisins.

Lignes 7 à 20 : Puisqu'initialement on se déplace dans le sens opposé à celui de $L_i(\mathbf{W}_1)$, et que $\mathbf{z}$ active le neurone i, à partir d'un certain nombre d'itérations, $\mathbf{y}$ change de zone affine (le déplacement tend à la désactivation de i), ce qui se manifeste par un changement de jacobienne. S'il a franchi la frontière $\mathcal{H}_i$, ou bien $\mathbf{z}$ et $\mathbf{y}$ sont i-voisins (ce que l'on vérifie en exploitant la Remarque 3.9), auquel cas on les garde en mémoire puis on change de neurone d'intérêt, ou bien il a franchi au moins deux frontières en un pas, auquel cas on recommence le déplacement dans l'autre sens avec un pas divisé par deux jusqu'à changer à nouveau de zone. Si en revanche $\mathbf{y}$ n'a pas franchi la frontière $\mathcal{H}_i$, il faut continuer le déplacement dans le même sens et avec le même pas jusqu'à franchir une autre frontière, après avoir vérifié si $\mathbf{z}$ et $\mathbf{y}$ forment un couple de i'-voisins pour un autre neurone caché i', ce qui permet d'éviter des calculs inutiles. La procédure converge manifestement, et aboutit au remplissage complet du tableau T_{voisins}.

Algorithme 4 : Reconstruction forte dans le cas avec biais, sous (H_1) et (H_2)

Entrées : Deux applications échantillonnables en tout point : $R_{\boldsymbol{\theta}}$ et $J_{\boldsymbol{\theta}}$, et un pas $\eta > 0$
Sorties : $(\mathbf{W}_1', \mathbf{W}_2', \mathbf{b}', \mathbf{c}')$ poids et biais équivalents à ceux de $\mathcal{R}$ modulo changements d'échelle strictement positifs

1 Appliquer les lignes 1 à 8 de l'Algorithme 3 ;
2 Définir T_{voisins} tableau de taille N initialisé à $(0, (\mathbf{0}_d, \mathbf{0}_d))$ dans chaque case ;
3 **pour** $\mathbf{z} \in \{\mathbf{x}, -\mathbf{x}\}$ **faire**
4 **pour** $i \in \mathcal{N}(\mathbf{z})$ **faire**
5 **si** $T_{\text{voisins}}[i][0] = 0$ **alors**
6 $\mathbf{y} \leftarrow \mathbf{z} - \eta L_i(\mathbf{W}_1')$;
7 **tant que** $\mathbf{y}$ *n'a pas changé de zone affine* **faire**
8 $\mathbf{y} \leftarrow \mathbf{z} - \eta L_i(\mathbf{W}_1')$;
9 **si** $\mathbf{y}$ *et* $\mathbf{z}$ *sont* i*-voisins* **alors**
10 $T_{\text{voisins}}[i] \leftarrow (1, (\mathbf{z}, \mathbf{y}))$;
11 **sinon**
12 **si** $\mathbf{y}$ *et* $\mathbf{z}$ *sont* i'*-voisins pour un certain* $i' \neq i$ **alors**
13 $T_{\text{voisins}}[i'] \leftarrow (1, (\mathbf{z}, \mathbf{y}))$ si $i' \in \mathcal{N}(\mathbf{z})$, et $(1, (\mathbf{y}, \mathbf{z}))$ sinon ;
14 Calculer $J_{\boldsymbol{\theta}}(\mathbf{y})$;
15 **si** $i \in \mathcal{N}(\mathbf{y})$, *i.e.* $J_{\boldsymbol{\theta}}(\mathbf{y}) \odot \boldsymbol{\Sigma}_i \neq \mathbf{0}_{\mathbb{R}^{s \times d}}$ **alors**
16 $\mathbf{z}, \mathbf{y} \leftarrow \mathbf{y}, \mathbf{y} - \eta L_i(\mathbf{W}_1')$;
17 réitérer la procédure à partir de la ligne 7 ;
18 **sinon**
19 $\eta \leftarrow -\eta/2$;
20 réitérer la procédure à partir de la ligne 7 ;
21 Calculer $\mathbf{b}'$ avec T_{voisins} en suivant la démonstration de la Proposition 3.10 ;
22 $\mathbf{c}' \leftarrow R_{\boldsymbol{\theta}}(\mathbf{x}) - \mathbf{W}_2' \,\mathtt{ReLU}(\mathbf{W}_1'\mathbf{x} + \mathbf{b}')$

3.3 Cas dégénéré

Dans tout ce qui précède, on a supposé que $\text{Supp}(\mathbf{W}_1) = \mathbf{S}_1$ et $\text{Supp}(\mathbf{W}_2) = \mathbf{S}_2$ avec $(\mathbf{S}_1, \mathbf{S}_2)$ admissible et connu. Qu'est-ce qui change si l'on suppose simplement $\text{Supp}(\mathbf{W}_1) \subset \mathbf{S}_1$ et $\text{Supp}(\mathbf{W}_2) \subset \mathbf{S}_2$, avec $(\mathbf{S}_1, \mathbf{S}_2)$ admissible et connu, mais $\text{Supp}(\mathbf{W}_1)$ et $\text{Supp}(\mathbf{W}_2)$ inconnus – en particulier, sans avoir nécessairement $(\text{Supp}(\mathbf{W}_1), \text{Supp}(\mathbf{W}_2))$ admissible ? Nous appellerons (D) cette hypothèse.

Hypothèse 3.17 (Hypothèse (D))**.** *Soit* $(\mathbf{S}_1, \mathbf{S}_2)$ *un couple admissible de supports, connu. Notons* $\mathbf{W}_1 \in \mathbb{R}^{N \times d}$ *et* $\mathbf{W}_2 \in \mathbb{R}^{s \times N}$ *les matrices de poids et* $(\mathbf{b}, \mathbf{c}) \in \mathbb{R}^N \times \mathbb{R}^s$ *les vecteurs de biais du réseau à reconstruire, dont on connaît l'architecture. On suppose que les supports de* $\mathbf{W}_1$ *et* $\mathbf{W}_2$ *sont respectivement inclus dans* $\mathbf{S}_1$ *et* $\mathbf{S}_2$*.*

Il est facile de vérifier que les couples $(\mathbf{W}_1, \mathbf{W}_2)$ vérifiant (D) et tels que $(\text{Supp}(\mathbf{W}_1), \text{Supp}(\mathbf{W}_2))$ n'est pas admissible sont exactement ceux vérifiant les deux conditions suivantes :

(i) $\text{Supp}(\mathbf{W}_1) \subset \mathbf{S}_1$ et $\text{Supp}(\mathbf{W}_2) \subset \mathbf{S}_2$,

(ii) $\mathbf{W}_1$ a au moins une ligne nulle ou $\mathbf{W}_2$ a au moins une colonne nulle.

Nous dirons que de tels couples sont dégénérés. Un réseau ayant un couple de matrices de poids dégénéré sera également dit dégénéré. On voit que la condition (*ii*) équivaut à l'existence d'un neurone dégénéré dans le réseau, au sens de la Définition 3.5.

Plaçons-nous donc sous l'hypothèse affaiblie (D), que nous appelons "cas dégénéré", au sens où il comprend des réseaux dégénérés. Certains résultats de la partie précédente ne sont plus valables. Tout d'abord, une reconstruction forte n'est pas toujours envisageable. En effet, supposons par exemple qu'il existe i tel que $C_i(\mathbf{W}_2)$ est nul. Alors la i-ème composante de rang 1 de $\mathbf{W}_2\mathbf{W}_1$ est nulle, et l'on ne peut retrouver les coefficients non nuls de $L_i(\mathbf{W}_1)$ s'il y en a, ceux-ci apportant une contribution nulle à $R_{\boldsymbol{\theta}}$ et $J_{\boldsymbol{\theta}}$. Ainsi, nous pouvons tout au plus arriver à une reconstruction faible en général (voir Définition 2.6). Une deuxième remarque est que certains neurones cachés peuvent ne plus être activables : c'est le cas des neurones associés à une ligne nulle de $\mathbf{W}_1$ et à un biais négatif. D'autres, au contraire, peuvent être actifs sur tout l'espace d'entrée, s'ils sont associés à une ligne nulle de $\mathbf{W}_1$ et à un biais strictement positif. D'autres, enfin, qu'ils soient activables ou non, n'interviennent pas du tout dans la réalisation : ce sont les i tels que $C_i(\mathbf{W}_2)$ est nul. Voyons comment s'adaptent nos résultats dans ce cadre, en distinguant les cas sans biais et avec biais.

3.3.1 Cas dégénéré sans biais

Dans le cas dégénéré sans biais, on peut adapter la Proposition 3.3 pour énoncer une caractérisation des familles permettant une reconstruction faible.

Proposition 3.18. *Supposons que le réseau inconnu est sans biais et vérifie l'hypothèse (D). Soit $(\mathbf{x}_k)_{1\leq k\leq K}$ une famille d'entrée. La connaissance de $(J_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ et $(R_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ permet de construire des poids fonctionnellement équivalents à ceux de $\mathcal{R}$ si, et seulement si pour tout neurone caché activable $i \in [\![N]\!]$, il existe $k \in [\![K]\!]$ tel que $\mathbf{x}_k$ active i.*

Voir démonstration en annexe.

Remarque 3.19. *Comme dans le cas de la Proposition 3.3, connaître $(J_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ suffit à la reconstruction faible, sous la condition et les hypothèses énoncées dans la Proposition 3.18.*

L'encadrement du cardinal fortement reconstruisant de réseaux sans biais vérifiant l'hypothèse (D) s'obtient comme dans le cas non dégénéré : la plus petite famille faiblement reconstruisante minimale est réduite à un vecteur, et la plus grande à deux vecteurs. L'algorithme de reconstruction s'adapte de l'Algorithme 2 selon les indications données dans la démonstration de la Proposition 3.18. En somme, tout se transpose bien au cas dégénéré quand on impose des biais nuls.

3.3.2 Cas dégénéré avec biais

Dans le cas dégénéré avec biais, les Algorithmes 3 et 4 ne peuvent être appliqués tels quels, car alors le **tant que** de la ligne 4 ne termine pas toujours. Étant donné $\mathbf{x}$ et $-\mathbf{x}$, on n'a en fait aucun moyen de vérifier si ces deux vecteurs activent tous les neurones cachés non dégénérés. Nous proposons deux façons de contourner ce problème.

La première est de rajouter une hypothèse sur les paramètres du réseau pour pouvoir vérifier qu'à partir d'un certain nombre d'itérations sur $\mathbf{x}$, on active bien tous les neurones non dégénérés.

Hypothèse 3.20. *On suppose que l'on connaît une constante $\alpha > 0$ telle que pour tout neurone i non dégénéré associé à un biais strictement négatif :*

$$\alpha \, \|L_i(\mathbf{W}_1)\| > |b_i|. \tag{14}$$

On peut alors reconstruire faiblement les poids du réseau.

Lemme 3.21. *Sous les hypothèses (D) et 3.20, et en notant* $(\mathbf{e}_1, \dots, \mathbf{e}_d)$ *la base canonique de* $\mathbb{R}^d$*, la famille* $(\alpha\sqrt{d}\,\mathbf{e}_1, \dots, \alpha\sqrt{d}\,\mathbf{e}_d, -\alpha\sqrt{d}\,\mathbf{e}_1, \dots, -\alpha\sqrt{d}\,\mathbf{e}_d)$ *permet de reconstruire faiblement les poids de* $\mathcal{R}$*.*

Voir démonstration en annexe.

Quant à la méthode de reconstruction des biais dans le cas non dégénéré, elle s'adapte bien ici, que l'on ajoute (H_2) ou (H_3) à (D).

Si en revanche on ne dispose d'aucune information du type de l'Hypothèse 3.20, on peut se contenter de reconstruire un réseau dont la réalisation coïncide avec $R_{\boldsymbol{\theta}}$ sur une partie de l'espace d'entrée, en négligeant les coefficients sur lesquels la famille d'entrée choisie ne donne pas d'information. Cette idée se fonde sur la remarque suivante.

Lemme 3.22. *On se place sous l'hypothèse (D). Soit* $\mathcal{X} = (\mathbf{x}_1, \dots, \mathbf{x}_K)$ *une famille d'entrée, et* $\mathcal{E}$ *l'enveloppe convexe de* $\mathcal{X}$*. Notons* $R_{\boldsymbol{\theta}'}$ *la réalisation obtenue en exploitant* $(J_{\boldsymbol{\theta}}(\mathbf{x}_k))_k$ *et* $(R_{\boldsymbol{\theta}}(\mathbf{x}_k))_k$ *pour la reconstruction des poids et des biais de* $\mathcal{R}$*, en mettant à 0 les coefficients sur lesquels on n'a pas obtenu d'information. Alors* $R_{\boldsymbol{\theta}}$ *et* $R_{\boldsymbol{\theta}'}$ *coïncident sur* $\mathcal{E}$*.*

Voir démonstration en annexe.

Dans cette perspective, prendre comme famille d'entrée $(A\,\mathbf{e}_1, \dots, A\,\mathbf{e}_d, -A\,\mathbf{e}_1, \dots, -A\,\mathbf{e}_d)$ semble une bonne solution, avec $A > 0$ assez grand, et $(\mathbf{e}_1, \dots, \mathbf{e}_d)$ la base canonique de $\mathbb{R}^d$ ou toute autre base de vecteurs unitaires de déterminant égal à 1 ou -1 (pour maximiser le volume de l'enveloppe convexe de la base de vecteurs unitaires choisie).

Penchons-nous à présent sur le cas multicouche, et plus spécifiquement sur les réseaux de type butterfly.

4 Reconstruction de réseaux butterfly multicouche

Pour pouvoir reconstruire un réseau inconnu comportant plusieurs couches cachées en exploitant les résultats de factorisation matricielle dont nous disposons, il nous faut avant tout imposer des supports adéquats aux matrices de poids du réseau. Au lieu d'adopter les hypothèses minimales de la Proposition 2.16, nous avons décidé de travailler avec des supports butterfly (voir Définition 2.17), ce qui permet de raisonner sur des exemples concrets comme d'exploiter certaines propriétés intéressantes de cette famille de supports. Soit donc $\mathcal{R}$ un réseau butterfly, c'est-à-dire dont les matrices de poids ont des supports butterfly, à $J+1$ couches avec $J \geq 2$, et de paramètres inconnus. On note $\mathbf{W}_1, \dots, \mathbf{W}_J \in \mathbb{R}^{N\times N}$ ses poids et $\mathbf{b}_1, \dots, \mathbf{b}_J \in \mathbb{R}^N$ ses biais, avec $N = 2^J$. On se place sous l'hypothèse (H) suivante :

Hypothèse 4.1 (Hypothèse (H))**.** *Soit* $J \geq 2$ *et* $N = 2^J$*. Pour tout* $j \in [\![J]\!]$*, notons* $\mathbf{S}_j \in \mathbb{R}^{N\times N}$ *le* j*-ème support butterfly de taille* N*. Soit* $\mathbf{W}_1, \dots, \mathbf{W}_J \in \mathbb{R}^{N\times N}$ *les poids du réseau à reconstruire, et* $\mathbf{b}_1, \dots, \mathbf{b}_J \in \mathbb{R}^N$ *ses biais. On suppose que l'on sait que les* $\mathbf{W}_j$ *n'ont pas de coefficients nuls au sein des supports butterfly, c'est-à-dire que pour tout* $j \in [\![J]\!]$*, le support* $\mathrm{Supp}(\mathbf{W}_j)$ *est égal à* $\mathbf{S}_j$*.*

Rappelons que la réalisation $R_{\boldsymbol{\theta}}$ et la jacobienne $J_{\boldsymbol{\theta}}$ de $\mathcal{R}$ s'expriment de la façon suivante :

$$R_{\boldsymbol{\theta}} : \mathbf{x} \in \mathbb{R}^N \mapsto \mathbf{W}_J\,\mathtt{ReLU}(\mathbf{W}_{J-1}\,\mathtt{ReLU}(\dots\mathtt{ReLU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \dots) + \mathbf{b}_{J-1})) + \mathbf{b}_J, \tag{15}$$

et

$$J_{\boldsymbol{\theta}} : \mathbf{x} \in \mathbb{R}^N \mapsto \mathbf{W}_J\mathbf{D}_{J-1}\mathbf{W}_{J-1}\dots\mathbf{D}_1\mathbf{W}_1, \tag{16}$$

avec $\mathbf{D}_j = \operatorname{diag}\left(\mathbf{1}_{(z_j(\mathbf{x}))_i > 0}\right)_{1 \leq i \leq N}$, où z_j est introduite dans la Définition 2.2.

En premier lieu, il s'agit de comprendre quelle information l'évaluation de $J_{\boldsymbol{\theta}}$ en un point d'entrée donne sur les paramètres du réseau. Dans l'optique d'établir un résultat analogue à celui du Lemme 3.2, introduisons les notions de chemin complet et chemin d'activation, inspirées de [6], en adoptant le point de vue par graphe sur les réseaux de neurones.

Définition 4.2 (chemin complet, chemin d'activation)**.** *Soit $\mathcal{R}$ un réseau butterfly. Dans le graphe de $\mathcal{R}$, on ne garde que les arêtes correspondant aux coefficients contenus dans les supports butterfly. Nous reprenons les notations de la Définition 2.3. Une famille d'arêtes consécutives $(\nu_j \to \nu_{j+1})_{0 \leq j \leq J-1}$ avec $\nu_j \in \mathcal{V}_j$ pour tout j est appelée un* chemin complet *de $\mathcal{R}$ (voir Figure 5). On pourra représenter une telle famille de façon équivalente par le $(J+1)$-uplet de ses neurones $(\nu_0, \ldots, \nu_J)$. Un* chemin d'activation *désigne une famille d'arêtes cachées consécutives $(\nu_j \to \nu_{j+1})_{1 \leq j \leq J-2}$ avec $\nu_j \in \mathcal{V}_j$ pour tout j, ou encore le $(J-1)$-uplet de neurones cachés $(\nu_1, \ldots, \nu_{J-1})$ associé.*

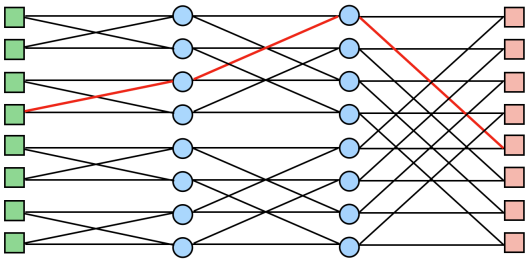


FIGURE 5 – Graphe d'un réseau butterfly à 2 couches cachées. En rouge, un chemin complet. Les chemins d'activation correspondent ici aux arêtes reliant deux neurones cachés.

On voit que les chemins complets relient la couche d'entrée à la couche de sortie, alors que les chemins d'activation vont de la première à la dernière couche cachée. Chaque chemin complet contient un unique chemin d'activation, et chaque chemin d'activation est contenu dans exactement quatre chemins complets. On compte N^2 chemins complets, et $N^2/4$ chemins d'activation. Comme on a défini l'état d'activation d'un neurone, on peut définir celui d'une arête, et d'un chemin.

Définition 4.3 (état d'activation d'un chemin, d'une arête)**.** *Un chemin d'activation est dit activé par un vecteur d'entrée $\mathbf{x}$ si, pour tout neurone ν dans le chemin, la préactivation de ν en $\mathbf{x}$ est strictement positive. Un chemin complet est dit activé par $\mathbf{x}$ si l'unique chemin d'activation associé est activé par $\mathbf{x}$. On dira de plus que $\mathbf{x}$ active l'arête $\nu \to \nu'$ si ν et ν' sont actifs en $\mathbf{x}$.*

On définit l'ensemble $\mathcal{C}$ des chemins activés par une famille d'entrée $\mathcal{X}$ comme l'union des chemins activés par chaque élément de la famille. Nous avons alors le résultat suivant, valable pour tout réseau ReLU à J couches.

Lemme 4.4. *Soit $\mathbf{x} \in \mathbb{R}^N$ un vecteur d'entrée. Pour tous $\nu_0 \in \mathcal{V}_0$ et $\nu_J \in \mathcal{V}_J$, on a :*

$$(J_{\boldsymbol{\theta}}(\mathbf{x}))_{\nu_J, \nu_0} = \sum_{(\nu_0, \ldots, \nu_J) \text{ chemin complet}} \mathbf{1}_{\{(\nu_0, \ldots, \nu_J) \text{ est activé par } \mathbf{x}\}} \prod_{j=1}^{J} (\mathbf{W}_j)_{\nu_j, \nu_{j-1}}. \tag{17}$$

Or, dans le cas de réseaux butterfly, pour tous $\nu_0 \in \mathcal{V}_0$ et $\nu_J \in \mathcal{V}_J$, il existe un unique chemin complet reliant ν_0 à ν_J, ce qui se réexprime plus formellement comme suit.

Lemme 4.5. *Soient $\mathbf{S}_1, \dots, \mathbf{S}_J$ les J supports butterfly de taille $N = 2^J$. En notant $\mathbf{1}_N$ la matrice carrée de côté N ne contenant que des 1, on a :*

$$\mathbf{S}_J \dots \mathbf{S}_1 = \mathbf{1}_N. \tag{18}$$

Voir démonstration en annexe.

On en déduit l'expression de la jacobienne pour un réseau ReLU de type butterfly.

Corollaire 4.6. *Soit $\mathbf{x} \in \mathbb{R}^N$ un vecteur d'entrée, et $\nu_0 \in [\![N]\!]$ un neurone d'entrée, $\nu_J \in [\![N]\!]$ un neurone de sortie. Pour un réseau butterfly quelconque ayant pour matrices de poids $\mathbf{W}_1, \dots, \mathbf{W}_J$, on a, en notant $(\nu_0, \nu_1, \dots, \nu_J)$ l'unique chemin complet reliant ν_0 à ν_J :*

$$(J_{\boldsymbol{\theta}}(\mathbf{x}))_{\nu_J,\nu_0} = \mathbf{1}_{\{(\nu_0,\nu_1,\dots,\nu_J) \text{ est activé par } \mathbf{x}\}} \prod_{j=1}^{J} (\mathbf{W}_j)_{\nu_j,\nu_{j-1}}. \tag{19}$$

Remarque 4.7. *Il découle du résultat précédent que sous l'hypothèse (H), observer $J_{\boldsymbol{\theta}}(\mathbf{x})$ pour un vecteur d'entrée $\mathbf{x}$ donné permet de déterminer exactement les chemins complets activés par $\mathbf{x}$. Précisément, pour tous neurones $\nu_0 \in \mathcal{V}_0$ et $\nu_J \in \mathcal{V}_J$, l'unique chemin reliant ν_0 à ν_J est activé par $\mathbf{x}$ si, et seulement si, $(J_{\boldsymbol{\theta}}(\mathbf{x}))_{\nu_J,\nu_0} \neq 0$. En effet, considérons l'unique chemin $\nu_0 \rightsquigarrow \nu_J$ reliant le neurone d'entrée ν_0 au neurone de sortie ν_J, avec $\nu_0, \nu_J \in [\![N]\!]$ quelconques. Si $(J_{\boldsymbol{\theta}}(\mathbf{x}))_{\nu_J,\nu_0} \neq 0$, alors d'après le Corollaire 4.6, le chemin $\nu_0 \rightsquigarrow \nu_J$ est activé par $\mathbf{x}$. Si en revanche on a $(J_{\boldsymbol{\theta}}(\mathbf{x}))_{\nu_J,\nu_0} = 0$, puisque d'après l'hypothèse (H) le produit $\prod_{j=1}^{J} (\mathbf{W}_j)_{\nu_j,\nu_{j-1}}$ est non nul, où l'on a noté $\nu_0 \rightsquigarrow \nu_J = (\nu_0, \nu_1, \dots, \nu_J)$, nécessairement le vecteur $\mathbf{x}$ n'active pas le chemin $\nu_0 \rightsquigarrow \nu_J$.*

La sous-section suivante introduit une matrice que l'on note $\mathbf{P}$, centrale dans l'étude des familles fortement reconstruisantes pour des réseaux butterfly.

4.1 La matrice P

Nous avons eu l'idée de traduire la propriété du Lemme 4.5 par une dépendance linéaire entre le logarithme des matrices butterfly et celui de leurs facteurs, après vectorisation.

Corollaire/Définition 4.8 (Matrice $\mathbf{P}$)**.** *Soit $\mathbf{A} = \mathbf{X}_J \dots \mathbf{X}_1 \in \mathbb{R}^{N \times N}$ une matrice butterfly, avec $\mathrm{Supp}(\mathbf{X}_j) = \mathbf{S}_j$ pour tout $j \in [\![J]\!]$. Les coefficients de $\mathbf{A} = (a_{i,j})_{i,j}$ sont alors non nuls. En notant $\log$ la détermination complexe du logarithme ayant $-i\mathbb{R}$ comme ligne de coupure, de sorte que*

$$\log a_{i,j} = \begin{cases} \ln a_{i,j} \ \textit{si} \ a_{i,j} > 0 \\ \ln a_{i,j} + i\pi \ \textit{si} \ a_{i,j} < 0, \end{cases} \tag{20}$$

on peut considérer la matrice $\log \mathbf{A} := (\log a_{i,j})_{i,j}$. On choisit un ordre pour les coefficients de $\mathbf{A}$, et un ordre pour ceux (non nuls) de $\mathbf{X} = (\mathbf{X}_1, \dots, \mathbf{X}_J)$. Alors il existe une matrice $\mathbf{P} \in \{0,1\}^{N^2 \times 2NJ}$, indépendante de la valeur des $2NJ$ coefficients non nuls de $\mathbf{X}_1, \dots, \mathbf{X}_J$, telle que :

$$\mathrm{vec}(\log \mathbf{A}) = \mathbf{P}\, \mathrm{vec}(\log \mathbf{X}), \tag{21}$$

où la vectorisation se fait suivant l'ordre choisi.

La matrice $\mathbf{P}$ nous sera utile dans la suite pour caractériser les familles fortement reconstruisantes. Fixons donc un ordre pour la vectorisation de la matrice produit d'une part, et des coefficients des facteurs d'autre part, et considérons $\mathbf{P} \in \{0,1\}^{N^2 \times 2NJ}$ la matrice associée. Cette matrice a un noyau non trivial, qui traduit l'invariance du produit $\mathbf{X}_J \dots \mathbf{X}_1$ par changements d'échelle inversibles sur les lignes et colonnes.

Lemme 4.9. *Le noyau de* $\mathbf{P}$ *est de dimension* $N(J-1)$*, soit le nombre de neurones cachés du réseau.*

Voir démonstration en annexe.

Remarque 4.10. *Chacune des* $2NJ$ *colonnes de* $\mathbf{P}$ *correspond naturellement à un élément du support butterfly de l'une des matrices de poids du réseau, ou encore à une arête de son graphe butterfly. Avec ce dernier point de vue, on peut associer à chaque ligne de* $\mathbf{P}$ *un chemin complet, dont les arêtes sont données par les colonnes contenant un coefficient 1 sur la ligne en question. Ceci établit une correspondance bijective entre lignes de* $\mathbf{P}$ *et chemins complets dans le graphe du réseau. Nous identifierons donc ces deux objets dans la suite. En particulier, nous pourrons dire qu'un vecteur d'entrée active une ligne de* $\mathbf{P}$ *s'il active le chemin complet correspondant.*

Nous pouvons à présent étudier les familles fortement reconstruisantes pour les réseaux butterfly.

4.2 Caractérisation des familles fortement reconstruisantes

Dans la recherche d'une condition nécessaire et suffisante pour qu'une famille d'entrée soit fortement reconstruisante, il convient de dissocier le raisonnement en trois étapes de nature distincte. La première consiste à identifier des conditions pour qu'une famille d'entrée permette la reconstruction des poids modulo changements d'échelle inversibles, ce qui revient à comprendre quels neurones, ou plutôt, quels chemins il faut activer pour retrouver une portion significative du produit des matrices de poids et pouvoir factoriser ensuite. La deuxième porte sur le redressement : quelles familles d'entrée apportent suffisamment d'information pour changer correctement le signe des lignes et colonnes mal orientées après factorisation ? Enfin, dans le cas avec biais, on peut se demander à quelle condition une famille d'entrée permet de retrouver les biais, connaissant les poids précédemment reconstruits. Nous regroupons ces trois propriétés dans une définition.

Définition 4.11 (propriétés (F), (R) et (B))**.** *Soit* $\mathcal{X} = (\mathbf{x}_1, \dots, \mathbf{x}_K)$ *une famille d'entrée.*

- *On dit que* $\mathcal{X}$ *vérifie la propriété* (F) *si la connaissance de* $(R_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ *et* $(J_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ *permet la reconstruction des poids du réseau inconnu* $\mathcal{R}$ *modulo changements d'échelle inversibles.*
- *Elle vérifie la propriété* (R) *si, connaissant les poids de* $\mathcal{R}$ *modulo changements d'échelle inversibles, elle permet par l'observation de* $(R_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ *et* $(J_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ *de remonter à des poids équivalents à ceux de* $\mathcal{R}$ *modulo changements d'échelle strictement positifs.*
- *On dit enfin que la famille* $\mathcal{X}$ *vérifie la propriété* (B) *si, connaissant les poids de* $\mathcal{R}$ *modulo changements d'échelle strictement positifs, elle permet par l'observation de* $(R_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ *et* $(J_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ *de reconstruire les biais associés aux poids connus.*

4.2.1 Reconstruction des poids modulo changements d'échelle inversibles

Avant d'établir une condition nécessaire et suffisante sur les familles ayant la propriété (F), nous pouvons énoncer une condition nécessaire (qui nous semble non suffisante dès que $J \geq 4$)

et une condition suffisante (non nécessaire dès que $J \geq 4$) naturelles, qui constituent une bonne introduction au problème.

Proposition 4.12. *Soit $(\mathbf{x}_k)_{1\leq k\leq K}$ une famille vérifiant la propriété (F). Alors pour toute arête e de $\mathcal{R}$, il existe un chemin d'activation c contenant e, et un indice $k \in [\![K]\!]$, tels que c est activé par $\mathbf{x}_k$.*

Voir démonstration en annexe.

Nous énonçons ensuite une condition suffisante.

Proposition 4.13. *Soit $(\mathbf{x}_k)_{1\leq k\leq K}$ tel que pour tous $i, j \in [\![N]\!]$, il existe $k \in [\![K]\!]$ tel que $(J_{\boldsymbol{\theta}}(\mathbf{x}_k))_{i,j} \neq 0$. Sous l'hypothèse (H), c'est équivalent au fait que pour tout chemin d'activation c du réseau, il existe $k \in [\![K]\!]$ tel que $\mathbf{x}_k$ active c. Alors $(\mathbf{x}_k)$ vérifie la propriété (F).*

Voir démonstration en annexe.

On dispose en fait d'une caractérisation exacte des familles vérifiant la propriété (F), sous l'hypothèse (H). Soit $\mathbf{B} \in \mathbb{R}^{2NJ\times N(J-1)}$ une base de $\ker \mathbf{P}$. La matrice $\left(\begin{smallmatrix}\mathbf{P}\\ \mathbf{B}^\top\end{smallmatrix}\right) \in \mathbb{R}^{(N^2+N(J-1))\times 2NJ}$ définie par blocs est injective par construction. Nous avons alors le résultat central suivant.

Proposition 4.14. *On suppose que $\mathcal{R}$ vérifie l'hypothèse (H). Soit $\mathcal{X} = (\mathbf{x}_k)_{1\leq k\leq K}$ une famille d'entrée. Notons $\mathbf{P}'$ la sous-matrice de $\mathbf{P}$ constituée des lignes activées par $\mathcal{X}$. Les conditions suivantes sont équivalentes :*

(i) *$\mathcal{X}$ vérifie la propriété (F),*

(ii) *la matrice $\mathbf{Q}' = \begin{pmatrix}\mathbf{P}'\\ \mathbf{B}^\top\end{pmatrix}$ est injective,*

(iii) $\operatorname{rg}\mathbf{P}' = \operatorname{rg}\mathbf{P}$.

Voir démonstration en annexe.

Quand $J \leq 3$, la condition nécessaire énoncée dans la Proposition 4.12 est également suffisante. En effet, dans ce cas, les chemins d'activation sont exactement les arêtes cachées (si $J = 3$) ou les neurones cachés (si $J = 2$), donc sous la condition de la Proposition 4.12, tout chemin d'activation est activé par au moins un vecteur de la famille considérée, ce qui, on l'a vu, est une condition suffisante pour avoir la propriété (F). En revanche, nous conjecturons que la condition de la Proposition 4.12 n'est plus suffisante dès que $J \geq 4$.

De même, la condition suffisante énoncée dans la Proposition 4.13 n'est pas nécessaire dès que $J \geq 4$, comme on peut le voir numériquement en se donnant un réseau butterfly sans biais et en tirant aléatoirement des vecteurs d'entrée jusqu'à obtenir une famille vérifiant les conditions de la Proposition 4.14 – ce que l'on peut vérifier d'après la Remarque 4.7 – mais pas celles de la Proposition 4.13, quitte à répéter plusieurs fois l'expérience.

Il semble intéressant de reformuler la condition nécessaire et suffisante énoncée dans la Proposition 4.14 en termes d'activation de chemins, pour avoir une caractérisation plus visuelle. Nous ne disposons que d'une conjecture à ce stade, pour l'énoncé de laquelle il nous faut introduire un peu de terminologie.

Définition 4.15 (successeurs, descendants, prédécesseurs, ancêtres)**.** *Soit ν un neurone de $\mathcal{R}$, situé dans la j-ème couche. Si $j < J$, on appelle* successeurs *de ν les deux neurones de la couche $j+1$ reliés à ν par une arête (dans le graphe de $\mathcal{R}$), et* descendants *de ν les neurones de la couche de sortie reliés à ν par un chemin allant de la couche j à la couche de sortie. Si $j > 0$, on appelle* prédécesseurs *de ν les deux neurones de la couche $j-1$ reliés à ν par une arête, et* ancêtres *de ν les neurones de la couche d'entrée reliés à ν par un chemin allant de la couche d'entrée à la couche j.*

Conjecture 4.16. *On suppose que $\mathcal{R}$ vérifie l'hypothèse (H). Soit $\mathcal{X} = (\mathbf{x}_k)_{1\leq k\leq K}$ une famille d'entrée. Alors $\mathcal{X}$ vérifie la propriété (F) si, et seulement si, pour tout neurone caché ν de $\mathcal{R}$, il existe un neurone d'entrée μ_0 et deux chemins complets c_1 et c_2 tels que :*

(i) *c_1 et c_2 partent de μ_0 et passent par ν (ils coïncident sur l'unique chemin partiel reliant μ_0 à ν, voir Figure 5),*

(ii) *en notant μ_1 et μ_2 les successeurs de ν, le chemin c_1 passe par μ_1 et le chemin c_2 par μ_2,*

(iii) *il existe deux vecteurs $\mathbf{x}$ et $\mathbf{y}$ de $\mathcal{X}$ tels que $\mathbf{x}$ active le chemin c_1 et $\mathbf{y}$ active le chemin c_2.*

Nous avons constaté numériquement jusqu'à $J = 9$ que cette condition était suffisante. D'autres essais numériques semblent indiquer qu'elle est également nécessaire.

4.2.2 Redressement

Dans le cas sans biais, on dispose d'une condition nécessaire et suffisante simple pour qu'une famille d'entrée permette le redressement des poids du réseau en construction.

Proposition 4.17. *Supposons que l'on impose que $\mathcal{R}$ soit sans biais et vérifie (H). Une famille d'entrée $\mathcal{X} = (\mathbf{x}_k)_{1\leq k\leq K}$ vérifie la propriété (R) si, et seulement si, pour toute arête cachée e de $\mathcal{R}$, il existe un chemin d'activation contenant e qui soit activé par au moins un des vecteurs de la famille.*

Voir démonstration en annexe.

Remarque 4.18. *Cette condition n'est autre que la condition nécessaire pour la propriété (F) énoncée dans la Proposition 4.12. En particulier, une famille vérifiant la propriété (F) vérifie aussi (R).*

En transposant cette condition au cas avec biais, on obtient une caractérisation semblable, mais qui, contrairement à la précédente, ne peut être vérifiée directement avec les informations dont on dispose, du moins si l'on veut réaliser le redressement avant de reconstruire les biais. Une telle caractérisation est donc peu satisfaisante d'un point de vue algorithmique à ce stade.

Proposition 4.19. *Dans le cas avec biais et sous l'hypothèse (H), une famille d'entrée $(\mathbf{x}_k)_{1\leq k\leq K}$ vérifie la propriété (R) si, et seulement si, pour tout neurone caché $\nu \in \mathcal{V}_j$ de $\mathcal{R}$, il existe un chemin d'activation c contenant ν et un indice $k \in [\![K]\!]$ tels que :*

(i) *$\mathbf{x}_k$ active c,*

(ii) *$|\langle L_\nu(\mathbf{W}_j), y_j(\mathbf{x}_k)\rangle| > |(\mathbf{b}_j)_\nu|$, où y_j est la post-activation des neurones de la j-ème couche.*

Voir démonstration en annexe.

Il reste à reconstruire les biais du réseau inconnu connaissant ses poids.

4.2.3 Reconstruction des biais

Plutôt que de nous concentrer dans la section 4 sur des réseaux butterfly sans biais, nous avons préféré considérer des réseaux généraux, car la plupart des théorèmes qui y sont énoncés sont valables aussi bien dans le cas sans biais que dans le cas avec biais. Cependant, nous n'avons pas eu le temps d'approfondir la question de la reconstruction des biais une fois que les poids ont été reconstruits, question qui reste donc ouverte ici.

En guise de synthèse, nous énonçons dans la sous-section suivante une caractérisation des familles d'entrée permettant une reconstruction forte dans le cas où l'on impose que le réseau inconnu soit sans biais.

4.2.4 Caractérisation des familles fortement reconstruisantes dans le cas butterfly sans biais

Les résultats énoncés dans les sous-sections précédentes fournissent une caractérisation des familles fortement reconstruisantes dans le cas butterfly sans biais.

Proposition 4.20. *On suppose que l'hypothèse (H) est satisfaite, et que l'on sait que le réseau à reconstruire $\mathcal{R}$ est sans biais. Soit $\mathcal{X} = (\mathbf{x}_k)_{1\leq k\leq K}$ une famille d'entrée. Notons $\mathbf{P}'$ la sous-matrice de $\mathbf{P}$ constituée des lignes activées par $\mathcal{X}$, où $\mathbf{P}$ est définie dans le Corollaire 4.8, et $\mathbf{Q}' := \begin{pmatrix} \mathbf{P}' \\ \mathbf{B}^\top \end{pmatrix}$, avec $\mathbf{B}$ une base de $\ker \mathbf{P}$. Alors la donnée de $(J_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ et $(R_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ permet une reconstruction forte de $\mathcal{R}$ si, et seulement si, les colonnes de $\mathbf{Q}'$ sont linéairement indépendantes.*

Voir démonstration en annexe.

4.3 Bornes sur le cardinal des familles fortement reconstruisantes minimales

L'objet de cette sous-section est de borner le cardinal des familles fortement reconstruisantes minimales, où la borne porte sur l'ensemble des réseaux butterfly à nombre de couches $J+1 \geq 3$ fixé vérifiant (H). Nous traitons uniquement le cas sans biais, seul cas pour lequel nous disposons d'une caractérisation des familles fortement reconstruisantes.

Il convient cependant de commencer par remarquer que tous les réseaux vérifiant l'hypothèse (H) ne peuvent faire l'objet d'une reconstruction forte.

Proposition/Définition 4.21 (Réseaux fortement reconstructibles)**.** *Soit $\mathcal{R}$ un réseau butterfly sans biais de matrices de poids $\mathbf{W}_1, \ldots, \mathbf{W}_J$, vérifiant l'hypothèse (H). Les propriétés suivantes sont équivalentes.*

(i) *$\mathcal{R}$ admet au moins une famille fortement reconstruisante,*

(ii) *tous les neurones de $\mathcal{R}$ sont activables,*

(iii) *pour tout neurone caché dans $\cup_{j=2}^{J-1}\mathcal{V}_j$, les poids des deux arêtes entrantes associées ne sont pas tous deux négatifs.*

Un réseau vérifiant l'une de ces trois propriétés équivalentes est dit fortement reconstructible.

Voir démonstration en annexe.

La démonstration de $(ii) \Rightarrow (iii)$ nous a permis d'obtenir le résultat suivant, qu'il convient de mettre en exergue.

Lemme 4.22. *Soit $\mathcal{R}$ un réseau butterfly sans biais vérifiant l'hypothèse (H), et supposé fortement reconstructible, au sens de la Proposition/Définition 4.21. Alors pour tout neurone de sortie ν, il existe un vecteur d'entrée $\mathbf{x}_\nu$ activant tous les neurones de $\mathcal{R}$ contenus dans l'arbre associé à ν (voir Figure 6).*

Dans le cas sans biais, il s'ensuit immédiatement de la caractérisation énoncée dans la Proposition 4.20 qu'il existe des réseaux butterfly à $J+1$ couches – vérifiant (H) – dont le cardinal fortement reconstruisant est égal à 1. C'est par exemple le cas des réseaux à coefficients tous strictement positifs au sein des supports butterfly. En effet, dans ce cas, le vecteur d'entrée $(1, \ldots, 1) \in \mathbb{R}^N$ active tous les chemins complets de $\mathcal{R}$, et donc $\mathbf{P}' = \mathbf{P}$, ce qui montre que $\{(1, \ldots, 1)\}$ est une famille fortement reconstruisante. Nous avons ensuite un encadrement –

malheureusement peu satisfaisant car trop peu contraignant – du plus grand cardinal fortement reconstruisant pour les réseaux butterfly fortement reconstructibles de type (H) à $J+1$ couches, noté $C_{\max}$.

Définition 4.23 ($C_{\max}$)**.** *Soit $J \geq 2$ et $N = 2^J$. On note $C_{\max}$ le plus grand cardinal reconstruisant d'un réseau butterfly à $J+1$ couches fortement reconstructible et vérifiant l'hypothèse (H).*

La proposition suivante établit un majorant de $C_{\max}$.

Proposition/Définition 4.24 (Majoration de $C_{\max}$)**.** *On a :*

$$C_{\max} \leq N. \tag{22}$$

Voir démonstration en annexe.

On peut ensuite déterminer un minorant de $C_{\max}$ en s'appuyant sur la Conjecture 4.16.

Proposition 4.25 (Minoration de $C_{\max}$)**.** *Si la Conjecture 4.16 est vraie, alors on a :*

$$2(J-1) \leq C_{\max}. \tag{23}$$

Voir démonstration en annexe.

Références

[1] James W Cooley et John W Tukey. « An algorithm for the machine calculation of complex Fourier series ». In : *Mathematics of computation* 19.90 (1965), p. 297-301.

[2] Xavier Glorot, Antoine Bordes et Yoshua Bengio. « Deep Sparse Rectifier Neural Networks ». In : *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. T. 15. Proceedings of Machine Learning Research. PMLR, nov. 2011, p. 315-323.

[3] Alex Krizhevsky, Ilya Sutskever et Geoffrey E. Hinton. « ImageNet Classification with Deep Convolutional Neural Networks ». In : *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. 2012, p. 1097-1105.

[4] Quoc-Tung Le et al. « Fast learning of fast transforms, with guarantees ». In : *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, p. 3348-3352.

[5] David Rolnick et Konrad Kording. « Reverse-engineering deep relu networks ». In : *International Conference on Machine Learning*. PMLR. 2020, p. 8178-8187.

[6] Pierre Stock et Rémi Gribonval. « An embedding of ReLU networks and an analysis of their identifiability ». In : *arXiv preprint arXiv :2107.09370* (2021).

[7] Léon Zheng, Elisa Riccietti et Rémi Gribonval. « Efficient Identification of Butterfly Sparse Matrix Factorizations ». In : *arXiv preprint arXiv :2110.01230* (2021).

[8] Léon Zheng, Elisa Riccietti et Rémi Gribonval. « Hierarchical Identifiability in Multilayer Sparse Matrix Factorization ». In : *arXiv preprint arXiv :2110.01230* (2021).

A Démonstrations

Démonstration de la Proposition 3.3. Commençons par le sens réciproque. Soit donc $\mathcal{X} = (\mathbf{x}_k)_k$ une famille d'entrée satisfaisant aux conditions (i) et (ii). Pour tout $i \in [\![N]\!]$, on peut calculer $C_i(\mathbf{W}_2)L_i(\mathbf{W}_1)^\top$, car cette matrice est égale à $J_{\boldsymbol{\theta}}(\mathbf{x}_k) \odot \boldsymbol{\Sigma}_i$ pour $\mathbf{x}_k$ un vecteur d'entrée tel que cette matrice est non nulle (c'est-à-dire activant i), qui existe d'après (i). On remonte ainsi à $\mathbf{W}_2\mathbf{W}_1 = \sum_{i=1}^N C_i(\mathbf{W}_2)L_i(\mathbf{W}_1)^\top$. À partir de ce produit, la Proposition 2.13 nous donne $\mathbf{W}_1'', \mathbf{W}_2''$ deux matrices équivalentes à $\mathbf{W}_1, \mathbf{W}_2$ modulo changements d'échelle inversibles, grâce à l'hypothèse (H_1). On dispose donc de réels non nuls $\lambda_1, \dots, \lambda_N$ tels que $\mathbf{W}_1'' = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{pmatrix} \mathbf{W}_1$ et $\mathbf{W}_2'' = \mathbf{W}_2 \begin{pmatrix} \lambda_1^{-1} & & \\ & \ddots & \\ & & \lambda_N^{-1} \end{pmatrix}$. Notons $\mathbf{W}_1' = \begin{pmatrix} |\lambda_1| & & \\ & \ddots & \\ & & |\lambda_N| \end{pmatrix} \mathbf{W}_1$ et $\mathbf{W}_2' = \mathbf{W}_2 \begin{pmatrix} |\lambda_1^{-1}| & & \\ & \ddots & \\ & & |\lambda_N^{-1}| \end{pmatrix}$, puis $\mathbf{b}'$ et $\mathbf{c}'$ les vecteurs de biais associés aux poids $\mathbf{W}_1'$ et $\mathbf{W}_2'$ de sorte à former des paramètres $\boldsymbol{\theta}'$ tels que $R_{\boldsymbol{\theta}'} = R_{\boldsymbol{\theta}}$. Ce sont les poids $\mathbf{W}_1'$ et $\mathbf{W}_2'$, équivalents à $\mathbf{W}_1$ et $\mathbf{W}_2$ modulo changements d'échelle strictement positifs, que l'on va reconstruire. Pour tout $i \in [\![N]\!]$, choisissons $\mathbf{x}_k$ tel que $\langle L_i(\mathbf{W}_1''), \mathbf{x}_k\rangle$ est maximal en valeur absolue. Alors, par hypothèse, $|\langle L_i(\mathbf{W}_1'), \mathbf{x}_k\rangle| \geq |b_i'|$. L'idée de la démonstration est que le signe de la somme $\langle L_i(\mathbf{W}_1'), \mathbf{x}_k\rangle + b_i'$, qui détermine l'état d'activation du neurone i en $\mathbf{x}_k$, est donné par le signe de $\langle L_i(\mathbf{W}_1'), \mathbf{x}_k\rangle$. Montrons donc que λ_i est strictement négatif si, et seulement si, l'on est dans l'une des deux situations suivantes :

— $\mathbf{x}_k$ active le neurone caché i et $\langle L_i(\mathbf{W}_1''), \mathbf{x}_k\rangle \leq 0$,

— $\mathbf{x}_k$ n'active pas le neurone caché i et $\langle L_i(\mathbf{W}_1''), \mathbf{x}_k\rangle > 0$.

Supposons que $\mathbf{x}_k$ active le neurone caché i. Alors $\langle L_i(\mathbf{W}_1'), \mathbf{x}_k\rangle + b_i' > 0$, donc, par (ii), le produit scalaire $\langle L_i(\mathbf{W}_1'), \mathbf{x}_k\rangle$ est nécessairement strictement positif. En effet, si $\langle L_i(\mathbf{W}_1'), \mathbf{x}_k\rangle \leq 0$, on obtient que $\langle L_i(\mathbf{W}_1'), \mathbf{x}_k\rangle + b_i' \leq -|\langle L_i(\mathbf{W}_1'), \mathbf{x}_k\rangle| + |b_i'| \leq 0$ ce qui est absurde. Un raisonnement analogue dans le cas où $\mathbf{x}_k$ n'active pas le neurone caché i achève la démonstration de l'équivalence. Ainsi, on peut retrouver $\mathbf{W}_1'$ et $\mathbf{W}_2'$ en changeant les signes des λ_i pour lesquels i vérifie l'un des deux tirets ci-dessus. Nous appelons *redressement* cette procédure. On obtient ainsi des matrices équivalentes à $\mathbf{W}_1$ et $\mathbf{W}_2$ modulo changements d'échelles strictement positifs, ce qui achève la reconstruction forte.

Montrons le sens direct par contraposée. Soit $\mathcal{X} = (\mathbf{x}_k)_k$ une famille d'entrée. Si $\mathcal{X}$ ne vérifie pas le point (i) de la proposition, soit $i \in [\![N]\!]$ un neurone caché qu'aucun des $\mathbf{x}_k$ n'active. Les familles $(J_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ et $(R_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ ne contiennent aucune information sur les coefficients de $L_i(\mathbf{W}_1)$, d'après le Lemme 3.2, donc toute modification de $L_i(\mathbf{W}_1)$ dans le réseau inconnu conduit aux mêmes observations de réalisation et de jacobienne. On ne peut donc parvenir à une reconstruction forte des poids. Si $\mathcal{X}$ ne vérifie pas la propriété (ii), on dispose de $i \in [\![N]\!]$ tel que pour tout $k \in [\![K]\!]$, on ait $|\langle L_i(\mathbf{W}_1), \mathbf{x}_k\rangle| < |b_i|$. L'état d'activation du neurone i est alors le même en chaque $\mathbf{x}_k$. Puisque (i) est nécessaire, le biais b_i est strictement positif (sinon aucun des $\mathbf{x}_k$ n'activerait le neurone i). Mais alors le réseau obtenu à partir de $\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}, \mathbf{c}$ en multipliant $L_i(\mathbf{W}_1)$ et $C_i(\mathbf{W}_2)$ par -1 et en remplaçant $\mathbf{c}$ par $\mathbf{c} + 2C_i(\mathbf{W}_2)$ conduit aux mêmes observations de réalisation et de jacobienne en les $\mathbf{x}_k$, notamment parce que changer le signe de $L_i(\mathbf{W}_2)$ ne modifie pas l'état d'activation du i-ème neurone. Il est donc impossible de reconstruire des matrices de poids équivalentes à $\mathbf{W}_1, \mathbf{W}_2$ modulo changements d'échelle strictement positifs. □

Démonstration de la Proposition 3.10. Notons $\mathcal{H}_i$ l'hyperplan associé au i-ème neurone caché. On a :

$$\mathcal{H}_i = \left\{ \mathbf{z} \in \mathbb{R}^d \ ; \ \langle L_i(\mathbf{W}_1), \mathbf{z}\rangle = -b_i \right\}. \tag{24}$$

Soient $k, \ell \in [\![K]\!]$ tels que $\mathbf{x}_k$ et $\mathbf{x}_\ell$ sont i-voisins. On peut supposer que c'est $\mathbf{x}_k$ qui active le neurone i. D'après la Remarque 3.9, en notant respectivement A_k et A_ℓ les applications affines avec lesquelles $R_{\boldsymbol{\theta}}$ coïncide sur les zones affines contenant $\mathbf{x}_k$ et $\mathbf{x}_\ell$, l'hyperplan $\mathcal{H}_i$ est donné par $\{\mathbf{z} \in \mathbb{R}^d \ ; \ A_k(\mathbf{z}) = A_\ell(\mathbf{z})\}$. D'où :

$$\begin{aligned}
\mathcal{H}_i &= \left\{\mathbf{z} \in \mathbb{R}^d \ ; J_{\boldsymbol{\theta}}(\mathbf{x}_k)(\mathbf{z} - \mathbf{x}_k) + R_{\boldsymbol{\theta}}(\mathbf{x}_k) = J_{\boldsymbol{\theta}}(\mathbf{x}_\ell)(\mathbf{z} - \mathbf{x}_\ell) + R_{\boldsymbol{\theta}}(\mathbf{x}_\ell)\right\} && (25)\\
&= \left\{\mathbf{z} \in \mathbb{R}^d \ ; \ (J_{\boldsymbol{\theta}}(\mathbf{x}_k) - J_{\boldsymbol{\theta}}(\mathbf{x}_\ell))\mathbf{z} = J_{\boldsymbol{\theta}}(\mathbf{x}_k)\mathbf{x}_k - J_{\boldsymbol{\theta}}(\mathbf{x}_\ell)\mathbf{x}_\ell + R_{\boldsymbol{\theta}}(\mathbf{x}_\ell) - R_{\boldsymbol{\theta}}(\mathbf{x}_k)\right\} && (26)\\
&= \left\{\mathbf{z} \in \mathbb{R}^d \ ; \ \langle L_i(\mathbf{W}_1'), \mathbf{z}\rangle C_i(\mathbf{W}_2') = J_{\boldsymbol{\theta}}(\mathbf{x}_k)\mathbf{x}_k - J_{\boldsymbol{\theta}}(\mathbf{x}_\ell)\mathbf{x}_\ell + R_{\boldsymbol{\theta}}(\mathbf{x}_\ell) - R_{\boldsymbol{\theta}}(\mathbf{x}_k)\right\}, && (27)
\end{aligned}$$

la dernière égalité provenant du Lemme 3.8, où le signe ε est connu puisque l'on a supposé que $\mathbf{x}_k$ active i. Ainsi, en notant $\mathbf{X} = J_{\boldsymbol{\theta}}(\mathbf{x}_k)\mathbf{x}_k - J_{\boldsymbol{\theta}}(\mathbf{x}_\ell)\mathbf{x}_\ell + R_{\boldsymbol{\theta}}(\mathbf{x}_\ell) - R_{\boldsymbol{\theta}}(\mathbf{x}_k)$, les vecteurs $C_i(\mathbf{W}_2')$ et $\mathbf{X}$ sont colinéaires, et l'on peut remonter à $b_i' = -\langle L_i(\mathbf{W}_1'), \mathbf{z}\rangle$ pour $\mathbf{z} \in \mathcal{H}_i$ (d'après l'équation 24) par exemple en choisissant $j \in [\![s]\!]$ tel que $(C_i(\mathbf{W}_2'))_j \neq 0$, ce qui est toujours possible car $\mathbf{W}_2'$ n'a pas de colonne nulle par hypothèse, et en calculant $b_i' = -\frac{X_j}{(C_i(\mathbf{W}_2'))_j}$. On retrouve ainsi le i-ème biais caché à partir de $J_{\boldsymbol{\theta}}(\mathbf{x}_k)$, $J_{\boldsymbol{\theta}}(\mathbf{x}_\ell)$, $R_{\boldsymbol{\theta}}(\mathbf{x}_k)$ et $R_{\boldsymbol{\theta}}(\mathbf{x}_\ell)$. Pour reconstruire ensuite le vecteur de biais de sortie $\mathbf{c}$, il suffit de calculer $R_{\boldsymbol{\theta}}(\mathbf{x}_k) - \mathbf{W}_2' \,\mathtt{ReLU}(\mathbf{W}_1'\mathbf{x}_k + \mathbf{b}') = \mathbf{c}$ pour un élément quelconque $\mathbf{x}_k$ de la famille d'entrée considérée. □

Démonstration de la Proposition 3.13. Notons $\mathbf{b}'$ et $\mathbf{c}'$ les vecteurs de biais associés à $\mathbf{W}_1', \mathbf{W}_2'$ de sorte à obtenir $R_{\boldsymbol{\theta}}$. Commençons par remarquer que les colonnes de $\mathbf{W}_2'$ sont linéairement indépendantes, car équivalentes modulo changements d'échelle à celles de $\mathbf{W}_2$, qui le sont par (H_3).

Supposons que $\mathcal{X}$ active tous les neurones cachés de $\mathcal{R}$, mais qu'aucun neurone caché n'est activé par tous les éléments de $\mathcal{X}$, et montrons que cette famille permet de retrouver $\mathbf{b}'$ et $\mathbf{c}'$. Fixons $i \in [\![N]\!]$, et $k, \ell \in [\![K]\!]$ tels que $\mathbf{x}_k$ active le neurone i, et $\mathbf{x}_\ell$ ne l'active pas. On a, pour tout vecteur d'entrée $\mathbf{x}$:

$$R_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{c} + \sum_{r=1}^{N} \mathbf{1}_{\langle L_r(\mathbf{W}_1'), \mathbf{x}\rangle + b_r' > 0} \ C_r(\mathbf{W}_2')(L_r(\mathbf{W}_1')^\top \mathbf{x} + b_r'), \tag{28}$$

d'où

$$R_{\boldsymbol{\theta}}(\mathbf{x}_k) - R_{\boldsymbol{\theta}}(\mathbf{x}_\ell) = C_i(\mathbf{W}_2')(L_i(\mathbf{W}_1')^\top \mathbf{x}_k + b_i') + \mathbf{z}_i, \tag{29}$$

pour un certain $\mathbf{z}_i \in \bigoplus_{r \neq i} \mathbb{R}\, C_r(\mathbf{W}_2')$. Par liberté des colonnes de $\mathbf{W}_2'$, on dispose d'un unique jeu de coefficients $\alpha_1, \ldots, \alpha_N \in \mathbb{R}$ tels que : $R_{\boldsymbol{\theta}}(\mathbf{x}_k) - R_{\boldsymbol{\theta}}(\mathbf{x}_\ell) = \sum_{r=1}^{N} \alpha_r C_r(\mathbf{W}_2')$. On identifie alors sans ambiguité $b_i' = \alpha_i - L_i(\mathbf{W}_1')^\top \mathbf{x}_k$. Une fois que l'on a reconstruit tous les biais de la couche cachée, on reconstruit $\mathbf{c}$ en calculant $R_{\boldsymbol{\theta}}(\mathbf{x}) - \mathbf{W}_2' \,\mathtt{ReLU}(\mathbf{W}_1'\mathbf{x} + \mathbf{b}') = \mathbf{c}$ pour un vecteur $\mathbf{x} \in \mathcal{X}$ choisi arbitrairement.

Réciproquement, montrons tout d'abord qu'il est nécessaire pour la reconstruction que la famille $\mathcal{X}$ active tous les neurones cachés. Si $\mathcal{X}$ n'active pas le neurone i, alors tout biais b_i'' vérifiant

$$\forall k \in [\![K]\!], \langle L_i(\mathbf{W}_1'), \mathbf{x}_k\rangle + b_i'' \leq 0 \tag{30}$$

est compatible avec les observations $(R_{\boldsymbol{\theta}}(\mathbf{x}_k))_k$ et $(J_{\boldsymbol{\theta}}(\mathbf{x}_k))_k$. Il est donc impossible de déterminer b_i'. Supposons ensuite qu'il existe un neurone caché i tel que tous les vecteurs de $\mathcal{X}$ activent i. Alors le réseau construit à partir de $\mathcal{R}$ pour lequel on remplace b_i par $\tilde{b}_i := b_i + \varepsilon$ avec $\varepsilon > 0$, de sorte qu'aucun des états d'activation du neurone i en les $\mathbf{x}_k$ ne change de b_i à $\tilde{b}_i$, et $\mathbf{c}$ par $\tilde{\mathbf{c}} := \mathbf{c} - \varepsilon C_i(\mathbf{W}_2)$, induit les mêmes observations de la réalisation et sa jacobienne en les éléments de $\mathcal{X}$. On ne peut donc espérer reconstruire $\mathbf{b}'$ et $\mathbf{c}'$. □

Démonstration de la Proposition 3.14. Tous les ingrédients nécessaires à la démonstration de la Proposition 3.14 ont déjà été énoncés : on a vu dans les propositions 3.3 et 3.13 que les conditions (i) et (ii) sont nécessaires à la reconstruction forte, et ces deux conditions réunies sont suffisantes d'après la démonstration de ces mêmes propositions. □

Démonstration de la Proposition 3.15. Soit $\mathcal{R}$ un réseau de la forme étudiée. On dispose de $\mathbf{x} \in \mathbb{R}^d$ tel que pour tout neurone caché i, on ait $|\langle L_i(\mathbf{W}_1), \mathbf{x}\rangle| > |b_i|$. On vérifie alors aisément avec la caractérisation de la Proposition 3.14 que $(\mathbf{x}, -\mathbf{x})$ est une famille fortement reconstruisante pour $\mathcal{R}$. De plus, on voit qu'un seul vecteur d'entrée ne peut constituer une famille fortement reconstruisante pour un réseau vérifiant les hypothèses (H_1) et (H_3), puisque la condition (i) de la Proposition 3.14 n'est pas vérifiée dans ce cas-là. □

Démonstration de la Proposition 3.18. En effet, une telle famille permet de connaître la valeur exacte du produit $\mathbf{W}_2\mathbf{W}_1$, suivant la même méthode que pour montrer la Proposition 3.3. La factorisation donne ensuite $(\mathbf{W}_1'', \mathbf{W}_2'')$ telles que $L_i(\mathbf{W}_1'')$ et $C_i(\mathbf{W}_2'')$ sont nuls pour tout neurone i associé à une composante de rang 1 nulle, et sont équivalents à $L_i(\mathbf{W}_1)$ et $C_i(\mathbf{W}_2)$ modulo changements d'échelle inversibles pour les autres i, ce qui, après redressement des lignes et colonnes associées aux autres neurones, fournit des matrices de poids conduisant à la même réalisation que $\mathbf{W}_1$ et $\mathbf{W}_2$. □

Démonstration du Lemme 3.21. En effet, soit $i \in [\![N]\!]$ un neurone non dégénéré. Puisque l'on a $\|L_i(\mathbf{W}_1)\|^2 = \sum_{\ell=1}^d \langle L_i(\mathbf{W}_1), \mathbf{e}_\ell\rangle^2$, on dispose de $\ell \in [\![d]\!]$ tel que $\langle L_i(\mathbf{W}_1), \mathbf{e}_\ell\rangle^2 \geq \frac{1}{d}\|L_i(\mathbf{W}_1)\|^2$. Pour un tel ℓ :

$$\langle L_i(\mathbf{W}_1), \mathbf{e}_\ell\rangle^2 \geq \frac{1}{d}\|L_i(\mathbf{W}_1)\|^2 > \frac{1}{d\alpha^2} b_i^2, \tag{31}$$

donc $|\langle L_i(\mathbf{W}_1), \alpha\sqrt{d}\,\mathbf{e}_\ell\,\rangle| > |b_i|$ et exactement un des deux vecteurs $\alpha\sqrt{d}\,\mathbf{e}_\ell$ et $-\alpha\sqrt{d}\,\mathbf{e}_\ell$ active le neurone i, ce qui permet la reconstruction faible des poids par un raisonnement analogue à celui de la sous-section 3.3.1. □

Démonstration du Lemme 3.22. En effet, nous avons :

$$R_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{c} + \sum_{i=1}^N \mathbf{1}_{\langle L_i(\mathbf{W}_1), \mathbf{x}\rangle + b_i > 0}\; C_i(\mathbf{W}_2)(L_i(\mathbf{W}_1)^\top \mathbf{x} + b_i), \tag{32}$$

et

$$R_{\boldsymbol{\theta}'}(\mathbf{x}) = \mathbf{c} + \sum_{i\in\mathcal{I}} \mathbf{1}_{\langle L_i(\mathbf{W}_1), \mathbf{x}\rangle + b_i > 0}\; C_i(\mathbf{W}_2)(L_i(\mathbf{W}_1)^\top \mathbf{x} + b_i), \tag{33}$$

où $\mathcal{I}$ est l'ensemble des neurones cachés activés par l'un des $\mathbf{x}_k$. Or, si un neurone caché i n'est activé par aucun des $\mathbf{x}_k$, on a :

$$\forall k \in [\![K]\!],\ \langle L_i(\mathbf{W}_1), \mathbf{x}_k\rangle + b_i \leq 0, \tag{34}$$

donc si $\mathbf{x} = \lambda_1\mathbf{x}_1 + \cdots + \lambda_K\mathbf{x}_K$ avec $\lambda_1, \ldots, \lambda_K \geq 0$ tels que $\sum_k \lambda_k = 1$, on a encore $\langle L_i(\mathbf{W}_1), \mathbf{x}\rangle + b_i \leq 0$, et $\mathbf{x}$ n'active pas i. Ainsi, sur $\mathcal{E}$, les applications $R_{\boldsymbol{\theta}}$ et $R_{\boldsymbol{\theta}'}$ coïncident. □

Démonstration du Lemme 4.5. Par définition de $\mathbf{S}_j$, on a pour tout $j \in [\![J]\!]$:

$$\mathbf{S}_j = \mathbf{I}_{2^{J-j}} \otimes \mathbf{1}_2 \otimes \mathbf{I}_{2^{j-1}}. \tag{35}$$

En utilisant que $(\mathbf{A}\otimes\mathbf{B})(\mathbf{C}\otimes\mathbf{D}) = (\mathbf{AC})\otimes(\mathbf{BD})$ quand les formats de matrices donnent un sens à l'expression, ainsi que l'associativité de $\otimes$, on montre par récurrence descendante sur j que pour tout $j \in [\![J]\!]$:

$$\mathbf{S}_J \ldots \mathbf{S}_j = \mathbf{1}_2^{\otimes(J-j+1)} \otimes \mathbf{I}_{2^{j-1}}. \tag{36}$$

En effet, cette égalité est vraie pour $j = J$, et si elle est vraie pour $j+1 \in [\![J]\!]$, alors

$$\begin{aligned}
\mathbf{S}_J \dots \mathbf{S}_j &= \left(\mathbf{1}_2^{\otimes(J-j)} \otimes \mathbf{I}_{2^j}\right)\left(\mathbf{I}_{2^{J-j}} \otimes \mathbf{1}_2 \otimes \mathbf{I}_{2^{j-1}}\right) && (37)\\
&= \left(\mathbf{1}_2^{\otimes(J-j)} \mathbf{I}_{2^{J-j}}\right) \otimes \left(\mathbf{I}_{2^j}(\mathbf{1}_2 \otimes \mathbf{I}_{2^{j-1}})\right) && (38)\\
&= \mathbf{1}_2^{\otimes(J-j)} \otimes \mathbf{1}_2 \otimes \mathbf{I}_{2^{j-1}} && (39)\\
&= \mathbf{1}_2^{\otimes(J-j+1)} \otimes \mathbf{I}_{2^{j-1}}, && (40)\\
& && (41)
\end{aligned}$$

ce qui conclut la récurrence. En prenant $j = 1$ on obtient :

$$\mathbf{S}_J^J \dots \mathbf{S}_1^J = \mathbf{1}_2^{\otimes J} = \mathbf{1}_{2^J}. \tag{42}$$

□

Démonstration du Lemme 4.9. En effet, les seules modifications des coefficients des facteurs qui laissent invariant le produit sont les changements d'échelle inversibles au niveau de chaque neurone caché, et ceux-ci peuvent se faire de façon indépendante entre neurones. Ainsi, on a la base suivante pour $\ker \mathbf{P}$:

$$\ker \mathbf{P} = \bigoplus_{\nu \in H} \mathbb{R}(\mathbf{e}_{i_{\nu,1}} + \mathbf{e}_{i_{\nu,2}} - \mathbf{e}_{j_{\nu,1}} - \mathbf{e}_{j_{\nu,2}}), \tag{43}$$

où $\mathbf{e}$ est la base canonique de $\mathbb{R}^{2NJ}$, les entiers $i_{\nu,1}$ et $i_{\nu,2}$ désignent les indices des deux arêtes entrant dans ν, et $j_{\nu,1}$ et $j_{\nu,2}$ ceux des arêtes sortant de ν. □

Démonstration de la Proposition 4.12. Supposons que $(\mathbf{x}_k)_{1\leq k\leq K}$ soit tel qu'il existe une arête de $\mathcal{R}$ n'appartenant à aucun des chemins d'activation activés par les $\mathbf{x}_k$. Soit e une telle arête. Alors la contribution de e aux familles $(J_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ et $(R_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$ est nulle (d'après le Lemme 4.4), c'est-à-dire que tout réseau construit à partir de $\mathcal{R}$ en modifiant légèrement le poids de e, assez peu pour ne pas changer les états d'activation de neurones en les $\mathbf{x}_k$, induit les mêmes observations de réalisation et de jacobienne en ces points. Il est donc impossible de reconstruire $\mathbf{W}_1, \dots, \mathbf{W}_J$ modulo changements d'échelle inversibles. □

Démonstration de la Proposition 4.13. Trouver $\mathbf{W}''_1, \dots, \mathbf{W}''_J$ telles qu'il existe $\mathbf{D}_1, \dots, \mathbf{D}_{J-1}$ des matrices diagonales inversibles vérifiant : $\forall j \in [\![J]\!], \mathbf{W}''_j = \mathbf{D}_j \mathbf{W}_j \mathbf{D}_{j-1}^{-1}$, avec $\mathbf{D}_0 = \mathbf{D}_J = \mathbf{I}_N$, est équivalent à connaître $\mathbf{W}_J \dots \mathbf{W}_1$ le produit des matrices de poids de $\mathcal{R}$, du fait de l'existence de l'algorithme de factorisation butterfly présenté dans la partie 2.3. Or si $\mathbf{x}$ et $i, j \in [\![N]\!]$ sont tels que $(J_{\boldsymbol{\theta}}(\mathbf{x}))_{i,j} \neq 0$, alors $(\mathbf{W}_J \dots \mathbf{W}_1)_{i,j} = (J_{\boldsymbol{\theta}}(\mathbf{x}))_{i,j}$, d'après le Lemme 4.5, ce qui permet de conclure. □

Démonstration de la Proposition 4.14. On note $\mathbf{W}_1, \dots, \mathbf{W}_J$ les matrices de poids du réseau à reconstruire.

i) $\Rightarrow$ *ii)*. Si $\mathbf{Q}' = \begin{pmatrix} \mathbf{P}' \\ \mathbf{B}^\top \end{pmatrix}$ n'est pas injective, alors il existe une famille de poids $\mathbf{X} = (\mathbf{X}_1, \dots, \mathbf{X}_J)$ différente de $\mathbf{W} = (\mathbf{W}_1, \dots, \mathbf{W}_J)$ telle que $\log(\mathrm{vec}(\mathbf{X}))$ et $\log(\mathrm{vec}(\mathbf{W}))$ ont la même image par $\mathbf{Q}'$ et induisent les mêmes états d'activation de chemins en chaque $\mathbf{x}_k$. En particulier, les produits $\mathbf{X}_J \dots \mathbf{X}_1$ et $\mathbf{W}_J \dots \mathbf{W}_1$ ont les mêmes coefficients aux emplacements (i, j) correspondant aux lignes de $\mathbf{P}'$. Puisque la famille $\mathcal{X}$ ne donne aucune information sur les autres coefficients de ces produits, $\mathbf{X}$ et $\mathbf{W}$ induisent les mêmes observations de réalisation et de jacobienne en chaque point de $\mathcal{X}$. En outre, par définition de $\mathbf{B}$, les familles $\mathbf{X}$ et $\mathbf{W}$ ne

sont pas équivalentes modulo changements d'échelle inversibles, car si c'était le cas, on aurait $\log(\mathrm{vec}(\mathbf{X})) - \log(\mathrm{vec}(\mathbf{W})) \in \operatorname{Im}\mathbf{B}$, ce qui contredit le fait que $\log(\mathrm{vec}(\mathbf{X}))$ et $\log(\mathrm{vec}(\mathbf{W}))$ ont la même image par $\mathbf{B}^\top$. On a ainsi trouvé deux familles de paramètres non équivalentes modulo changements d'échelle inversibles, mais induisant les mêmes observations de réalisation et de jacobienne en chaque point de $\mathcal{X}$. Cette famille d'entrée ne vérifie donc pas (F).

ii) $\Rightarrow$ *i)*. Rappelons que l'on note $\mathbf{W}_1, \dots, \mathbf{W}_J$ les matrices de poids du réseau à reconstruire. Soit $\mathbf{A} = (a_{i,j})_{i,j}$ la matrice définie par $a_{i,j} = (\mathbf{W}_J \dots \mathbf{W}_1)_{i,j}$ s'il existe $k \in [\![K]\!]$ tel que $(J_{\boldsymbol{\theta}}(\mathbf{x}_k))_{i,j} \neq 0$ (on a vu que dans ce cas $(J_{\boldsymbol{\theta}}(\mathbf{x}_k))_{i,j} = (\mathbf{W}_J \dots \mathbf{W}_1)_{i,j}$), et $a_{i,j} = 0$ sinon. On peut construire cette matrice uniquement à partir de la donnée de $(J_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$. Notons $\tilde{\mathbf{a}}$ le sous-vecteur de $\mathrm{vec}(\mathbf{A})$ où l'on n'a gardé que les éléments non nuls. Soit ensuite $\mathbf{a}' = \begin{pmatrix}\tilde{\mathbf{a}}\\ \mathbf{1}\end{pmatrix}$, où $\mathbf{1} \in \mathbb{R}^{N(J-1)}$ a tous ses coefficients égaux à 1. On a donc $\mathbf{a}' \in \mathbb{R}^{\sharp(\text{coefficients non nuls de } \mathbf{A}) + N(J-1)}$. Alors le système $\log(\mathbf{a}') = \mathbf{Q}'\,\mathrm{vec}(\log \mathbf{X})$ d'inconnue $\mathbf{X}$ admet au plus une solution, par injectivité de $\mathbf{Q}'$ et du log. D'autre part, il admet au moins une solution. En effet, notons $\mathbf{y} = \mathbf{B}^\top \mathrm{vec}(\log(\mathbf{W}))$. La matrice $\mathbf{B}^\top\mathbf{B}$, carrée de taille $N(J-1)$, est inversible car $\mathbf{B}$ est une base, donc en notant $\mathbf{u} = -(\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{y}$, puis $\mathbf{z} = \mathbf{B}\mathbf{u}$, on a $\mathbf{B}^\top\mathbf{z} = -\mathbf{y}$. Alors $\mathrm{vec}(\log(\mathbf{W})) + \mathbf{z}$ est une solution du système considéré, car $\begin{pmatrix}\mathbf{P}'\\ \mathbf{B}^\top\end{pmatrix}\mathrm{vec}(\log(\mathbf{W})) = \begin{pmatrix}\mathbf{a}'\\ \mathbf{y}\end{pmatrix}$ et $\begin{pmatrix}\mathbf{P}'\\ \mathbf{B}^\top\end{pmatrix}\mathbf{z} = \begin{pmatrix}\mathbf{0}\\ -\mathbf{y}\end{pmatrix}$. C'est cette unique solution que l'on obtient en résolvant le système à l'aide de la pseudo-inverse de $\mathbf{Q}'$, et elle nous permet de construire des matrices de poids équivalentes à celles de $\mathcal{R}$ modulo changements d'échelle inversibles. Ainsi, $\mathcal{X}$ vérifie (F).

ii) $\Leftrightarrow$ *iii)*. La condition *ii)* se réexprime : $\operatorname{rg}\mathbf{Q}' = 2NJ$, ce qui équivaut à $\operatorname{rg}\mathbf{Q}'^\top = 2NJ$. Remarquons ensuite que $\operatorname{Im}\mathbf{P}'^\top \cap \operatorname{Im}\mathbf{B} = \{0\}$. En effet, si $\mathbf{b} \in \operatorname{Im}\mathbf{P}'^\top \cap \operatorname{Im}\mathbf{B}$, on peut écrire $\mathbf{b} = \mathbf{P}'^\top\mathbf{u}$ avec $\mathbf{u} \in \mathbb{R}^{2NJ}$, auquel cas par définition de $\mathbf{b} \in \operatorname{Im}\mathbf{B}$ on a $\mathbf{P}'\mathbf{b} = \mathbf{P}'\mathbf{P}'^\top\mathbf{u} = 0$, mais puisque $\ker\mathbf{P}'\mathbf{P}'^\top = \ker\mathbf{P}'^\top$, on a aussi $\mathbf{P}'^\top\mathbf{u} = 0$ et donc $\mathbf{b} = 0$. Ainsi, puisque $\mathbf{Q}'^\top = \begin{pmatrix}\mathbf{P}'^\top & \mathbf{B}\end{pmatrix}$, on obtient $\operatorname{rg}\mathbf{Q}' = \operatorname{rg}\mathbf{P}'^\top + \operatorname{rg}\mathbf{B}$. La condition $\operatorname{rg}\mathbf{Q}'^\top = 2NJ$ est donc équivalente à $\operatorname{rg}\mathbf{P}'^\top = 2NJ - N(J-1) = N(J+1) = \operatorname{rg}\mathbf{P}$. On conclut puisque $\operatorname{rg}\mathbf{P}'^\top = \operatorname{rg}\mathbf{P}'$. □

Démonstration de la Proposition 4.17. La condition est nécessaire par un argument analogue à celui employé pour montrer la Proposition 4.12. Montrons qu'elle est suffisante en explicitant un algorithme de redressement par récurrence qui exploite les observations $(J_{\boldsymbol{\theta}}(\mathbf{x}_k))_{1\leq k\leq K}$. On commence avec $\mathbf{W}'_1, \dots, \mathbf{W}'_J = \mathbf{W}''_1, \dots, \mathbf{W}''_J$. Pour chaque neurone $\nu \in \mathcal{V}_1$ successivement, on choisit $k \in [\![K]\!]$ tel que $\mathbf{x}_k$ active un chemin d'activation passant par une arête quelconque sortant de ν, ce que l'on peut faire en exploitant la Remarque 4.7 et puisque l'on a supposé que toute arête cachée est contenue dans un chemin activé par au moins l'un des vecteurs de la famille $\mathcal{X}$, et l'on calcule $\langle L_\nu(\mathbf{W}''_1), \mathbf{x}_k\rangle$. Ce produit scalaire est nécessairement non nul, car $\mathbf{x}_k$ active le neurone ν, et $\langle L_\nu(\mathbf{W}''_1), \mathbf{x}_k\rangle$ est égal à la préactivation de ν en $\mathbf{x}_k$ au signe près. Si le produit scalaire obtenu est strictement négatif, on change le signe de $L_\nu(\mathbf{W}'_1)$ et $C_\nu(\mathbf{W}'_2)$, car les bons signes doivent induire une préactivation strictement positive pour ν, donc un produit scalaire $\langle L_\nu(\mathbf{W}'_1), \mathbf{x}_k\rangle$ strictement positif. Ensuite, en supposant que l'on a corrigé les signes des lignes et colonnes associées aux neurones des $j-1$ premières couches cachées, on détermine les signes associés aux neurones de la couche suivante en calculant pour chaque tel neurone ν le produit scalaire (à nouveau non nul) $\big\langle L_\nu(\mathbf{W}''_j), \mathtt{ReLU}(\mathbf{W}'_{j-1} \dots \mathtt{ReLU}(\mathbf{W}'_1\mathbf{x}_k))\big\rangle$, où $\mathbf{x}_k$ active un chemin d'activation passant par une arête cachée quelconque contenant ν : s'il est strictement négatif, on modifie le signe de $L_\nu(\mathbf{W}'_j)$ et $C_\nu(\mathbf{W}'_{j+1})$, suivant le même raisonnement que pour les neurones de la première couche cachée. Ce qui permet de conclure. □

Démonstration de la Proposition 4.19. La démonstration de la Proposition 4.17 se transpose exactement ici, grâce à l'hypothèse (*ii*), sur le modèle de la Proposition 3.3. □

Démonstration de la Proposition 4.20. On a vu dans la Proposition 4.14 que la condition de linéaire indépendance des colonnes de $\mathbf{Q}'$ est nécessaire pour que $\mathcal{X}$ soit reconstruisante. De plus, dans le cas sans biais, cette condition est suffisante d'après la Remarque 4.18. □

Démonstration de la Proposition/Définition 4.21. $(i) \Rightarrow (ii)$. La Proposition 4.12 montre qu'une famille fortement reconstruisante active nécessairement toutes les arêtes de $\mathcal{R}$, donc aussi tous ses neurones.

$(ii) \Rightarrow (iii)$. Supposons par contraposée qu'il existe un neurone caché $\nu \in \mathcal{V}_j$ avec $j \geq 2$ associé à deux arêtes entrantes de poids négatifs w_1 et w_2. Alors pour tout vecteur d'entrée $\mathbf{x}$, on a $(z_j(\mathbf{x}))_\nu = w_1 a + w_2 b$ avec a et b des réels positifs, car coordonnées du vecteur $\texttt{ReLU}\,(z_{j-1}(\mathbf{x}))$. Ainsi, $z_j(\mathbf{x}) \leq 0$, et le neurone ν n'est jamais activé, donc tous les neurones de $\mathcal{R}$ ne sont pas activables.

$(iii) \Rightarrow (i)$. Si ν est un neurone de la couche de sortie, nous appelons arbre associé à ν l'ensemble des neurones reliés à ν dans le graphe de $\mathcal{R}$ (voir Figure 6). Il y a autant d'arbres que de neurones sur la dernière couche, soit N.

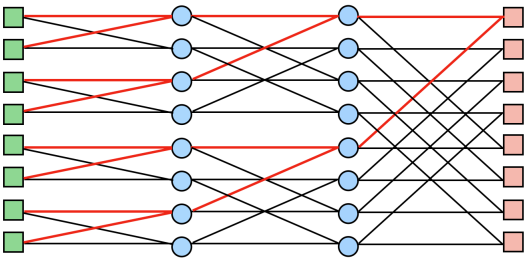


FIGURE 6 – En rouge, l'arbre associé au premier neurone de $\mathcal{V}_3$.

Nous nous appuyons sur la remarque suivante, dont nous donnons la démonstration dans un deuxième temps. Pour tout neurone de sortie ν, il existe un vecteur d'entrée $\mathbf{x}_\nu$ activant tous les neurones de $\mathcal{R}$ contenus dans l'arbre associé à ν. En supposant que cette affirmation est vraie, on peut trouver pour chaque arbre de $\mathcal{R}$ un vecteur activant tous les neurones de l'arbre. L'ensemble des vecteurs d'entrée ainsi obtenus forme une famille fortement reconstruisante de $\mathcal{R}$, d'après la Proposition 4.13, puisque cette famille active tous les chemins complets de $\mathcal{R}$, et que l'on est dans le cas sans biais. Ce qui prouve l'implication étudiée.

Montrons à présent que pour tout neurone de sortie ν, il existe un vecteur d'entrée $\mathbf{x}_\nu$ activant tous les neurones de $\mathcal{R}$ contenus dans l'arbre associé à ν. Notons A l'ensemble des neurones contenus dans l'arbre associé à ν, et supposons sans perte de généralité que $\nu = 1$. Montrons par récurrence que pour tout indice de couche cachée $j \in [\![J-1]\!]$, il existe un vecteur d'entrée $\mathbf{x}_j$ qui active tous les neurones de $\left(\cup_{\ell=0}^{j} \mathcal{V}_\ell\right) \cap A$. C'est vrai pour $j = 1$. En effet, il s'agit d'activer simultanément tous les neurones d'indice impair de la première couche cachée. Or, le vecteur $\mathbf{x}_1 \in \mathbb{R}^N$ défini par

$$(\mathbf{x}_1)_n = \begin{cases} w_{(n \in \mathcal{V}_0 \to n \in \mathcal{V}_1)} \text{ si } n \text{ est impair,} \\ w_{(n \in \mathcal{V}_0 \to n-1 \in \mathcal{V}_1)} \text{ si } n \text{ est pair} \end{cases} \tag{44}$$

pour $n \in [\![N]\!]$ induit des préactivations strictement positives pour tous les neurones d'indice impair de la première couche cachée, donc convient. Supposons ensuite que le résultat est vrai pour un certain $j \in [\![J-2]\!]$, et montrons qu'il est vrai au rang $j+1$. Soit $\mathbf{x}_j$ activant les neurones de $\left(\cup_{\ell=0}^{j} \mathcal{V}_\ell\right) \cap A$. Si μ_1 et μ_2 sont deux neurones distincts de la $j+1$-ème couche appartenant

à l'ensemble A, on voit que l'ensemble des ancêtres de μ_1 et celui des ancêtres de μ_2 au sens de la Définition 4.15 sont disjoints. Puisque l'état d'activation d'un neurone de la $j+1$-ème couche ne dépend que des coordonnées d'entrée associées à ses ancêtres, il suffit donc de montrer que pour tout neurone μ de $\mathcal{V}_{j+1} \cap A$, on peut modifier les coordonnées de $\mathbf{x}_j$ associées aux ancêtres de μ de façon à activer μ tout en préservant l'état d'activation des neurones de $\left(\cup_{\ell=0}^{j} \mathcal{V}_\ell\right) \cap A$. Soit donc μ un neurone quelconque de $\mathcal{V}_{j+1} \cap A$. Notons μ_1 et μ_2 les prédécesseurs de μ. Ces deux neurones appartiennent à A, et leurs ensembles d'ancêtres sont disjoints. Alors il existe un scalaire $r > 0$ assez grand pour que, en notant $\mathbf{x}'_j$ le vecteur obtenu à partir de $\mathbf{x}_j$ en multipliant par r toutes les composantes associées aux ancêtres de μ_1, et $\mathbf{x}''_j$ le vecteur obtenu à partir de $\mathbf{x}_j$ en multipliant par r toutes les composantes associées aux ancêtres de μ_2, au moins l'un des vecteurs $\mathbf{x}'_j$ et $\mathbf{x}''_j$ active μ. En effet, $\mathbf{x}_j$ active les neurones μ_1 et μ_2, et donner $\mathbf{x}'_j$ en entrée revient à multiplier la préactivation de μ_1 par r par rapport à celle de $\mathbf{x}_j$, tandis que μ_2 induit une multiplication par r dans la préactivation de μ_2. Or pour tout vecteur d'entrée $\mathbf{x}$, on a

$$(z_{j+1}(\mathbf{x}))_\mu = w_{(\mu_1 \to \mu)}(z_j(\mathbf{x}))_{\mu_1} + w_{(\mu_2 \to \mu)}(z_j(\mathbf{x}))_{\mu_2}, \tag{45}$$

avec au moins un terme strictement positif parmi $w_{(\mu_1 \to \mu)}(z_j(\mathbf{x}))_{\mu_1}$ et $w_{(\mu_2 \to \mu)}(z_j(\mathbf{x}))_{\mu_2}$, puisque l'on est sous l'hypothèse (iii). Ceci justifie l'affirmation précédente. De plus, on peut vérifier que $\mathbf{x}'_j$ comme $\mathbf{x}''_j$ induisent les mêmes états d'activation de neurones que $\mathbf{x}_j$ sur $\cup_{\ell=0}^{j} \mathcal{V}_\ell$. Ce qui clôt la récurrence. □

Démonstration de la Proposition/Définition 4.24. Soit $\mathcal{R}$ un réseau butterfly à $J+1$ couches, fortement reconstructible et vérifiant l'hypothèse (H). En vertu du Lemme 4.22, on dispose pour chaque neurone de sortie ν d'un vecteur d'entrée $\mathbf{x}_\nu$ activant l'arbre associé à ν. Alors $(\mathbf{x}_\nu)_{\nu \in \mathcal{V}_J}$ est de cardinal N, et vérifie la propriété (F), car cette famille active tous les chemins d'activation de $\mathcal{R}$, ce qui correspond à la condition suffisante de la Proposition 4.13. Ainsi, $C_{\max} \leq N$. □

Démonstration de la Proposition/Définition 4.25. Ce résultat s'obtient en considérant un réseau particulier, que nous nommerons *réseau adversaire*, au sens où il a un grand cardinal reconstruisant.

Définition A.1 (réseau adversaire)**.** *Le réseau adversaire $\mathcal{A}_J$ à $J+1$ couches (sans biais) est défini par $\mathbf{W}_j = -\mathbf{S}_j + 2\mathbf{I}_N$, où $\mathbf{S}_j$ est le j-ème support butterfly, pour $j \in [\![J]\!]$. On montre aisément par récurrence que $\mathcal{A}_J$ est un réseau butterfly, et que ses matrices de poids ont des 1 sur leur diagonale et des -1 ailleurs au sein des supports butterfly.*

Ce réseau a la propriété particulière suivante : pour tout neurone $\nu \in \mathcal{V}_0 \cup H$, ses successeurs μ_1 et μ_2 ne sont jamais simultanément activés par un vecteur d'entrée, car par construction les préactivations de μ_1 et μ_2 sont toujours de signe opposé. Il s'ensuit que pour tout $j \in [\![J-2]\!]$, les arêtes reliant $\mathcal{V}_j$ à $\mathcal{V}_{j+1}$ peuvent être divisées en $\frac{N}{2}$ groupes de 4 arêtes deux à deux incompatibles (c'est-à-dire non activables simultanément), de la forme représentée dans la Figure 7, où les neurones de gauche sont les successeurs d'un même neurone de la couche $j-1$.

Montrons par récurrence sur $J \geq 2$ que le cardinal fortement reconstruisant du réseau $\mathcal{A}_J$ est supérieur ou égal à $2(J-1)$. Quand $J = 2$, c'est le cas car tous les neurones cachés de $\mathcal{A}_2$ ne sont pas activables simultanément, donc un seul vecteur d'entrée ne permet pas la reconstruction. Soit $J \geq 2$ tel que le résultat est vrai pour $\mathcal{A}_J$. Donnons-nous une famille $\mathcal{F} = (\mathbf{x}_1, \ldots, \mathbf{x}_K)$ fortement reconstruisante minimale pour $\mathcal{A}_{J+1}$ et montrons que son cardinal est supérieur ou égal à $2(J-1)$. Notons $M = (M_1, M_2) \colon \mathbb{R}^N \to \mathbb{R}^{N/2} \times \mathbb{R}^{N/2}$ l'opérateur divisant tout vecteur $\mathbf{x}$ en deux sous-vecteurs, le premier constitué de ses $N/2$ premières coordonnées, et le deuxième de ses $N/2$ dernières coordonnées. Alors les familles $(M_1(\mathbf{x}_1), \ldots, M_1(\mathbf{x}_K))$ et $(M_2(\mathbf{x}_1), \ldots, M_2(\mathbf{x}_K))$ sont toutes deux fortement reconstruisantes pour le réseau $\mathcal{A}_J$. Ceci découle de la Conjecture

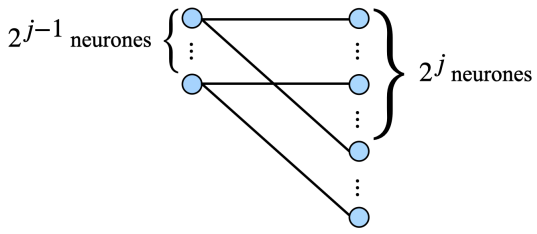


FIGURE 7 – Groupe de quatre arêtes incompatibles reliant la j-ème couche de neurones à la $(j+1)$-ème.

4.16, en remarquant que le graphe de $\mathcal{A}_{J+1}$ contient deux sous-graphes identiques au graphe de $\mathcal{A}_J$, obtenus comme les deux composantes connexes du graphe de $\mathcal{A}_{J+1}$ auquel on aurait enlevé la dernière couche de neurones ainsi que les arêtes correspondantes (voir Figure 5). Nous pouvons donc choisir des indices $i_1, \dots, i_\ell$ tels que $(M_1(\mathbf{x}_{i_1}), \dots, M_1(\mathbf{x}_{i_\ell}))$ est une famille fortement reconstruisante pour $\mathcal{A}_J$, et dont on ne peut enlever aucun vecteur sans perdre le caractère reconstruisant. On montre en exploitant les propriétés d'incompatibilité entre arêtes qu'une telle famille, de cardinal supérieur ou égal à $2(J-1)$ par hypothèse de récurrence, ne peut cependant vérifier les propriétés (i) à (iii) de la Conjecture 4.16 en ce qui concerne les neurones de la dernière couche cachée de $\mathcal{A}_{J+1}$. La famille $\mathbf{x}_1, \dots, \mathbf{x}_K$ contient donc au moins deux vecteurs de plus que $\mathbf{x}_{i_1}, \dots, \mathbf{x}_{i_\ell}$ (deux et non pas un en raison des propriétés d'incompatibilité entre arêtes), ce qui permet de conclure. □