



**HAL**  
open science

## Learning attributed graphs representations

Yacouba Kaloga

► **To cite this version:**

Yacouba Kaloga. Learning attributed graphs representations. Intelligence artificielle [cs.AI]. Université de Lyon, 2021. Français. NNT : 2021LYSEN086 . tel-03610111

**HAL Id: tel-03610111**

**<https://theses.hal.science/tel-03610111v1>**

Submitted on 16 Mar 2022

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



N° National de Thèse : 2021LYSEN086

**THESE DE DOCTORAT DE L'UNIVERSITE DE LYON**  
opérée par  
**l'Ecole Normale Supérieure de Lyon**

**ECOLE DOCTORALE N° 52**  
**École Doctorale de Physique et d'Atrophysique**

**Discipline : Physique**

Soutenue publiquement le 17 décembre 2021

**Yacouba Kaloga**

---

**Apprentissage de représentations de graphes  
attribués**

---

Devant le jury composé de :

Mme.	Céline HUDELLOT	PU	<i>CentraleSupélec</i>	Rapportrice
M.	Paul HONEINE	PU	<i>Université de Rouen Normandie</i>	Rapporteur
M.	Romain COUILLET	PU	<i>Université Grenoble-Alpes</i>	Président
M.	Pierre BORGNAT	DR-CNRS	<i>ENS de Lyon, CNRS</i>	Directeur
M.	Amaury HABRARD	PU	<i>Université Jean Monnet de Saint-Etienne</i>	Co-encadrant



Laboratoire de Physique - IXXI  
ENS de Lyon  
46 allée d'Italie  
69364 Lyon cedex 07

École doctorale PHAST (ED 52)  
Université Lyon 1  
43 boulevard du 11 Novembre 1918  
69622 Villeurbanne cedex



# Remerciements

Merci ! Ce mot que l'on prononce et entend quotidiennement sans y prêter attention, une marque de politesse qui ne se remarque souvent que par son absence. C'est au moment de l'écrire que son sens vous frappe et que sa portée symbolique vous touche. Alors je voudrais dire merci, merci et encore merci.

Je remercie mes deux directeurs de thèse Pierre BORGNAT et Amaury HABRARD qui m'ont fait confiance et qui m'ont laissé une grande liberté de choix dans ma thèse.

Je remercie également Marion FOARE et Ievgen REDKO pour leurs temps et les conseils qu'ils ont pu me prodiguer.

Je veux aussi remercier Nelly PUSTELNIK, Pablo JENSEN, Thibaut DIVOUX et Eric FREYSSINGEAS qui par leurs actions m'ont convaincu et mené mes pas jusqu'à la réalisation de cette thèse.

Je tiens également à remercier mes professeurs de classe préparatoire tout particulièrement Serge LEGOFF et Alain WALBRON qui m'ont donné comme à beaucoup d'autres le goût des sciences. Je remercie aussi Céline HUDELLOT, Paul HONEINE et Romain COUILLET d'avoir accepté et pris le temps d'évaluer mes travaux de thèse.

Mille mercis à Selma pour son aide et ses précieux conseils.

Enfin, je remercie mes amis d'être tout simplement mes amis, et de participer à mon bonheur.

Enfin, merci à ma famille et à mes parents, à qui je dois tant.



# Table des matières

<b>Index des notations</b>	<b>7</b>
<b>1 Introduction</b>	<b>10</b>
1.1 Les graphes comme modèles de données	13
1.2 Réseaux de neurones sur graphes	15
1.3 Motivations de la thèse	16
<b>2 Généralités sur les graphes</b>	<b>19</b>
2.1 Notions de théorie des graphes	19
2.1.1 Graphe	20
2.1.2 isomorphisme de graphes	26
2.1.3 Heuristiques, noyaux et invariants	27
2.2 Analyse spectrale	30
2.2.1 Laplacien, spectre et harmonique	31
2.2.2 Transformée de Fourier sur graphes	34
2.3 Réseaux de neurones convolutifs sur graphes	36
2.3.1 Préliminaires : apprentissage avec réseaux de neurones	37
2.3.2 Motivations	42
2.3.3 Problématique : des CNN aux GCN	42
2.3.4 Approximation polynomiale des GCN	44
2.3.5 Deux cas particuliers : Truncated Krylov et Simple GCN	46
2.3.6 Conclusion	48
<b>3 Représentation hiérarchique non supervisée de graphes</b>	<b>49</b>
3.1 Introduction	49
3.1.1 Contexte	50
3.2 Définitions et contexte	53
3.2.1 Algorithme de Weisfeiler-Lehman - WL	54
3.2.2 <i>Negative Sampling</i> et Information Mutuelle	55
3.2.3 Graph2vec	57
3.3 <i>Hierarchical Graph2vec</i> - HG2V	58
3.3.1 Méthode de <i>pooling</i> de Loukas	59
3.3.2 Construction hiérarchique du modèle	65



3.3.3	Attributs continus et <i>truncated Krylov</i> . . . . .	65
3.3.4	<i>Negative Sampling</i> hiérarchique . . . . .	67
3.4	Expériences . . . . .	69
3.4.1	Jeux de données . . . . .	71
3.4.2	Classification supervisée . . . . .	72
3.4.3	Inductivité . . . . .	74
3.4.4	Étude ablativité . . . . .	76
3.4.5	Espace latent . . . . .	77
3.5	Conclusion . . . . .	77
<b>4</b>	<b>Analyse des corrélations canoniques appliquée aux graphes</b>	<b>79</b>
4.1	Introduction . . . . .	79
4.2	Définitions . . . . .	81
4.2.1	Jeux de données multivues . . . . .	82
4.2.2	. . . et localement structurés . . . . .	83
4.3	Etat de l'art . . . . .	83
4.3.1	CCA linéaire . . . . .	83
4.3.2	Méthodes Algébriques . . . . .	85
4.3.3	Méthodes Probabilistes . . . . .	88
4.3.4	Auto-encodeur variationnel - VAE . . . . .	90
4.4	<i>Multiview Graph Canonical Correlation Analysis</i> . . . . .	92
4.4.1	Modèle : propriétés et mise en équation . . . . .	92
4.4.2	Vraisemblance . . . . .	94
4.4.3	Approche variationnelle . . . . .	95
4.4.4	Entraînements et inférences . . . . .	97
4.4.5	Robustesse et <i>views dropout</i> . . . . .	98
4.5	Expériences . . . . .	101
4.5.1	Jeux de données . . . . .	101
4.5.2	Paramètres . . . . .	102
4.5.3	Classification et <i>clustering</i> . . . . .	103
4.5.4	Twitter friend recommandation . . . . .	105
4.5.5	Inférences . . . . .	107
4.6	Conclusion . . . . .	112
<b>5</b>	<b>Apprentissage de métrique entre graphes</b>	<b>113</b>
5.1	Introduction . . . . .	113
5.2	Définitions et contexte . . . . .	115
5.2.1	Apprentissage de métrique . . . . .	116
5.2.2	Transport optimal . . . . .	120
5.2.3	Distance et noyau sur graphe avec le transport optimal . . . . .	124
5.3	<i>Simple Graph Metrics Learning</i> . . . . .	126
5.3.1	Génération des caractéristiques des graphes . . . . .	127
5.3.2	Distance de <i>Sliced-Wasserstein</i> appliquée aux graphes . . . . .	127
5.3.3	Fonctionnelle d'apprentissage : <i>Nearest Class Cloud Metric Learning</i> . . . . .	132

---

5.3.4	Complexité et optimisation	133
5.4	Expériences	134
5.4.1	Classification	134
5.4.2	Noyau	134
5.4.3	Méthodes des plus proches voisins	135
5.5	Conclusion	136
<b>6</b>	<b>Conclusion</b>	<b>137</b>
6.1	Résumé des contributions	137
6.2	Perspectives futures	138
<b>A</b>	<b>Résultats complémentaires</b>	<b>141</b>
A.1	Chapitre 2 - Généralités	141
A.1.1	Complexité informatique	141
A.1.2	Graphes co-spectraux	142
A.2	Chapitre 3 - HG2V	143
A.2.1	Weisfeiler-Lehman	143
A.2.2	Continuité des GNN	144
A.2.3	Diffusion Limited Aggregation	145
A.2.4	Construction de graphes à partir d'images : USPS, MNIST et FASHION-MNIST	146
A.2.5	HG2V entraînement sur une fraction des données	146
A.2.6	Visualisation de l'espace latent	147
A.3	Chapitre 4 - MVGCCA	149
A.3.1	Modèles algébriques alternatifs	149
A.3.2	Forme explicite de l'ELBO	150
A.3.3	Représentation latente	151
A.3.4	Construction d'un graphe synthétique	152
A.4	Chapitre 5 - SGML	153
A.4.1	<i>Graph Metric Learning</i> dans la littérature	153
A.4.2	Processus ponctuels déterminantaux	153
A.4.3	Quasi Monte-Carlo sur l'hypersphère	154



# Index des notations

## Généralités

$i, j, k, l, m, n, p, q, d$  Ces lettres désignent des entiers naturels.  
 **$y, Y$**  Les lettres en gras minuscules désignent des vecteurs, et celles en gras majuscules des matrices. La  $i$ -ième composante du vecteur  $y$  est notée  $y_i$ . La composante en position  $i, j$  de la matrice  $Y$  est notée  $Y_{i,j}$ , tandis que la  $i$ -ième ligne et la  $i$ -ième colonne de  $Y$  sont notées respectivement  $Y(i, :)$  et  $Y(:, i)$ .

## Ensembles

$\mathbb{N}, \mathbb{R}$  Ensemble des entiers naturels et des réels.  
 $\mathbb{R}_+$  Ensemble des réels positifs.  
 $\mathbb{R}^*, \mathbb{N}^*$  Ensemble des réels/entiers naturels privé de 0.  
 $\mathbb{S}^n$  Hypersphère de dimension  $n$ ,  $\mathbb{S}^n = \{\theta \in \mathbb{R}^{n+1} \mid \|\theta\|_2 = 1\}$ .  
 $\mathcal{S}_n$  Ensemble des permutations de taille  $n$ .  
 $|\mathbb{X}|$  Désigne le cardinal de l'ensemble  $\mathbb{X}$ , égale à son nombre d'éléments lorsque ce dernier est fini.  
 $\mathbb{X} \simeq \mathbb{Y}$   $\mathbb{X}$  et  $\mathbb{Y}$  sont isomorphes.  
 $\mathbb{X} \cup \mathbb{Y}$  Désigne l'ensemble constitué des éléments de  $\mathbb{X}$  et  $\mathbb{Y}$ .  
 $f \in \mathbb{Y}^{\mathbb{X}}$  Désigne une fonction  $f : \mathbb{X} \rightarrow \mathbb{Y}$ .  
 $f^{-1}(y) \subseteq \mathbb{X}$   $f^{-1}(y) = \{x \in \mathbb{X} \mid y = f(x)\}$ .  
 $[a, b], ]a, b[$  Ensemble des réels compris entre  $a$  et  $b$  inclus (resp. exclus).  
 $\llbracket p, q \rrbracket$  Ensemble des entiers compris entre  $p$  et  $q$  inclus,  $\{p, p+1, \dots, q-1, q\}$ .

## Algèbre linéaire

$A, A^T, A^{-1}$  Une matrice, sa transposée et son inverse.  $(A)_{i,j} = (A^T)_{j,i}$ .  $AA^{-1} = I_n$ .  
 $A \succcurlyeq 0$  Matrice semi-définies positives. Pour  $A \in \mathbb{R}^{n \times n}$ ;  $\forall x, x^T A x \geq 0$ .  
 $\|A\|_{\mathcal{F}}$  Norme de Frobenius.  $\|A\|_{\mathcal{F}} = \sqrt{\text{tr}(AA^T)}$ .  
 $\mathcal{O}_n(\mathbb{R})$  Ensemble des matrices orthogonales de  $\mathbb{R}^{n \times n}$ .  $A^{-1} = A^T$ .  
 $I_n, O_n$  Matrice identité, nulle de  $\mathbb{R}^{n \times n}$ .  
 $\mathbf{1}_n, \mathbf{0}_n$   $\mathbf{1}_n = [1, \dots, 1]^T \in \mathbb{R}^n$ ,  $\mathbf{0}_n = [0, \dots, 0]^T \in \mathbb{R}^n$ .  

 $[ \ ]$  Opérateur de concaténation horizontale.  $[A \in \mathbb{R}^{p \times n}, B \in \mathbb{R}^{p \times m}] \in \mathbb{R}^{p \times (n+m)}$ .

**Graphes**

$\mathcal{G} = (V, E)$	Un graphe caractérisé par l'ensemble de ses sommets (ou nœuds) et de ses arêtes.
$u_i, u_j \in V$	Ensemble des sommets du graphe.
$\{u_i, u_j\} \in E$	Ensemble des arêtes du graphe.
$x$	Signal $x : V \rightarrow \mathbb{X}$ sur les nœuds du graphe.
$x(u_i), \mathbf{x}$	Lorsque $\mathbb{X} = \mathbb{R}$ on pourra identifier le signal $x$ à un vecteur $\mathbf{x}$ de $\mathbb{R}^{ V }$ , tel que $x_i = x(u_i)$ .
$x(u_i), \mathbf{X}$	Lorsque $\mathbb{X} = \mathbb{R}^q$ avec $q > 1$ , on pourra identifier le signal $x$ à une matrice $\mathbf{X}$ de $\mathbb{R}^{ V  \times q}$ , tel que $\mathbf{X}(i, :) = x(u_i)^T$ .
$\mathbf{A}, \mathbf{D}$	Matrice d'adjacence du graphe. $\mathbf{A}_{i,j} > 0$ . Matrice diagonale des degrés. $\mathbf{D}_{i,i} = \sum_j \mathbf{A}_{i,j}$ .
$\mathbf{L}, \mathcal{L}$	Laplacien combinatoire $\mathbf{L} = \mathbf{D} - \mathbf{A}$ et laplacien normalisé $\mathcal{L} = \mathbf{I}_n - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}$ .
$d_{i,j}^{\mathbf{A}}$	Plus courte distance dans le graphe entre les sommets $u_i$ et $u_j$ dans le graphe $\mathcal{G}$ avec pour matrice d'adjacence $\mathbf{A}$ .
$\mathcal{V}_p(u_i)$	$\mathcal{V}_p(u_i) = \{v_j \in V \mid d_{i,j}^{\mathbf{A}} \leq p\}$ . Ensemble contenant $u_i$ (car $d_{i,i}^{\mathbf{A}} = 0$ ) et ses voisins se trouvant à une distance inférieure à $p$ . Noté $\mathcal{V}(u_i)$ lorsque $p = 1$ .

**Probabilités**

$\mathcal{P}(\mathbb{X})$	Ensemble des distributions sur l'ensemble $\mathbb{X}$ .
$x \sim p$	La variable $x$ suit la distribution $p$ .
$\mathcal{N}(\mu, \sigma^2)$	Loi normale de moyenne $\mu$ et de variance $\sigma^2$ .
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Psi})$	Loi normale multidimensionnelle de moyenne $\boldsymbol{\mu}$ et de matrice de covariance $\boldsymbol{\Psi}$ .
$\mathcal{B}(n, p)$	Loi de Bernouilli de paramètres $n$ et $p$ .
$\delta_{x,y}$	$\delta_{x,y} = 0$ si et seulement si $x = y$ .
$\delta_x$	$\delta_x : y \mapsto \delta_{x,y}$ .
$\tau$	Fonction d'activation sigmoïde, $\tau : x \mapsto \frac{1}{1+e^{-x}}$ .
ReLU	Fonction d'activation <i>Rectified Linear Unit</i> , ReLU : $x \mapsto \max(x, 0)$ .

**Abréviations**

<i>GNN</i>	<i>Graph Neural Networks.</i>
<i>GCN</i>	<i>Graph Convolutionnal Neural Networks.</i>
<i>RNN</i>	<i>Recurrent Neural Networks.</i>
<i>CCA</i>	<i>Canonical Correlation Analysis.</i>
<i>WL</i>	<i>Weisfeiler-Lehman.</i>
<i>SW</i>	<i>Sliced-Wasserstein.</i>





# Chapitre 1

## Introduction

Au XVIII<sup>e</sup> siècle, la ville prussienne de Königsberg<sup>1</sup> s'étale sur les rives du fleuve Pregolia et ses deux îles fluviales. Les différentes parties de la ville sont reliées par 7 ponts, les habitants se demandent s'il existe une promenade permettant de passer une et une seule fois par chaque pont tout en revenant à son point de départ. C'est le fameux problème des 7 ponts de Königsberg.

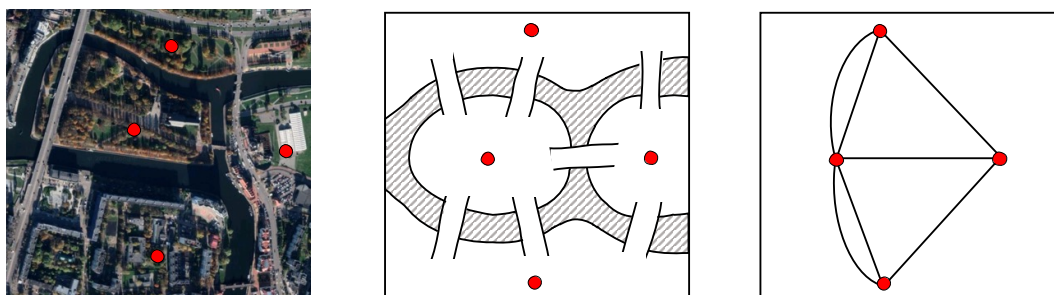


FIGURE 1.1 – *Ville de Königsberg*. **A gauche**, la ville de Königsberg, ses presqu'îles et ses ponts. **Au milieu**, un schéma de la ville, deux des ponts reliant les berges à la presqu'île à gauche n'existent plus aujourd'hui. **A droite**, le graphe équivalent.

En 1735, Leonhard Euler présente dans un article à l'Académie des sciences de Saint-Petersbourg une description mathématique du problème. Il montre que le problème se réduit à l'étude d'un nouvel objet mathématique constitué de points reliés entre eux par des arêtes, ce qu'on appelle aujourd'hui un graphe<sup>2</sup>. Dans sa démarche, Euler généralise le problème et traite la question dans le cas d'un graphe quelconque, il pose ainsi la première pierre de la théorie des graphes. Dans ce cadre, la question équivaut à rechercher un chemin qui part d'un sommet et y revient en passant une et une seule fois par chaque arête (ce qu'on appelle aujourd'hui un cycle Eulérien). Euler postule qu'il existe un tel cycle si et seulement si le graphe est connexe (il existe toujours un chemin allant d'un nœud à un autre) et si le nombre d'arêtes connectées à chaque nœud est pair (à chaque fois qu'on rentre dans un sommet par une arête, il en faut

1. Devenue Kaliningrad en Russie.

2. Plus précisément, Euler a présenté le graphe sous-jacent du problème sous la forme d'une liste d'arêtes. Les arêtes seront définies au Chapitre 2.



une autre pour en sortir). Le graphe de la ville de Königsberg comportant un nombre impair d'arêtes, la promenade rêvée par ses habitants est impossible. La démonstration rigoureuse du postulat d'Euler n'est apportée qu'en 1873 par le mathématicien Badois<sup>3</sup>, Carl Hierholzer<sup>4</sup>.

Par la suite, l'ensemble des types de graphes possibles a été caractérisé plus finement (arbre, graphe planaire, cyclique, *etc.*) et nettement enrichi (graphe pondéré, orienté, attribué, *etc.*).

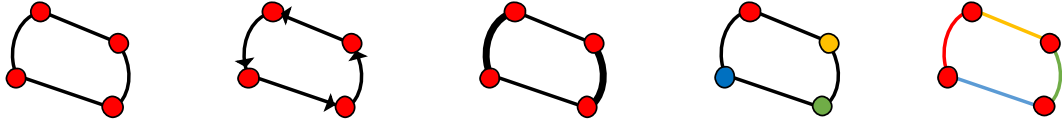


FIGURE 1.2 – *Différentes familles de graphes.* De gauche à droite : Un **graphe standard**, les nœuds et les arêtes sont de même nature. Un **graphe orienté**, les arêtes sont dirigées. Un **graphe pondéré**, les arêtes sont associées à des poids positifs qui caractérisent l'intensité du lien. Pour terminer, deux **graphes attribués**, les nœuds et/ou les arêtes portent une information. La liste n'est pas exhaustive ; un graphe peut avoir plusieurs de ses propriétés simultanément.

L'étude de ces familles de graphes a mené à la conception de nouvelles théories (il y a de fait *des* théories des graphes) dont la portée va bien au-delà des mathématiques, avec des apports certains à la physique, les sciences des matériaux, l'économie, l'épidémiologie, la chimie et les systèmes complexes entre autres. Les graphes sont notamment très utiles pour modéliser des mouvements avec des contraintes de déplacement et de volume, en attribuant des capacités aux arêtes. Ils permettent ainsi de modéliser (et donc optimiser) des flots, provenant de sources à acheminer vers des puits. Ils en découlent des applications directes telles que la conception et le dimensionnement d'oléoducs, de réseaux routiers ou encore de réseaux fluviaux [Williamson 2019 ; Sharifov and Kutucu 2018].

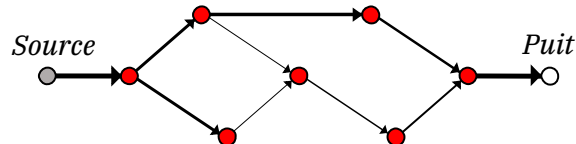


FIGURE 1.3 – *Graphe de flots.* On utilise un graphe orienté et pondéré pour modéliser l'unidirectionnalité de certains flots et leur capacité maximale de transport. Ces graphes sont l'objet du théorème flot-max/coupe-min, un théorème important en théorie des graphes et en optimisation, permettant notamment de déterminer le flot maximal qui puisse passer d'un ensemble de sources à des puits.

Les apports des théories des graphes ne se limitent pas à la modélisation de flots. En physique statistique, les graphes permettent de prévoir des phénomènes de transition de phase complexe : le modèle d'Ising [Ising 1925] modélise les matériaux ferromagnétiques<sup>5</sup> comme des réseaux de

3. Aujourd'hui c'est une région du sud-est de l'Allemagne, intégrée au lander de Bade-Wurtemberg.

4. Mort prématurément en 1871. Ce résultat a été publié à titre posthume.

5. Matériaux acquérant une aimantation sous l'action d'un champ magnétique dans le sens de ce dernier et

spins et permet d'expliquer la capacité de ces matériaux à devenir paramagnétiques<sup>6</sup> à partir d'une certaine température<sup>7</sup>.

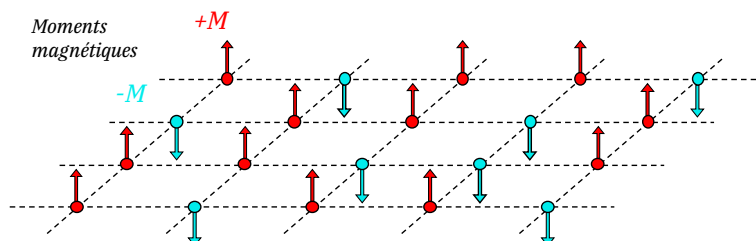


FIGURE 1.4 – *Modèle d'Ising 2d*. A chaque nœud de la grille, il y a un atome qui possède un moment magnétique  $\pm \mathbf{M}$  qui fluctue selon l'agitation thermique environnante. Les atomes interagissent avec leurs voisins et ont tendance à aligner leurs moments magnétiques, ce qui se traduit macroscopiquement par des propriétés ferromagnétiques. A partir d'une certaine température, l'agitation thermique domine et empêche les atomes de s'aligner, le matériau devient alors paramagnétique.

En informatique, le fameux<sup>8</sup> problème des quatre couleurs consiste à rechercher la coloration d'une carte avec 4 couleurs de telle sorte que les pays voisins aient des couleurs différentes, c'est en fait un problème de théorie des graphes. Cela équivaut à rechercher une coloration avec 4 couleurs des sommets d'un graphe planaire, telle que les sommets adjacents aient des couleurs différentes. C'est le tout premier théorème démontré avec l'assistance d'un ordinateur [Appel and Haken 1977], il est à l'origine du développement de nombreuses techniques de vérification de preuve automatique.

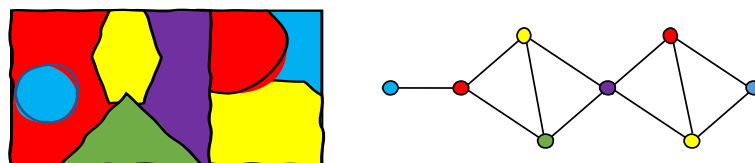


FIGURE 1.5 – *Illustration du problème des 4 couleurs*. Une carte colorée avec 4 couleurs et le graphe planaire associé.

Ces exemples témoignent de la diversité des applications des théories des graphes. Près de 150 ans après leur introduction par Euler, les graphes n'ont pas cessé d'être des sujets fertiles pour la recherche suscitant un intérêt constant de la communauté scientifique jusqu'à nos jours.

qui la *conserve* lorsque le champ devient nul : fer, nickel, cobalt, *etc.*

6. A contrario des matériaux ferromagnétiques, ceux-ci perdent leur aimantation dès que le champ appliqué devient nul : aluminium, lithium, platine *etc.*

7. Température de Curie.

8. Introduit en 1852 par le botaniste et mathématicien sud-africain Francis Guthrie, il fut résolu seulement en 1976 soit 124 ans après, et à l'aide d'un ordinateur !

## 1.1 Les graphes comme modèles de données

Aujourd’hui, les graphes se retrouvent dans de nombreux domaines, où ils permettent de représenter de nombreux types de données (réseaux sociaux, molécules, ...) qui possèdent une structure locale *irrégulière*; en opposition aux données euclidiennes telles que les vecteurs (position, vitesse, ...), les matrices (image en noir et blanc, musique, ...) ou encore les tenseurs (image couleur, vidéo ...) qui possèdent une structure locale *régulière* (directement la cause du fait qu’ils sont *globalement structurés*). Par exemple, les images sont constituées de pixels reliés entre eux par une grille fixe, qui permet de distinguer sans ambiguïté les voisins autour d’un pixel et donc de définir aisément des notions de similarités locales<sup>9</sup> [Petrou and Petrou 2010]. Cette structure permet également d’établir des distances à l’échelle globale entre les images (de même taille). A contrario, les graphes ne possèdent pas de structure régulière; le plus souvent les différents voisins autour d’un nœud ne sont pas distinguables les uns des autres. Plus encore, deux nœuds peuvent ne pas avoir le même nombre de voisins, ce qui se traduit par une certaine difficulté à définir des notions de similarités entre graphes tant localement que globalement [Caetano et al. 2009]. En conséquence, les distances entre graphes sont en général difficiles à calculer et ne serait-ce que de déterminer si des graphes quelconques sont identiques est un problème difficile (voir Section 2.1.2 sur les isomorphismes de graphes). Malgré ces écueils, les graphes ont suscité ces dernières années un vif intérêt de la communauté de l’apprentissage automatique car ils sont des modèles naturels de description pour un grand nombre de données et les applications potentielles associées sont prometteuses.

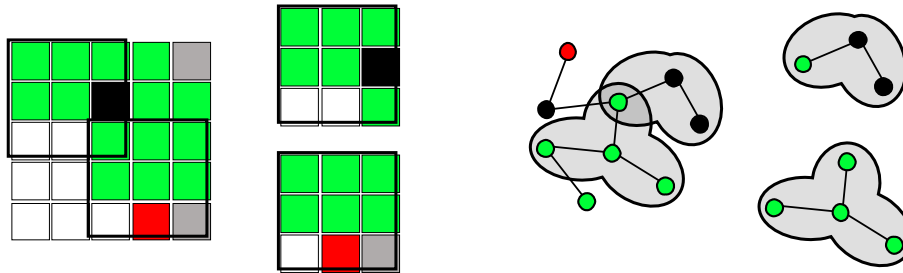


FIGURE 1.6 – *Données à structure locale régulière et non régulière.* **A gauche**, une image  $5 \times 5$  et les voisinages de deux de ses pixels. La comparaison des deux voisinages est aisée, on constate qu’ils ne diffèrent que de 3 pixels. On peut ainsi définir quantitativement si un voisinage est plus ou moins proche de l’autre. Seuls les pixels du bord de l’image possèdent des voisinages atypiques, mais il suffit de considérer que l’image est périodique ou d’ajouter des pixels de valeurs nulles au bord de l’image pour résoudre cette difficulté. **A droite**, un graphe et les voisinages de deux de ses nœuds. Là, la comparaison est beaucoup plus difficile, on peut observer que les deux voisinages ont un nœud en commun, mais le nombre de voisins étant différent il est difficile d’évaluer quantitativement leur proximité.

Ces applications se distinguent selon deux points de vues desquels résultent deux façons

9. Cette propriété est très importante pour les réseaux de neurones convolutionnels (*Convolutional neural networks* – CNN – voir Section 2.3.2) qui ont connu un très grand succès ces dernières années.

d'aborder les questions d'apprentissage relatives aux graphes. D'une part, le graphe peut être considéré comme une représentation de relations existantes entre des *objets* d'études, qui constitueront donc les nœuds de ce dernier [Ortega et al. 2018]. Ce point de vue privilégie les tâches d'apprentissage relatives à ses nœuds. Par exemple dans un réseau social, où les nœuds sont des individus et les arêtes des relations entre les individus ; les tâches de détection de communauté, de classification des utilisateurs ou encore de recommandation de contenus et d'amitié sont des tâches relatives à ce point de vue et qui peuvent bénéficier de la prise en compte de l'information structurelle apportée par le graphe.

D'autre part, l'autre point de vue consiste à considérer le graphe comme une structure (potentiellement dynamique) support d'une information portée par les nœuds ou les arêtes, à l'instar des images dont les pixels portent les informations qui constituent l'image. Cette seconde approche s'intéresse aux propriétés du graphe en tant que tel [Riesen and Bunke 2010]. Les tâches de classification, de *clustering*, de régression ou encore de visualisation de graphe sont des tâches relatives à ce point de vue. La chimie est un domaine qui pourrait grandement bénéficier d'amélioration dans ce sens [Balaban 1985 ; Sakai et al. 2021] ; la possibilité de pouvoir classer, trier, ordonner ou générer des molécules en fonction de propriétés d'intérêts, c'est la perspective de passer outre le temps et les coûts de développement faramineux de nouveaux médicaments, et ainsi accélérer la recherche médicale.

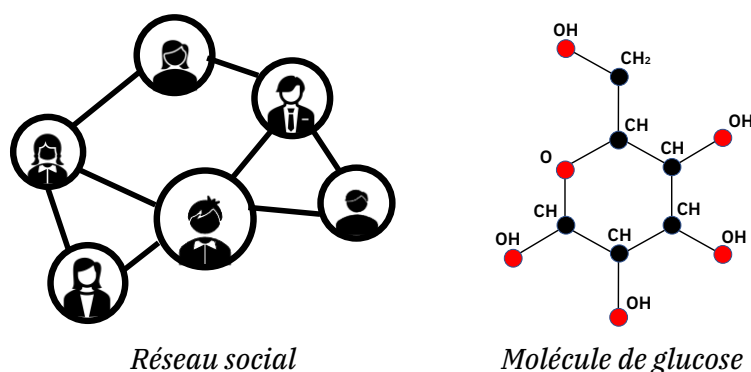


FIGURE 1.7 – *Différentes échelles d'intérêt.* **A gauche**, l'échelle d'intérêts est celle des nœuds, tandis qu'**à droite** c'est celle du graphe. Toutefois, pour comprendre l'une ou l'autre de ces échelles, on est souvent amené à considérer l'autre échelle ; voire des échelles intermédiaires.

Évidemment, ces deux points de vue ne constituent pas une délimitation exhaustive des méthodes d'apprentissage liées aux graphes ; on est souvent dans des situations mixtes. Ils permettent néanmoins de cerner les deux échelles d'intérêt dans les graphes. Dans cette thèse, on va s'intéresser à des tâches d'apprentissage sur les graphes provenant de l'un et de l'autre point de vue. Nous discuterons notamment des limitations existantes de certaines méthodes d'apprentissage sur graphes et des solutions que nous avons retenues pour les surmonter. Avant de rentrer dans le détail des contributions de cette thèse, nous allons décrire brièvement le contexte dans lequel s'est inscrit ce travail, en particulier vis-à-vis des réseaux de neurones qui sont devenus très présents dans l'état de l'art. Les résultats spectaculaires obtenus ces dernières

années à l'aide de ces réseaux expliquent l'orientation prise par la communauté de l'apprentissage automatique et la concentration de la recherche autour de ces réseaux, y compris dans le domaine de l'apprentissage sur graphes.

## 1.2 Réseaux de neurones sur graphes

Au début des années 2010, la conjonction de l'abondance des données, de l'amélioration importante de la puissance de calcul et du travail de la communauté de l'apprentissage automatique a permis d'obtenir à l'aide des réseaux de neurones des améliorations fulgurantes notamment dans le traitement et la classification d'images (notamment grâce aux CNN) et plus généralement des données avec une structure locale régulière [LeCun et al. 1998; Krizhevsky et al. 2012]. L'efficacité des réseaux de neurones a incité la communauté de l'apprentissage automatique à étendre leurs domaines d'applications, notamment aux graphes. Les réseaux de neurones sur graphes (*Graph Neural Network* - GNN) ont alors rapidement gagné en popularité.

**Historique.** S'il existait déjà des réseaux de neurones pour graphes à la fin des années 90 [Sperduti and Starita 1997; Frasconi et al. 1998], il s'agissait essentiellement d'adaptations de réseaux de neurones récurrents<sup>10</sup> (*Recurrent Neural Networks* - RNN) s'appliquant exclusivement à des graphes acycliques, et des graphes de tailles modestes ; ce qui limitait leur intérêt. C'est au milieu des années 2000 que le terme réseau de neurones sur graphes apparaît [Gori et al. 2005] avec les premières tentatives d'extension de ces RNN à des familles de graphes plus grandes. Ces efforts ont été poursuivis à la fin des années 2000 avec la publication de méthodes d'apprentissage intégrant les graphes cycliques [Micheli 2009] (et potentiellement de grande taille [Scarselli et al. 2009]) combinant des RNN et des réseaux de neurones à propagation avant (*Feed forward Neural Networks* - FNN). Toutefois, le point de bascule à partir duquel ces réseaux sont devenus fortement populaires se situe au début des années 2010 et trouve son origine dans le domaine de la vision par ordinateur. En effet, en 2011 et 2012 [Ciresan et al. 2011; Krizhevsky et al. 2012] présentent DanNet et AlexNet deux architectures de réseaux de neurones basés sur des *Convolutional Neural Networks* – CNN – (que nous présenterons en Section 2.3.2) implémentés sur *Graphical Processing Unit* - GPU. Ces réseaux ont obtenu des performances très supérieures à ce qu'on pouvait trouver dans l'état de l'art. Bien que ces derniers soient des variantes du modèle de LeCun et al. [1989] daté déjà de 30 ans à l'époque, leurs performances ont été un signal important qui a contribué au retour sur le devant de la scène des CNN et plus généralement des réseaux de neurones.

**Réseau de convolution.** Les CNN sont composés notamment de filtres localisés entraîna- bles qui s'appliquent à chaque pixel et son voisinage. La succession de ces filtres permet d'extraire automatiquement des informations à différentes échelles de l'image, adaptées à la tâche d'apprentissage souhaitée (voir Section 2.3.3). Cette démarche automatique s'oppose à celle de la recherche de caractéristiques discriminantes « à la main », qui nécessite l'intervention d'un spécialiste. C'est une approche qui peut se révéler utile mais qui est fastidieuse et manque de

---

10. Que l'on appelle aussi réseaux de neurones récurrents.

généricité. Compte tenu de ce contexte, on comprend alors aisément la volonté d’extension des CNN aux graphes. S’il est très facile de définir des filtres sur les graphes attribués [Sandryhaila and Moura 2013; Ricaud et al. 2019], en utilisant notamment la transformée de Fourier pour graphe, il est plutôt difficile d’assurer leur localisation spatiale. Les solutions à cette problématique ont été apportées itérativement avec les contributions de Hammond et al. [2011], Bruna et al. [2014], Henaff et al. [2015], Defferrard et al. [2016] et finalement Kipf and Welling [2016c]. Les détails techniques de ces contributions pourront être retrouvés en Section 2.3.3. Elles ont permis de définir des analogues des CNN appelés réseaux de convolution sur graphe (*Graph Convolutional Network* - GCN). Cette percée importante dans l’histoire des GCN a permis d’améliorer les *benchmarks* de nombreuses tâches d’apprentissage sur graphes. Suite à ces succès, de nombreuses variantes de GCN ont notamment été développées, d’où leur présence importante dans la littérature ces dernières années. Plus généralement, des GNN ne s’appuyant pas sur la théorie spectrale des graphes ont également été développés de manière importante. On citera notamment des approches basées sur des mécanismes d’attention dont *Graph Attention Networks* – GAT - [Veličković et al. 2018] est le plus célèbre représentant ; ou encore des approches basées sur des mécanismes de récurrence comme *Gated Graph sequence Neural Networks* - CGNN [Li et al. 2015]. Cette littérature est vaste et ne peut être traitée de manière exhaustive ici ; une *review* complète de [Zhou et al. 2021] peut être consultée pour plus de détails. Dans ce manuscrit, nous nous sommes restreints à l’utilisation de quelques GCN, car des travaux récents [Wu et al. 2020] ont montré que la plupart des GNN atteignent des degrés de performances proches.

Malgré leurs succès, les GCN souffrent de certaines limites. Alors que les CNN classiques ont tendance à améliorer la qualité des caractéristiques extraites lorsque l’on augmente le nombre de couches cachées<sup>11</sup>, on observe une saturation voire une dégradation des performances des GCN lorsqu’on empile plus de 4 à 5 couches. Néanmoins en combinant et en intégrant les GCN au sein de modèles ou d’architectures particulières, on peut a priori surmonter ces difficultés. La problématique des limitations des GNN sera (re)discutée dans les chapitres 2 et 3.

### 1.3 Motivations de la thèse

Comme on a pu l’évoquer, les réseaux de neurones ont pris une place importante dans la littérature de l’apprentissage automatique et ont permis de grandes avancées. Malgré tout, de nombreuses tâches relativement triviales pour les données euclidiennes sont encore difficiles à faire sur les graphes. Il y a souvent tout un travail à effectuer avant de pouvoir transposer une méthode de l’un à l’autre<sup>12</sup>, il faut parfois inventer de tous nouveaux outils dans cette optique.

Le simple établissement d’une distance *rapide* entre les graphes est un problème difficile (même la détermination de l’égalité entre deux graphes est difficile - voir Section 2.1.2 -). Or, nombre d’algorithmes classiques d’apprentissage se basent sur des mesures de distances, on peut citer<sup>13</sup> entre autres la méthode des *k*-moyennes (*k-means*), un algorithme de *clustering*

---

11. A condition que le nombre de données soit suffisamment important et que la tâche soit suffisamment complexe.

12. En témoigne le temps qu’il aura fallu pour avoir des GCN fonctionnels.

13. Il s’agit des algorithmes canoniques pour ces tâches.

ou encore la méthode des  $k$  plus proches voisins, un algorithme de classification. Ces méthodes sont de fait difficilement utilisables pour faire du *clustering* ou de la classification de graphes. Ces limitations inhérentes à la nature des graphes ont motivé cette thèse. Dans ce manuscrit, nous présenterons 3 problématiques liées à l'apprentissage sur graphes et nous proposerons 3 solutions qui prendront la forme de modèles se basant en partie sur des GCN. Pour chacune de ces problématiques, nous aurons le soin de toujours proposer des méthodes avec des complexités algorithmiques raisonnables ; permettant ainsi leur utilisation sur des données de grandes tailles et dans un contexte réaliste.

Le Chapitre 2 de cette thèse introduira formellement les graphes, de la difficulté de l'apprentissage sur ces derniers et les réseaux de convolutions sur graphe. Les chapitres 3, 5 et 4 présenteront 3 problématiques et nos solutions. Ces chapitres seront relativement indépendants les uns des autres, ils s'appuieront en partie sur des notions définies dans le Chapitre 2, mais ces chapitres abordant des domaines relativement différents, un certain nombre de notions propres à leur problématique seront définies en leur sein.

- Dans le Chapitre 3, nous proposerons une méthode d'apprentissage de représentation de graphes, qui permet de plonger les graphes dans des espaces euclidiens et qui permet alors l'utilisation de méthodes classiques d'apprentissage sur ces représentations, en lieu et place de traiter directement avec le graphe. Les publications suivantes ont été faites sur ce travail : [Béthune et al. 2020a,b].
- Dans le Chapitre 4, nous proposerons une méthode d'apprentissage de représentation de nœuds pour les graphes portant des données multivues<sup>14</sup>. Il est courant que plusieurs caractéristiques soient définies sur les nœuds d'un graphe. Utiliser une seule de ces caractéristiques ou les concaténer n'est souvent pas optimal. Nous proposons donc une méthode de représentation euclidienne des nœuds multivues, ces représentations peuvent alors être utilisées par des méthodes classiques d'apprentissage pour faire de l'apprentissage sur les nœuds du graphes. Les publications suivantes ont été faites sur ce travail : [Kaloga et al. 2021a,b].
- Dans le Chapitre 5, nous proposerons une méthode d'apprentissage de métrique entre graphes permettant d'adapter la métrique à la tâche envisagée. Ce travail est disponible sous la forme d'un rapport technique [Kaloga et al. 2021c].
- Enfin, le Chapitre 6 conclura ce manuscrit et donnera quelques éléments de perspectives sur le sujet de l'apprentissage sur graphes.

L'Annexe A apportera des informations complémentaires sur les chapitres le précédant.

**Remarque 1 (*Discrete Mumford-Shah on Graph.*)** *Le travail suivant [Kaloga et al. 2019] qui a trait à la résolution d'un problème d'optimisation sur graphe a été finalisé puis publié pendant la thèse, mais il ne sera pas présenté ici. L'objectif dans cet article est de déterminer les pourcentages de report de votes entre deux élections (notamment entre les deux tours de la présidentielle française) dans chaque bureau de vote. Il existe une infinité de solutions de report possible. Pour surmonter ce problème il est nécessaire d'introduire des informations supplémentaires. Dans cet article, on fait l'hypothèse que les pourcentages de report sont des fonctions géographiquement lisses, c'est-à-dire que les bureaux de vote proches auront des comportements*

14. C'est-à-dire des graphes dont les nœuds possèdent plusieurs caractéristiques.

---

*de report proches. On construit alors un graphe dont les nœuds sont les bureaux de votes et les arêtes leurs plus proches voisins. L'article définit alors une fonctionnelle à partir de ce graphe pour résoudre le problème des reports de voix et propose deux algorithmes pour la minimiser.*





## Chapitre 2

# Généralités sur les graphes

*Le traitement du signal est une discipline aux confins de la physique et de l'informatique. Cette thèse s'inscrivant en partie dans cette discipline, il est utile de réintroduire des éléments de bases provenant à la fois de l'informatique comme la théorie classique des graphes et de la physique comme la transformée de Fourier sur graphes pour pouvoir être lu plus largement par les deux communautés. C'est pourquoi ce chapitre sera consacré à une introduction formelle des graphes et de quelques-unes de leurs propriétés. Nous présenterons dans un premier temps quelques notions très générales issues de théorie des graphes, qui d'une part nous permettront de les caractériser et d'autre part d'apprécier la difficulté de la tâche de comparaison entre graphes. Dans un second temps, nous prendrons le point de vue de la théorie spectrale des graphes, ce qui sera l'occasion d'introduire de nouveaux outils permettant à la fois de caractériser la structure d'un graphe mais également les signaux sur ces derniers. Cette partie permettra de construire le cadre théorique nécessaire à la construction des GCN qui sera abordé en dernière partie. Dans la partie finale de ce chapitre, nous aborderons plus en détails les questions d'apprentissage sur graphes, notamment au travers des GCN qui sont au cœur des travaux de cette thèse.*

### 2.1 Notions de théorie des graphes

Les graphes sont des objets récurrents dans de nombreux domaines, cela s'explique par le caractère général de ces objets. Dès lors qu'un modèle ou un système est composé d'éléments discrets qui interagissent d'une manière ou d'une autre, on peut souvent le décrire par un graphe. Les graphes sont donc des représentations abstraites d'une situation qu'on peut très couramment retrouver. Cependant, ce caractère général n'est pas sans prix, il s'accompagne de nombreuses problématiques s'exprimant très simplement mais qui s'avèrent être d'une redoutable difficulté. Malgré tout, ces difficultés théoriques sont des défis particulièrement intéressants car les surmonter c'est résoudre des problèmes issus de plusieurs disciplines simultanément avec de nombreuses applications à la clé. Un exemple d'un tel problème est le problème du voyageur de commerce. Dans ce problème, il est question de déterminer l'itinéraire le plus court (pour un représentant de commerce) pour visiter une liste de villes (les sommets) reliées par des routes (les arêtes) de distance fixée. C'est un des problèmes d'algorithmie les plus célèbres et les plus étudiés, les avancées sur ce problème ont des implications non seulement en informatique théorique, mais

également sur des applications très concrètes en logistique ou encore en génétique [Tang and Chilkoti 2016].

Dans cette section, nous allons formellement introduire les graphes et nous répondrons essentiellement aux questions : *Qu'est-ce qu'un graphe ? Et comment se caractérise-il ?* Pour aller plus loin, on pourra notamment se référer à [West et al. 2001 ; Diestel 2010].

### 2.1.1 Graphe

Un graphe au sens classique du terme décrit les relations entre un certain nombre d'éléments. Formellement, c'est la donnée d'une paire d'ensemble  $\mathcal{G} = (V, E)$ , tel que  $E \subseteq V \times V$  ; les éléments de  $E$  sont des paires d'éléments de  $V$ . Les éléments de  $V$  sont les sommets (ou les nœuds) du graphe. La taille du graphe correspond au nombre de ses sommets  $|V|$  qu'on notera généralement  $n$ . Les éléments de  $E$  sont les arêtes (ou les liens) du graphe. Lorsque deux nœuds sont portés par une même arête, on dit qu'ils sont adjacents.

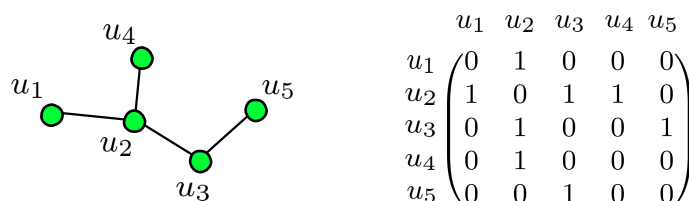


FIGURE 2.1 – **Représentation graphique et matricielle.** **A gauche** une représentation graphique d'un graphe, constitué des nœuds  $V = \{u_1, u_2, u_3, u_4, u_5\}$  et des arêtes  $E = \{\{u_1, u_2\}, \{u_2, u_3\}, \{u_2, u_4\}, \{u_3, u_5\}\}$ . **A droite** une représentation matricielle de ce même graphe, c'est la matrice d'adjacence. Cette représentation n'est pas unique et est définie à une permutation près des nœuds sur les lignes et les colonnes.

Il est usuel de représenter le graphe par sa matrice d'adjacence  $\mathbf{A}$  qui en considérant que  $V = \{u_1, \dots, u_n\}$ , se définit telle que :

$$\mathbf{A}_{i,j} = \begin{cases} 1 & \text{si } \{u_i, u_j\} \in E \\ 0 & \text{sinon.} \end{cases} \quad (2.1)$$

On remarquera que la matrice d'adjacence est symétrique<sup>1</sup>,  $\mathbf{A}_{i,j} = \mathbf{A}_{j,i}$ . Cette représentation est très commode, elle permet d'utiliser notamment les puissants outils de l'algèbre linéaire qui sont très utiles pour l'étude de ces structures. La matrice d'adjacence permet également d'exprimer un certain nombre de grandeurs utiles relatives aux graphes. On peut notamment exprimer le degré  $d_i$  d'un nœud  $u_i \in V$  qui correspond au nombre de nœuds dans le graphe connectés à ce dernier avec la matrice d'adjacence :

$$d_i = \sum_{u_j \in V} \mathbf{A}_{i,j}. \quad (2.2)$$

1. Lorsque le graphe est orienté, c'est à dire que les arêtes sont orientées, la matrice d'adjacence n'est pas symétrique.

De même pour le volume d'un graphe qui correspond à la somme des degrés de ses nœuds :

$$\text{vol } \mathcal{G} = \sum_{u_i, u_j \in V} \mathbf{A}_{i,j}. \quad (2.3)$$

Le voisinage *immédiat* d'un nœud  $u_i$  est noté  $\mathcal{V}(u_i)$  et peut être également défini via la matrice d'adjacence :

$$\mathcal{V}(u_i) = \{u_j \in \mathcal{V} | \mathbf{A}_{i,j} > 0\} \cup \{u_i\}. \quad (2.4)$$

On dit d'un graphe qu'il est *connexe* si quelle que soit la paire  $\{u_i, u_j\}$  de ses sommets choisie dans  $V \times V$ , il existe toujours une chaîne d'arêtes permettant de se déplacer de l'un à l'autre.

$$\forall u_i, u_j \in V, \exists p \in \mathbb{N}, v_1, v_2, \dots, v_p \in V, \text{ tels que } \{u_i, v_1\}, \{v_1, v_2\}, \dots, \{v_p, u_j\} \in E \quad (2.5)$$

Le graphe est alors d'*un seul tenant*. De manière générale, l'essentiel des graphes étudiés sont connexes, car l'étude des graphes non connexes se ramène très souvent à l'étude de ses composantes connexes. Comme les graphes sont des objets *finis*, il existe alors toujours un plus petit chemin (en le nombre d'arêtes) entre deux nœuds  $u_i$  et  $u_j$  d'un graphe connexe. Ce nombre est noté  $d_{i,j}^{\mathbf{A}}$  et est compris entre 1 et  $n - 1$ .

**Remarque 2 (Plus court chemin.)** *Les puissances de la matrice d'adjacence permettent de déterminer ces plus courts chemins. En effet, pour  $p \in \llbracket 1, n - 1 \rrbracket$ ,  $\mathbf{A}_{i,j}^p$  est égale au nombre de chemins de longueur  $p$  entre  $u$  et  $v$ . Ainsi,*

$$d_{i,j}^{\mathbf{A}} = \min_p \{p \in \llbracket 1, n - 1 \rrbracket | \mathbf{A}_{i,j}^p > 0\}. \quad (2.6)$$

On ne considérera que des graphes sans boucle, c'est-à-dire  $\forall u_i \in V, \{u_i, u_i\} \notin E$  alors  $d_{i,i}^{\mathbf{A}} = 0$ . Le diamètre d'un graphe est le plus grand des plus courts chemins qu'il contient :

$$D(\mathcal{G}) = \max_{u_i, u_j \in V} d_{i,j}^{\mathbf{A}}. \quad (2.7)$$

Avec la notion de plus court chemin, on peut étendre la notion de voisinage dans les graphes connexes<sup>2</sup>, on notera  $\mathcal{V}_p(u)$  l'ensemble des nœuds à une distance inférieure à  $p$  de  $u$  :

$$\mathcal{V}_p(u_i) = \{u_j \in \mathcal{V} | d_{i,j}^{\mathbf{A}} \leq p\}. \quad (2.8)$$

On notera qu'avec cette définition,  $u_i \in \mathcal{V}_p(u_i)$  sans avoir à recourir à l'opérateur d'union.

---

2. Évidemment pour les graphes non connexes, s'il n'existe pas de chemin entre deux nœuds, il suffit de fixer la valeur de la distance entre ces derniers à  $+\infty$  pour que la définition reste valide. Ce voisinage étendu peut également être défini par récursivité avec le voisinage *immédiat*.

### 2.1.1.a Structures particulières

A partir de cette définition très générale des graphes, on peut distinguer certaines structures particulières que nous serons amenés à rencontrer dans ce manuscrit :

- **Cycles.** Un cycle est un chemin qui se referme sur lui-même dans le graphe. Les graphes qui sont constitués d'un seul et unique cycle sont des **graphes cycliques**.
- **Arbres.** Les arbres sont des graphes connexes (2.5) dans lesquels pour toute paire de nœuds, il y a un et unique chemin qui permet de se déplacer de l'un à l'autre. Cette condition équivaut à dire qu'il n'y pas de cycle dans le graphe :

$$\forall u \in V, \nexists p \in \mathbb{N}, u_1, u_2, \dots, u_p \in V, \text{ tels que } \{u, u_1\}, \{u_1, u_2\}, \dots, \{u_p, u\} \in E \quad (2.9)$$

Un cas particulier intéressant d'arbres sont les arbres enracinés : il s'agit d'arbre dans lesquels on a désigné un nœud en particulier comme étant la racine, l'objet ainsi formé ressemble alors à un *arbre* tel qu'on peut le considérer dans la nature ou dans d'autres domaines comme la généalogie ou la phylogénétique. Ces arbres sont très intéressants car ils peuvent permettre dans certaines situations de fournir un outil complémentaire de caractérisation de graphes plus complexes (une illustration pourra être trouvée dans la Section 3.2.1 sur Weisfeiler-Lehman).

- **Régulier.** Une graphe régulier est un graphe dont l'ensemble des nœuds possèdent le même degré. C'est un cas particulier important car ils constituent une passerelle entre les graphes et les objets euclidiens ; les objets euclidiens peuvent notamment être décrits par certains de ces graphes.
- **Roue.** Un graphe roue est formé en prenant un graphe cyclique et en y ajoutant un nœud auquel on relie tous les nœuds du cycle.
- **Échelle.** Un graphe échelle est un graphe formé de deux chaînes de nœuds appariées tel que :

$$\begin{aligned} \exists p \in \mathbb{N}, V = \{u_1, u_2, \dots, u_p, v_1, v_2, \dots, v_p\}, \text{ tels que } & \{u_1, u_2\}, \{u_2, u_3\}, \dots, \{u_{p-1}, u_p\} \in E \\ & \{v_1, v_2\}, \{v_2, v_3\}, \dots, \{v_{p-1}, v_p\} \in E \\ & \{u_1, v_1\}, \{u_2, v_2\}, \dots, \{u_p, v_p\} \in E \end{aligned} \quad (2.10)$$

- **Planaire.** Un graphe planaire est un graphe qu'on peut représenter dans le plan, sans que ses arêtes ne se croisent. Les molécules chimiques peuvent souvent être représentées par ce type de graphe.

### 2.1.1.b Graphes attribués et pondérés

Les graphes attribués sont des extensions naturelles de la notion classique de graphe, qui permettent d'étendre leurs champs d'application. Dans ces graphes, les nœuds et les arêtes sont porteurs d'informations. Ils sont donc définis par le quadruplet  $(\mathcal{G}, x, w) = (V, E, x, w)$  :

$$\begin{aligned} x &: V \rightarrow \mathbb{R}^q \\ u_i &\mapsto x(u_i) \end{aligned} \tag{2.11}$$

où  $x$  est une fonction qui à chaque sommet associe un vecteur de l'espace euclidien  $\mathbb{R}^q$ . Lorsque  $q = 1$  on pourra notamment identifier  $x \in \mathbb{R}^V \simeq \mathbb{R}^n$  à un vecteur de  $\mathbb{R}^n$ , dans ce cas-là  $x$  sera noté  $\mathbf{x}$  et

$$\forall u_i \in V, x_i = x(u_i).$$

Lorsque  $q > 1$  on pourra également à l'occasion identifier  $x \in \mathbb{R}^V \simeq \mathbb{R}^{n \times q}$ <sup>3</sup> à une matrice de  $\mathbb{R}^{n \times q}$ , dans ce cas-là  $x$  sera noté  $\mathbf{X}$ , et

$$\forall u_i \in V, \mathbf{X}(i, :) = x(u_i)^T.$$

Où  $\mathbf{X}(i, :)$  désigne la  $i$ -ième ligne de  $\mathbf{X}$ . De même :

$$\begin{aligned} w &: E \rightarrow \mathbb{R}^{q_w} \\ \{u_i, u_j\} &\mapsto w(\{u_i, u_j\}) \end{aligned} \tag{2.12}$$

où  $w : E \rightarrow \mathbb{R}^{q_w}$  est une fonction qui à chaque arête associe un vecteur de l'espace euclidien quelconque  $\mathbb{R}^{q_w}$ . Dans ce manuscrit, nous ne considérerons que le cas particulier où  $w$  est à valeur dans  $\mathbb{R}_+$ . Dans ce cas, l'interprétation de  $w$  est simple, il s'agit d'une mesure de l'importance des liens qui unissent les sommets. Dans ce cadre, les graphes sont dits pondérés. Dans la suite, les graphes sont supposés non pondérés et nous préciserons explicitement dans le texte lorsqu'ils le seront ; ainsi la notation  $w$  sera toujours omise par souci de simplification. L'ensemble d'arrivée de  $x$  restera lui quelconque (sauf si précisé) et pourra désigner selon le contexte une quantité numérique, vectorielle ou même tensorielle. Dans le Chapitre 3, on abordera en particulier le cas où des graphes sont attribués par plusieurs fonctions  $x$ .

Les différentes définitions introduites plus haut, restent valides pour les graphes pondérés à condition de modifier la définition de la matrice d'adjacence de ces graphes ainsi :

$$\mathbf{A}_{i,j} = \begin{cases} w(\{u_i, u_j\}) & \text{si } \{u, v\} \in E \\ 0 & \text{sinon.} \end{cases} \tag{2.13}$$

L'intensité des liens entre les sommets des graphes pondérés est donc directement accessible dans la matrice d'adjacence.

**Remarque 3 (Plus court chemin des graphes pondérés.)** *Usuellement, la notion de plus court chemin dans un graphe pondéré, tient compte de la valeur des poids présents sur les nœuds. La définition (Eq. (2.6)) ne dépend que de la nullité ou non de ces poids : elle compte seulement le nombre d'arêtes (de poids strictement positif) qu'il faut traverser indépendamment de leur*

---

3. La notation est impropre, ces deux espaces ne sont pas isomorphes,  $\mathbb{R}^{q \times n}$  est isomorphe à l'ensemble des applications linéaires dans  $\mathbb{R}^{qV}$ .

valeurs. Cette définition permet de définir la même notion de voisinage (Eq. (2.8)) dans les graphes pondérés. Mais elle diffère toutefois de la définition communément adoptée pour ces graphes.

**Remarque 4 (Graphes pondérés.)** Dans des situations réelles, il est souvent judicieux de modéliser la situation par un graphe pondéré. Toutefois, il existe bien des situations dans lesquelles la restriction de l'image de  $w$  à  $\mathbb{R}_+$  est limitante. En chimie notamment, les différentes liaisons chimiques (simple, double, triple, hydrogène, etc.) caractérisent bien plus qu'une simple force de liaison, ces différentes liaisons ont des impacts directs sur certaines propriétés de la molécule, comme sa stabilité par exemple. Toutefois nous n'utiliserons pas ce type d'information dans notre travail.

### 2.1.1.c Relations entre graphes

**Égalité.** On peut définir un certain nombre de relations entre les structures de graphes, la plus immédiate d'entre elles est l'égalité. Deux graphes  $\mathcal{G} = (V, E)$  et  $\mathcal{G}' = (V', E')$  sont dits égaux s'ils sont structurellement équivalents, on dit alors qu'ils sont isomorphes. Dans ce cas, il existe une permutation des nœuds  $\sigma \in \mathcal{S}_n$  qui met en correspondance les éléments de  $V$  à  $V'$  et de  $E$  à  $E'$  :

$$\begin{aligned} \forall u \in V, \exists ! u' \in V, u = \sigma^{-1}(u') \\ \forall u, v \in V, \{\sigma(u), \sigma(v)\} \in E' \Leftrightarrow \{u, v\} \in E. \end{aligned} \quad (2.14)$$

**Inclusion.** S'ils ne sont pas égaux,  $\mathcal{G}$  peut éventuellement contenir  $\mathcal{G}'$ , on dira alors que  $\mathcal{G}'$  est un **sous-graphe** de  $\mathcal{G}$ . Il existe alors une permutation  $\sigma \in \mathcal{S}_{n'}$ ,  $n' \leq n$  qui met en correspondance certains nœuds de  $\mathcal{G}$  à l'ensemble des nœuds de  $\mathcal{G}'$  en préservant leurs structures :

$$\begin{aligned} \forall u \in V, \exists ! u' \in V, u = \sigma^{-1}(u') \\ \forall u, v \in V, \{\sigma(u), \sigma(v)\} \in E' \Rightarrow \{u, v\} \in E. \end{aligned} \quad (2.15)$$

On dira de  $\mathcal{G}'$  que c'est un **sous-graphe induit**, lorsque  $\sigma^{-1}$  préserve également la structure du sous-graphe :

$$\forall u, v \in V, \{u, v\} \in E \Rightarrow \{\sigma(u), \sigma(v)\} \in E'. \quad (2.16)$$

La relation d'inclusion de graphe traduit une relation d'isomorphie entre un graphe et une partie d'un autre graphe. Lorsque le sous-graphe a une structure particulière, on pourra le qualifier de sous-(*nom de la structure*), par exemple en Figure 2.2 les sous-graphes sont manifestement des arbres, on pourra alors les qualifier de sous-arbres. La relation d'inclusion de graphe est intéressante car un graphe peut être caractérisé (du moins partiellement) par certains de ses sous-graphes (voir Section 3.2.1) généralement plus *simples* que le graphe lui-même [?Narayanan et al. 2017; Taheri 2018].

Néanmoins, c'est une notion qui reste relativement limitée en elle-même pour une comparaison *directe* de graphes, puisque la relation d'inclusion dans un sens ou dans l'autre n'est presque jamais vérifiée pour deux graphes quelconques. On peut toutefois se servir de cette relation pour construire d'autres notions plus abouties de comparaison de graphes (voir Section 2.1.2).

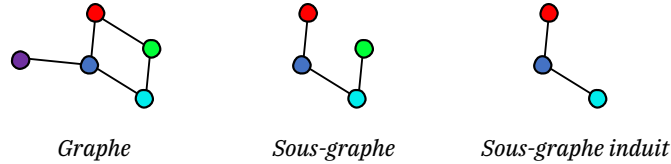


FIGURE 2.2 – *Un graphe et deux de ses sous-graphes.* Le premier sous-graphe n'est pas induit car il manque le lien entre le nœud rouge et le nœud vert qui existe dans le graphe à gauche.

**Homomorphie.** Un autre type de relation plus *souple* peut être définie grâce à la notion fonctionnelle d'homomorphisme. On dira que  $\mathcal{G}$  est **homomorphe** à  $\mathcal{G}'$  s'il existe une application *quelconque*  $h$  de  $V$  dans  $V'$  qui préserve la structure de  $\mathcal{G}$  :

$$\{u, v\} \in V \Rightarrow \{h(u), h(v)\} \in V'. \quad (2.17)$$

L'application  $h$  est appelée un homomorphisme. Cette relation n'est pas *nécessairement* symétrique, puisque les nœuds de  $\mathcal{G}'$  peuvent avoir des arêtes que leurs antécédents par  $h$  n'ont pas dans  $\mathcal{G}$  (voir Fig. (2.3) - à gauche) ; mais lorsque  $\mathcal{G}'$  est également homomorphe à  $\mathcal{G}$  on dit que ces deux graphes sont homomorphiquement équivalents (voir Fig. (2.3) - au milieu). On remarquera que les sous-graphes de  $\mathcal{G}$  sont toujours homomorphes à  $\mathcal{G}$  alors que les sous-graphes induits sont eux homomorphiquement équivalents. La notion d'homomorphie est très intéressante, c'est notamment dans ce cadre que la notion de *pooling* bien connue pour les images se développe pour les graphes comme on le verra en Section 3.3.1. Par ailleurs, si  $\mathcal{G}$  est homomorphe à  $\mathcal{G}'$  par une fonction  $h$  injective, surjective et dont la réciproque est encore une fonction homomorphe alors  $h$  est une permutation de  $\mathcal{S}_n$  et  $\mathcal{G}$  et  $\mathcal{G}'$  sont isomorphes.

Pour conclure, la notion d'isomorphie traduit l'égalité entre deux graphes, celle de sous-graphe traduit l'inclusion tandis que la relation d'homomorphie traduit une *certaine ressemblance* entre graphes. Alors que ces relations sont relativement simples et assez peu informatives des différences structurelles entre graphes, l'évaluation de ces relations forme des problèmes difficiles au sens de la théorie de la complexité informatique (voir Annexe A.1.1 pour des rappels sur la complexité des algorithmes).

**Remarque 5 (Isomorphie des graphes attribués.)** *Le problème d'isomorphisme de graphes s'étend facilement aux graphes attribués  $\mathcal{G}$  et  $\mathcal{G}'$  associés aux fonctions  $x$  et  $x'$  ; il suffit d'ajouter la condition suivante à la fonction  $h$  :  $\forall u \in V, x(u) = x'(h(u))$*



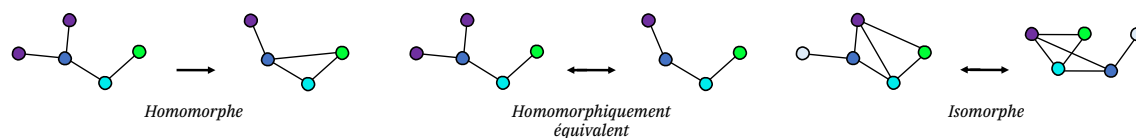


FIGURE 2.3 – **Homomorphie et isomorphie.** **A gauche**, les nœuds d’une certaine couleur du graphe de gauche sont envoyés sur le nœud de la couleur correspondante à droite. Cette opération est un homomorphisme de graphe. Le graphe de gauche est donc homomorphe au graphe de droite. À l’inverse, on ne peut pas trouver d’homomorphisme du graphe de droite vers le graphe de gauche à cause de l’absence du cycle dans le graphe de gauche présent à droite. **Au milieu**, la relation entre les nœuds de même couleur est un morphisme de gauche à droite et de droite à gauche, ces deux graphes sont donc homomorphiquement équivalents, mais ils ne sont pas isomorphes. **À droite**, le morphisme décrit est *injectif, surjectif* et sa réciproque est aussi un homomorphisme c’est donc un isomorphisme, les deux graphes sont identiques.

## 2.1.2 isomorphisme de graphes

Si la question de savoir si deux graphes sont identiques ou même partiellement identiques paraît simple à première vue, c’est en fait une question difficile. Tout d’abord, remarquons que le problème d’isomorphisme de sous-graphe (qui consiste à déterminer si un graphe est un sous-graphe d’un autre) est au moins aussi difficile que le problème d’isomorphisme de graphes. En effet, deux graphes isomorphes sont des sous-graphes l’un de l’autre et donc résoudre le problème d’isomorphisme partiel c’est résoudre le problème d’isomorphisme. En pratique, le problème d’isomorphisme partiel a été démontré comme étant NP-Complet (voir Annexe A.1.1), c’est à dire qu’il est au moins aussi difficile que tous les problèmes de la classe NP. La classe NP désignant les problèmes pour lesquels il n’existe pas d’algorithme (pour le moment) capable de les résoudre en un temps polynomial mais dont on est capable de vérifier la validité d’une solution en un temps polynomial. Évidemment, étant donné un appariement de nœuds entre deux graphes il est facile de vérifier que ce problème traduit bien une relation d’inclusion, mais il est beaucoup moins évident a priori que ce problème soit *difficile*.

Quant au problème d’isomorphisme de graphes, il a été l’objet d’un très grand nombre de travaux de part les applications qui en découlent (ex : indexation, recherche et comparaison de molécules chimiques dans une base données) mais aussi parce qu’il est un des problèmes les plus importants de la théorie de la complexité. Il s’agit d’un des rares problèmes de la classe NP, dont l’appartenance à la classe des algorithmes P ou NP-complet n’a pas encore été déterminée. À ce titre, c’est un problème qui fait et a fait l’objet de nombreuses recherches et reste cependant irrésolu. Toutefois, il existe des algorithmes de résolution polynomiaux de ce problème pour certaines classes de graphes, notamment pour les graphes de degrés bornés, les arbres ou encore les graphes planaires, etc. d’où (en partie) l’intérêt de la caractérisation des graphes par des sous-graphes plus simples et plus faciles à comparer.

**Remarque 6 (isomorphisme de graphes.)** *Ce problème a connu une importante percée en 2015, où Babai [2015] a déterminé un algorithme quasi-linéaire c’est à dire dont la complexité est en  $\exp(\log(n)^{O(1)})$ , qui consiste en une juxtaposition astucieuse de différents certificats de non-*

isomorphisme (nous aurons à travailler avec l'un d'entre eux en Section 3.2.1) qui permettent de décider si deux graphes sont isomorphes en un temps raisonnable, du moins au sens de la complexité.

Malgré leurs complexités, les problèmes d'isomorphisme total ou partiel sont relativement peu informatifs sur les différences structurelles. Toutefois il est possible de construire des mesures de similarités plus fines, comme on peut le voir dans l'exemple ci-dessous.

**Exemple 1 (Distances à partir des relations d'inclusion entre graphes.)** *Considérons deux graphes  $\mathcal{G}$  et  $\mathcal{G}'$  de tailles strictement supérieures à 1. Ces deux graphes ont un sous-graphe de taille maximale en commun (maximum common subgraph - mcs) qu'on notera  $\mathcal{G}_{mcs} = (V_{mcs}, E_{mcs})$ . Ces deux graphes sont eux-mêmes des sous-graphes de certains graphes plus grands qu'eux, considérons le plus petit d'entre eux qu'on notera  $\mathcal{G}_{big} = (V_{big}, E_{big})$ . Alors on peut définir la notion de distance suivante [Fernandez and Valiente 2001] :*

$$d(\mathcal{G}, \mathcal{G}') = n_{big} - n_{mcs}$$

Où  $n_{mcs} = |V_{mcs}|$  et  $n_{big} = |V_{big}|$ . Bien entendu, d'autres choix peuvent être faits, on pourrait par exemple évaluer la similarité entre les graphes via cette quantité [Bunke and Shearer 1998] :

$$d(\mathcal{G}, \mathcal{G}') = 1 - \frac{n_{mcs}}{\max(n, n')}$$

Hélas, ces distances héritent de la complexité des notions sur laquelle elles se reposent. Dans ces cas, le problème de la détermination de  $\mathcal{G}_{mcs}$  et  $\mathcal{G}_{big}$  (il s'agit de problèmes équivalents) est NP-Complet [Conte et al. 2007].

Cet exemple est représentatif de la difficulté de l'établissement de métriques sur les graphes, elles sont aussi souvent fastidieuses à calculer. Or, c'est une tâche récurrente en apprentissage que d'évaluer des distances entre les objets étudiés, il est donc crucial que cette opération soit *rapide*, particulièrement de nos jours, où les jeux de données sont de plus en plus volumineux. Pour répondre à cette problématique, au fil des ans, les différentes communautés scientifiques travaillant avec des graphes ont développé des heuristiques relativement simples à calculer mais très informatives sur certains aspects de ces graphes, et permettant notamment d'évaluer leurs similarités par rapport à ces dernières.

### 2.1.3 Heuristiques, noyaux et invariants

**Heuristiques.** Les heuristiques sont des méthodes qui fournissent simplement et rapidement un résultat ou une solution à un problème. Une heuristique abandonne souvent l'exactitude ou l'optimalité de la solution, au profit de la vitesse d'exécution tout en fournissant une solution de bonne qualité. Ainsi, face aux difficultés évoquées plus haut, de nombreuses méthodes d'évaluation heuristiques de similarité entre graphes ont été proposées. Chacune de ces heuristiques propose une manière de caractériser le graphe de manière relativement efficace, c'est-à-dire rapidement et de manière suffisamment discriminative. Néanmoins, ces heuristiques ont leurs limites : il s'agit en général de pseudo-distance (voir Section 5.2), et elles sont souvent adaptées

à la réalisation de certaines tâches et manquent donc de généralité. C'est tout à fait attendu, certaines tâches sont plus à même d'être impactées par telles ou telles propriétés des graphes, propriétés qui seront plus ou moins bien appréciées selon l'heuristique choisie. Au même titre qu'il existe des métriques différentes dans les espaces euclidiens qui sont plus ou moins pertinentes selon le problème que l'on souhaite traiter.

**Noyaux.** Ces heuristiques permettent de construire des similarités et des noyaux adaptés, permettant notamment de faire de la classification de graphe. Les premiers noyaux *entre* graphes ont été introduits par Gärtner et al. [2003] et Kashima et al. [2003] depuis, de nombreux auteurs ont proposé des noyaux sur graphes permettant d'enrichir et d'élargir l'horizon des tâches faisables sur les graphes. On peut notamment désigner 2 catégories importantes :

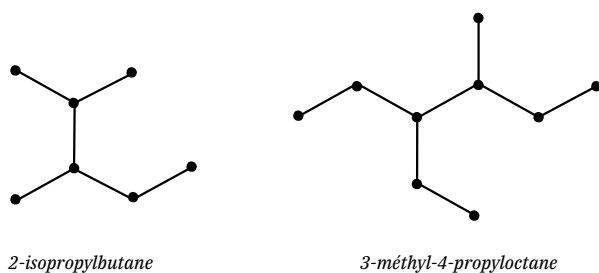
- Une première catégorie de noyaux caractérise les graphes via des heuristiques listant les structures qu'ils contiennent. Certaines heuristiques analysent les chemins que les graphes contiennent, notamment via des processus de marche aléatoire ou en s'intéressant aux plus courts chemins entre les nœuds [Borgwardt et al. 2005] ; d'autres caractérisent les graphes via les sous-arbres qu'ils contiennent, on peut notamment citer le très important Weisfeiler-Lehman (voir Section 3.2.1), qui est une référence dans la littérature de noyaux sur graphes ; les cycles, les cliques, ou encore les sous-graphes induits, de nombreuses autres structures peuvent être utilisées à cet effet.
- Une deuxième grande catégorie correspond aux noyaux se basant sur des modèles. Il s'agit essentiellement de construire un modèle sur un graphe et d'établir une mesure de similarités sur la base des résultats de ce modèle sur chacun des graphes considérés. L'exemple canonique de ce type de noyau est le noyau de la chaleur [Tsitsulin et al. 2018].

La liste ci-dessus n'est évidemment pas exhaustive mais englobe les méthodes les plus couramment rencontrées : une *review* récente et très complète de ces méthodes peut-être trouvée dans l'article de Ghosh et al. [2018].

**Invariants.** Une autre famille d'heuristiques consiste à calculer des invariants, pour mesurer des similarités ou dissimilarités entre graphes. Les invariants sont des quantités qui sont identiques lorsque deux graphes sont isomorphes (la réciproque est souvent fautive). De tels invariants sont très courants en théorie des graphes et peuvent s'avérer très utiles notamment pour des tâches d'apprentissage (voir exemple ci-dessous).

**Exemple 2 (Indices topologiques.)** *En chimie, les molécules peuvent être avantageusement décrites par des graphes. En effet, certaines de leurs propriétés sont décrites et prédites par des notions de théorie des graphes. En particulier, il existe un certain nombre d'invariants appelés indices topologiques (topological index) décrivant la structure de ces molécules en tant que graphes et qui sont directement liés à leurs propriétés chimiques. L'un des exemples parmi les plus simples de ce type de relation lie l'indice de Wiener, que nous rappelons ci-dessous, à certaines propriétés des alcanes. Les alcanes sont des composés chimiques qui sont exclusivement composés de carbone et d'hydrogène, ils peuvent être décrits par le graphe  $G_a$  de leur carbone, où les sommets sont les carbones et les arêtes les liens entre ces carbones.*

*L'indice de Wiener est un invariant topologique introduit par le chimiste Harold Wiener en 1947, et qui correspond à la somme des plus courtes distances dans le graphe :*

FIGURE 2.4 – *Exemples d’alcanes.*

$$W(\mathcal{G}_a) = \frac{1}{2} \sum_{u_i, u_j \in V} d_{i,j}^{A_a} \quad (2.18)$$

Il se trouve que cette quantité permet d’approximer la température d’ébullition des alcane  $T_e(a)$  (en Kelvin) [M. Yamuna 2018] :

$$T_e(a) \approx 181 \cdot W(\mathcal{G}_a)^{0.1771} \quad (2.19)$$

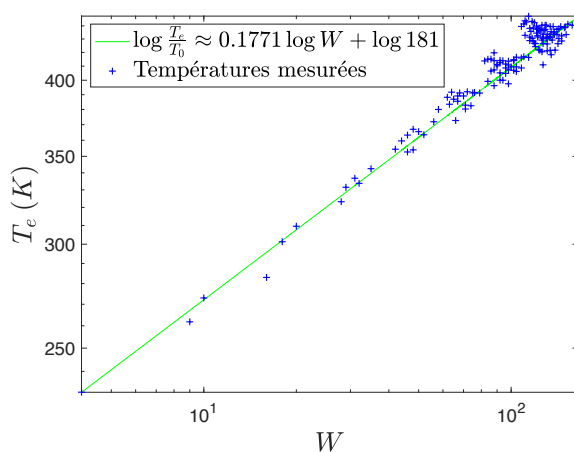


FIGURE 2.5 – *Régression linéaire de la température d’ébullition des alcane en fonction de leur indice de Wiener.*  $T_0 = 1K$ . Les points bleus correspondent aux températures d’ébullition des 148 plus petits alcane avec leurs indices de Wiener en abscisse.

Cette approximation peut être raffinée en introduisant d’autres invariants [Nikolić and Trinajstić 1995] et généralisée à d’autres hydrocarbures [Brezovnik et al. 2021]. Cet indice peut également être relié à d’autres propriétés chimiques comme la densité, la viscosité ou encore la tension de surface de ces molécules.

L’indice de Wiener est un exemple historiquement important, c’est le tout premier invariant issu de la théorie des graphes introduit en chimie. Depuis son introduction, bien d’autres

*invariants pertinents ont été introduits : l'indice de Szeged qui est une généralisation de l'indice de Wiener, l'indice hyper-Wiener qui en est une autre généralisation, l'indice d'Hosoya, etc.*

Comme on peut le voir, il existe de nombreuses manières de caractériser un graphe, le choix de la méthode est souvent fonction de la tâche envisagée. Toutefois, cette profusion de choix nécessite l'intervention d'un expert pour construire la solution. C'est une forte contrainte alors que ces dernières années les résultats des RNN (puis transformeurs) en traitement du langage naturel et des CNN pour les images, ont montré que les méthodes automatiques basées sur des réseaux de neurones pouvaient se révéler très efficaces en particulier lorsqu'un grand nombre de données est disponible. Naturellement, la communauté de l'apprentissage automatique a voulu construire des analogues de ces réseaux de neurones pour les graphes, pour d'une part s'affranchir des limites liées à la nécessité de recourir à un expert et d'autre part, bénéficier des performances spectaculaires des réseaux de neurones. Ainsi, au début des années 2010, de nombreux auteurs ont tenté de construire des équivalents des CNN pour les graphes. Cette recherche a donné naissance aux GCN et à de nombreuses autres familles de réseaux de neurones sur graphes. Pour comprendre comment cette généralisation a pu s'opérer, il est nécessaire de s'intéresser à la théorie spectrale des graphes, qui généralise certaines notions issues du traitement du signal sur des données globalement structurées dont notamment la transformée de Fourier et les filtres linéaires qui sont les principaux outils sur lesquels se basent les GCN.

## 2.2 Analyse spectrale

Les définitions de cette partie sont basées sur les sources suivantes [Chung and Graham 1997; Ricaud et al. 2019; Hammond et al. 2011]. La théorie spectrale des graphes introduite au début des années 80 a connu de nombreux développements ces 30 dernières années et a également apporté de nombreux apports à la théorie des graphes. Les valeurs propres associées aux graphes au centre de cette théorie sont directement liées aux invariants topologiques de ces derniers et sont à ce titre des grandeurs caractéristiques de leur topologie. Ces valeurs propres trouvent leurs utilités aussi bien en chimie où elles sont associées à des questions de stabilité de molécules, qu'en physique où elles s'interprètent comme des fréquences (aux carrés) et permettent l'analyse harmonique de signaux sur graphes. La théorie spectrale des graphes établit une surprenante connexion entre la théorie des graphes et la géométrie différentielle, permettant l'apport des outils de cette discipline dans le domaine discret ; ce lien est d'ailleurs activement exploité par certains auteurs pour construire des noyaux et des réseaux de neurones sur graphes [Monti et al. 2016]. Dans cette partie, nous présenterons les notions de base de la théorie spectrale des graphes, et quelques résultats attestant du lien entre la topologie des graphes et leurs spectres ainsi que sur l'interprétation du spectre comme des fréquences. Munis de cette base théorique, nous présenterons l'analogue de la transformée de Fourier sur graphe et les filtres linéaires sur graphe, sur lesquels se basent les GCN.

### 2.2.1 Laplacien, spectre et harmonique

Soit un graphe  $\mathcal{G} = (V, E)$ . Dans le cadre de la théorie spectrale des graphes, on considère des graphes attribués. La fonction  $x : V \rightarrow \mathbb{R}^q$  est interprétée comme un signal dont le support est le graphe  $\mathcal{G}$ .

Dans un premier temps, on considérera que  $q = 1$ . La théorie spectrale des graphes est une théorie qui s'intéresse aux spectres d'opérateurs linéaires (agissant sur ces signaux) associés aux graphes. En particulier, l'opérateur linéaire laplacien  $L : \mathbb{R}^V \rightarrow \mathbb{R}^V$  est particulièrement important dans cette théorie, c'est notamment par son intermédiaire que la transformée de Fourier sur graphe peut être définie. Pour le comprendre, intéressons nous au laplacien usuel  $\Delta$  et à ses connexions avec la transformée de Fourier.

**Analogie.** Le laplacien  $\Delta$  est un opérateur différentiel du second ordre, qui est très présent en physique, il permet notamment d'écrire l'équation de propagation d'une onde :

$$\Delta\psi = \frac{1}{c^2} \frac{\partial^2 \psi}{\partial t^2}.$$

où encore l'équation de diffusion de la chaleur :

$$\Delta T = \frac{1}{D} \frac{\partial T}{\partial t}.$$

C'est en voulant déterminer les solutions de cette dernière équation que Joseph Fourier introduisit les séries de Fourier, qui fournissent un cadre dans lequel on peut décomposer toute fonction périodique en une série trigonométrique. Pour rappel, étant donnée  $f$  une fonction  $T$ -périodique<sup>4</sup>, sous certaines hypothèses relativement larges, cette fonction peut s'écrire :

$$\forall x \in [0, T], f(x) = a_0(f) + \sum_{k=1}^{+\infty} a_k(f) \cos(k \frac{2\pi}{T} x) + b_k(f) \sin(k \frac{2\pi}{T} x) \quad (2.20)$$

où les suites  $(a_k)_{k \in \mathbb{N}}$  et  $(b_k)_{k \in \mathbb{N}^*}$  dépendent entièrement de  $f$ . Cette série trigonométrique fait intervenir des sinusoides qu'on nomme fonctions harmoniques. Ces harmoniques sont intimement liées au laplacien. En effet, si on considère le problème aux valeurs propres du laplacien de dimension 1 suivant (Eq. (2.21)), où  $\Delta = \frac{\partial^2}{\partial x^2}$  et  $f$  est une fonction réelle,  $T$ -périodique :

$$\frac{\partial^2 f}{\partial x^2} = \lambda f \quad (2.21)$$

où  $\lambda$  est une valeur propre de l'opérateur laplacien. Il est immédiat que les valeurs propres du laplacien sont les  $k \frac{2\pi}{T}$  où  $k \in \mathbb{N}$  et que les vecteurs propres associés sont les éléments de l'ensemble vectoriel engendré par la base :

$$\text{Vect} \left\{ \cos(k \frac{2\pi}{T} \cdot), \sin(k \frac{2\pi}{T} \cdot) \right\}.$$

---

4. La décomposition est également possible si la fonction est définie sur un intervalle de longueur  $T$ .

Les bases de ces espaces vectoriels propres sont exactement les fonctions harmoniques sous lesquelles les fonctions  $T$ -périodique se décomposent ; et les valeurs propres sont les carrés des pulsations associées. On peut donc voir la décomposition en série de Fourier comme une décomposition sur les fonctions propres de l'opérateur laplacien. Par analogie, dès lors qu'on est en capacité de définir un laplacien dans un espace quelconque, on peut éventuellement y définir un analogue de la transformée de Fourier.

**Remarque 7 (Abus de notation.)** *On rappelle qu'on peut identifier le signal  $x$  de  $V$  dans  $\mathbb{R}$  avec le vecteur correspondant  $\mathbf{x}$  dans  $\mathbb{R}^n$ , i.e  $\mathbb{R}^V \approx \mathbb{R}^n$ . De ce fait, les opérateurs **linéaires** comme le laplacien normalisé  $\mathcal{L} : \mathbb{R}^V \rightarrow \mathbb{R}^V$  sont également identifiés avec la matrice correspondante agissant sur  $\mathbb{R}^n$ , i.e  $\mathbb{R}^V \rightarrow \mathbb{R}^V \simeq \mathbb{R}^n \rightarrow \mathbb{R}^n \simeq \mathbb{R}^{n \times n}$ <sup>5</sup>.*

**Laplacien combinatoire et normalisé.** En théorie des graphes, le spectre d'un graphe  $\mathcal{G} = (V, E)$  désigne donc généralement le spectre de l'opérateur Laplacien  $L : \mathbb{R}^V \rightarrow \mathbb{R}^V$  qui agit sur les signaux  $x : V \rightarrow \mathbb{R}$  de ce dernier. En notant  $y$  le signal image de  $x$  par  $L$ , son action s'écrit :

$$\forall u_i \in V, y(u_i) = \sum_{u_j \in \mathcal{V}(u_i)} (x(u_i) - x(u_j)) \quad (2.22)$$

Dans ce cadre, l'interprétation du laplacien est claire, c'est l'analogue discret du laplacien  $\Delta$ , i.e c'est une fonction qui évalue la différence entre la valeur d'un nœud et celles de ses voisins. Il est parfois plus commode de définir l'action de  $L$ , par l'action de la matrice associée  $\mathbf{L} \in \mathbb{R}^{n \times n}$  sur le vecteur  $\mathbf{x} \in \mathbb{R}^n$  associé au signal  $x$  :

$$\mathbf{L}_{i,j} = \begin{cases} d_i & \text{si } u_i = u_j \\ -1 & \text{si } \{u_i, u_j\} \in E \\ 0 & \text{sinon.} \end{cases} \quad (2.23)$$

Alors :

$$\mathbf{y} = \mathbf{L}\mathbf{x}.$$

On a une certaine liberté de choix sur la forme du laplacien, on pourra notamment lui préférer le laplacien normalisé  $\mathcal{L}$  qui possède un spectre compris entre 0 et 2 (cela facilite les comparaisons entre des graphes de tailles très différentes) et qui est notamment plus commode pour traiter des signaux sur graphes. Cet opérateur se définit directement à partir du laplacien combinatoire avec la matrice des degrés  $\mathbf{D}$  :

$$\mathbf{D}_{i,j} = \begin{cases} d_i & \text{si } u_i = u_j \\ 0 & \text{sinon.} \end{cases} \quad (2.24)$$

par la relation :

$$\mathcal{L} = \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}} \quad (2.25)$$

---

5. Même remarque que précédemment, la notation est impropre,  $\mathbb{R}^{n \times n}$  est isomorphe à l'ensemble des **applications linéaires** dans  $\mathbb{R}^n \rightarrow \mathbb{R}^n$ .

soit explicitement :

$$\mathcal{L}_{i,j} = \begin{cases} 1 & \text{si } u_i = u_j \\ -\frac{1}{\sqrt{d_i d_j}} & \text{si } \{u_i, u_j\} \in E \\ 0 & \text{sinon.} \end{cases} \quad (2.26)$$

Son action sur  $x$  s'écrit :

$$\mathbf{y} = \mathcal{L}\mathbf{x} \Leftrightarrow \forall u_i \in V, y(u_i) = \frac{1}{\sqrt{d_i}} \sum_{u_j \in \mathcal{V}(u_i)} \left( \frac{x(u_i)}{\sqrt{d_i}} - \frac{x(u_j)}{\sqrt{d_j}} \right) \quad (2.27)$$

Les valeurs propres de cet opérateur sont les  $\lambda$ , pour lesquelles :

$$\exists \mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}_n\}, \text{ tel que } \mathcal{L}\mathbf{x} = \lambda \mathbf{x} \quad (2.28)$$

Les valeurs propres de ce laplacien par analogie avec le laplacien usuel peuvent s'interpréter comme des fréquences mais elles sont aussi liées à certaines propriétés topologiques du graphe.

**Spectres.** La matrice du laplacien normalisée est une matrice symétrique réelle. A ce titre, il existe une famille de ses vecteurs propres formant une base unitaire orthogonale de l'espace  $\mathbb{R}^n$  (puisque l'espace de signaux sur le graphe est identifié à  $\mathbb{R}^n$ ) qu'on notera  $(\phi_0, \dots, \phi_n)$ ; la matrice dont la  $i$ -ième colonne est le vecteur propre  $\phi_i$  est notée  $\Phi$ . Le laplacien est également une matrice définie positive, ses valeurs propres sont donc positives et notées  $\lambda_0 \leq \lambda_1 \leq \dots \leq \lambda_{n-1}$  dans l'ordre croissant. On notera également  $\Lambda$  l'ensemble de ses valeurs propres et  $\mathbf{D}^\Lambda$  la matrice diagonale  $\mathbf{D}_{i,i}^\Lambda = \lambda_i$ . Ses valeurs propres s'interprètent comme des fréquences et les vecteurs propres associés comme des signaux évoluant à ces fréquences. Par exemple, quel que soit le graphe la première de ses valeurs propres est toujours nulle  $\lambda_0 = 0$  et est associée au vecteur propre unitaire :

$$\phi_0(u_i) = \frac{\sqrt{d_i}}{\sqrt{\sum_{u_j \in V} d_j}}. \quad (2.29)$$

Lorsque le graphe est régulier, alors ce vecteur propre est le signal constant, il est donc tout naturellement associé à la valeur propre 0 qui correspond donc bien à une *fréquence nulle*. Dans le cas d'un graphe quelconque, c'est à dire certainement irrégulier, la *non-constance* est un artefact de la normalisation du laplacien ; on peut voir ce vecteur comme une interprétation de la constance dans un espace non régulier<sup>6</sup>. Par ailleurs, cette valeur propre est liée à une propriété topologique du graphe, en effet sa multiplicité énumère le nombre de composantes connexes. Autre exemple, la valeur propre  $\lambda_1$  est liée au diamètre et au volume du graphe, en effet elle vérifie l'inégalité suivante :

$$\lambda_1 \geq \frac{1}{D(\mathcal{G}) \text{ vol } \mathcal{G}} \quad (2.30)$$

---

6. On notera que les vecteurs propres associés à la valeur propre 0 du laplacien combinatoire sont les vecteurs constants quel que soit le graphe.



Cette inégalité peut aussi être interprétée en termes de fréquence, en effet les graphes étant des objets finis, les signaux sur les graphes ne peuvent pas avoir des fréquences arbitrairement faibles. Il existe donc nécessairement une fréquences minimale (en dehors de 0), qui ne peut être inférieure à l'inverse de la taille caractéristique du graphe (ou réseau dans ce contexte). Ce qui se caractérise par une borne inférieure pour  $\lambda_1$ . Cette borne inférieure n'est évidemment pas optimale et peut être affinée selon le type de graphe.

Les valeurs propres suivantes sont également interprétables comme des fréquences plus élevées et leurs vecteurs propres comme des signaux évoluant à ces fréquences ; ils sont également liés à d'autres propriétés du graphe : les distances de plus court chemin, la distribution des degrés, son type, etc (voir [Chung and Graham 1997] pour plus de détails) ; les propriétés globales (resp. locales) étant plutôt liées aux valeurs propres les plus faibles (resp. élevées). C'est pourquoi, ces quantités sont un très bon moyen de caractériser un graphe et ont pu notamment être directement utilisées pour construire des mesures de similarités entre graphes [Tsitsulin et al. 2018].

**Remarque 8 (Caractérisation via le spectre.)** *Le spectre est un bon outil de caractérisation des graphes, toutefois son calcul est coûteux notamment lorsque les graphes sont de grandes tailles. De plus, il ne s'agit pas non plus d'un discriminateur parfait de graphe, car il existe des graphes différents, i.e., non-isomorphes, qui ont des spectres identiques (voir Annexe A.1.2.).*

## 2.2.2 Transformée de Fourier sur graphes

Le traitement du signal est la discipline qui se charge de l'analyse, le traitement et l'interprétation de signaux de toutes natures. La transformée de Fourier est un des outils de choix de cette discipline ; c'est pourquoi la transformée de Fourier sur graphe est importante, elle permet de transposer de nombreuses techniques de cette discipline portant sur des signaux euclidiens vers des signaux sur graphes [Shuman et al. 2012a]. Nous allons ici présenter quelques aspects essentiels de cette transformée.

### 2.2.2.a Principe

Considérons le graphe  $\mathcal{G}$ , la transformée de Fourier sur graphe noté  $\mathcal{GF}$  associe à chaque signal  $x \in \mathbb{R}^V$  un signal  $\hat{x} \in \mathbb{R}^\Lambda$  sur son spectre  $\Lambda$  de la manière suivante,  $\forall i \in \{0, \dots, n-1\}$  :

$$\hat{x}(\lambda_i) = \mathcal{GF}[x](\lambda_i) = \mathbf{\Phi}^T(i, :) \mathbf{x} = \sum_{k=0}^{n-1} x(u_k) \phi_i(u_k) \quad (2.31)$$

$\hat{x}$  est la représentation du signal  $x$  dans le domaine spectral, comme dans la transformée de Fourier usuelle on peut retrouver le signal réel à partir de cette représentation via la transformation de Fourier inverse définie comme suit,  $\forall i \in \{0, \dots, n-1\}$  :

$$x(u_i) = \mathcal{IGF}[\hat{x}](u_i) = \mathbf{\Phi}(i, :) \hat{\mathbf{x}} = \sum_{k=0}^{n-1} \hat{x}(\lambda_k) \phi_k(u_i) \quad (2.32)$$

Cette formule est analogue à la décomposition en série de Fourier usuelle (2.20). On remarquera que ces deux transformées peuvent s'écrire matriciellement respectivement :

$$\hat{\mathbf{x}} = \Phi^T \mathbf{x} \quad (2.33)$$

et

$$\mathbf{x} = \Phi \hat{\mathbf{x}} \quad (2.34)$$

Lorsque le signal est à valeur dans  $\mathbb{R}^q$  avec  $q > 1$ , on peut le représenter par la matrice  $\mathbf{X} \in \mathbb{R}^{n \times q}$  et la transformée de Fourier s'écrit composante par composante :

$$\hat{\mathbf{X}} = \Phi^T \mathbf{X} \quad (2.35)$$

et

$$\mathbf{X} = \Phi \hat{\mathbf{X}} \quad (2.36)$$

Ce sont les formes privilégiées notamment dans la littérature de l'apprentissage automatique.

La transformée de Fourier sur graphe partage de nombreuses propriétés en commun avec la transformée de Fourier classique. Elle est évidemment linéaire :

$$\forall \alpha \in \mathbb{R}, \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^V, \mathcal{GF}(\alpha \mathbf{x} + \mathbf{y}) = \alpha \mathcal{GF}(\mathbf{x}) + \mathcal{GF}(\mathbf{y}) \quad (2.37)$$

et l'identité de Parseval y est vérifiée :

$$\hat{\mathbf{x}} \cdot \hat{\mathbf{y}} = \mathbf{x} \cdot \mathbf{y}. \quad (2.38)$$

### 2.2.2.b Filtres linéaires

Les filtres linéaires dans les espaces euclidiens sont simplement des opérateurs linéaires qui s'appliquent sur des signaux. Ce sont des outils privilégiés de longue date du traitement du signal. Ils ont été notamment activement développés dans la communauté du traitement d'image (entre autres) qui s'est attachée à construire des filtres pour débruiter, déflouter, segmenter, dénombrer, extraire de l'information etc, et procéder à toutes sortes d'opérations sur les images. Si les filtres peuvent être définis dans le domaine spatial de l'image, ils se caractérisent tout particulièrement par leur action sur la transformée de Fourier du signal, où leur application est très simple, permettant de les appliquer à coût de calcul très faible. Les filtres sont des outils très puissants du traitement du signal, c'est donc tout naturellement que, disposant d'une transformée de Fourier sur graphe le concept de filtre a été généralisé aux graphes. Étant donné un graphe  $\mathcal{G}$  support du signal  $x$  comme défini dans la partie précédente, un filtre  $\mathcal{H} : \mathbb{R}^V \rightarrow \mathbb{R}^V$  est une fonctionnelle qui à un signal associe un autre signal, il est défini dans ce contexte dans le domaine spectral  $\hat{\mathbf{h}} = [\hat{h}(\lambda_0), \dots, \hat{h}(\lambda_n)] \in \mathbb{R}^\Lambda$ . Ce filtre agit sur le signal  $x$  par l'intermédiaire de sa transformée de Fourier  $\hat{x}$  de la manière suivante :

$$\mathcal{H}[x](u_i) = \sum_{k=0}^{n-1} \hat{h}(\lambda_k) \hat{x}(\lambda_k) \phi_k(u_i) \quad (2.39)$$

Ce qui s'écrit matriciellement :

$$\mathcal{H}[\mathbf{x}] = \Phi \text{diag}(\hat{\mathbf{h}}) \hat{\mathbf{x}} \quad (2.40)$$

Le filtre agit directement sur les composantes spectrales  $\hat{x}(\lambda_k)$  du signal  $x$ . Ce filtre a une contrepartie réelle  $h = \mathcal{IGF}(\hat{h})$ . Dans la transformée de Fourier classique, il y a une formule qui lie le filtre et sa contrepartie réelle  $h$ . Transposée aux graphes, elle s'écrirait :

$$\forall x \in \mathbb{R}^V, \mathcal{H}[x] = h * x \quad (2.41)$$

où  $*$  désigne l'opération de convolution. Cependant l'opération de convolution n'est pas définie sur les graphes car l'invariance par translation n'est pas définie pour un graphe quelconque. Le parti pris de la théorie spectrale des graphes est alors *justement* de définir l'opération de convolution par cette égalité obtenue par analogie, ainsi :

$$\forall x, h \in \mathbb{R}^V, h * x = \mathcal{IGF}[\text{diag}(\hat{\mathbf{h}}) \hat{\mathbf{x}}] \quad (2.42)$$

L'intérêt de cette formulation est qu'elle permet de rendre compte comme dans le cas euclidien qu'un filtre sur graphe peut être défini aussi bien dans le domaine spectral  $\mathbb{R}^A$  que dans le domaine spatial  $\mathbb{R}^V$ . Dans le cadre de la transformée de Fourier classique, ces deux formulations sont importantes car bien que l'on peut passer aisément de l'une à l'autre, elles se complètent dans leur utilisation. En effet, une relation d'incertitude lie les signaux avec leur contrepartie spectrale. Lorsqu'un signal est localisé spatialement il est alors *délocalisé* spectralement et inversement. Ainsi, si l'on souhaite éliminer un bruit localisé en fréquence, on aura plutôt tendance à définir le filtre en fréquence pour assurer qu'il cible son action sur les fréquences indésirables ; a contrario si l'on souhaite déflouter une image on aura tout intérêt à définir le filtre dans le domaine spatial, pour s'assurer que l'image soit traitée localement. Les mêmes considérations sont applicables aux filtres sur graphes. A la différence fondamentale qu'il n'est pas possible d'appliquer un filtre dans le domaine spatial, il est nécessaire de l'appliquer dans le domaine spectral via la transformée sur graphes. Or, cette opération est coûteuse en termes de calcul puisque la transformée de Fourier rapide n'existe pas pour les graphes. Cette complexité était un des principaux freins à la généralisation des réseaux de neurones convolutifs sur graphes que nous allons présenter dans la section suivante.

## 2.3 Réseaux de neurones convolutifs sur graphes

Pour conclure ce chapitre, nous nous intéressons maintenant aux réseaux de neurones convolutifs. Ces réseaux ont permis de construire des modèles capables de construire *automatiquement* des caractéristiques complexes et de qualité en vue de réaliser une tâche donnée. Nous allons décrire brièvement comment ces réseaux définis sur des données régulières et globalement structurées comme des images ont pu être adaptés aux données irrégulières et localement structurées que sont les graphes et les graphes attribués.

### 2.3.1 Préliminaires : apprentissage avec réseaux de neurones

Avant d'aborder la question de l'adaptation des réseaux de convolutions aux graphes, nous allons introduire brièvement certains principes de l'apprentissage importants pour cette thèse. Notamment à travers l'exemple des réseaux de neurones (en particulier du perceptron multi-couche) et de la tâche de classification. Nous introduirons également des éléments de vocabulaire qui seront réutilisés dans ce manuscrit.

#### 2.3.1.a Apprentissage automatique

L'apprentissage automatique est l'ensemble des méthodes qui vise à améliorer automatiquement les performances pour des tâches à exécuter sur un ordinateur sans être explicitement programmé pour le faire, en s'appuyant notamment sur des méthodes issues des statistiques et de l'optimisation mathématique. En général, ces méthodes d'apprentissage se basent sur la connaissance de certaines informations ou données pour « apprendre » à effectuer la tâche. Ces méthodes d'apprentissage se regroupent en plusieurs grandes catégories, dépendant du type de données sur lesquelles elles apprennent <sup>7</sup> :

- **Méthodes supervisées.** Il s'agit des méthodes qui apprennent à partir de données étiquetées. En général, l'objectif de ces méthodes est de déterminer le lien entre les données et les étiquettes par exemple pour faire de la classification des données. Dans le Chapitre 5, nous proposons une telle méthode pour construire une métrique adaptée en vue notamment d'améliorer les performances en classification d'autres méthodes supervisées qui se basent sur des mesures de distance entre les données.
- **Méthodes non-supervisées.** Il s'agit de méthodes qui apprennent à partir de données non étiquetées. Il s'agit alors de déterminer la structure et les caractéristiques d'un jeu de données sans informations supplémentaires. Les algorithmes de *clustering* ou de génération de données appartiennent par exemple à cette catégorie. Les chapitres 3 et 4 présentent chacun un modèle non-supervisé, le premier présente un modèle de représentation de graphes tandis que le second présente un modèle de représentation de nœuds multi-attribués. Dans les deux cas les étiquettes ne sont pas utilisées.
- **Méthodes semi-supervisées.** Il s'agit d'algorithmes qui apprennent à partir de données partiellement étiquetées. Ces algorithmes ont un intérêt dans la mesure où l'étiquetage de données est fastidieux et coûteux. Ils permettent alors à la fois d'exploiter les informations apportées par ces étiquettes, notamment lorsqu'elles sont peu disponibles, tout en tirant partie de la structure des données non étiquetées qui sont souvent présente en plus grand nombre. Ceci les différencie strictement des méthodes supervisées qui ont besoin d'un nombre conséquent de données labellisées ainsi que des approches non supervisées qui ne peuvent exploiter ces données étiquetées.

---

7. Il existe une quatrième catégorie, en dehors du cadre de cette thèse, l'apprentissage par renforcement, qui consiste à apprendre à exécuter des actions à partir d'expériences.

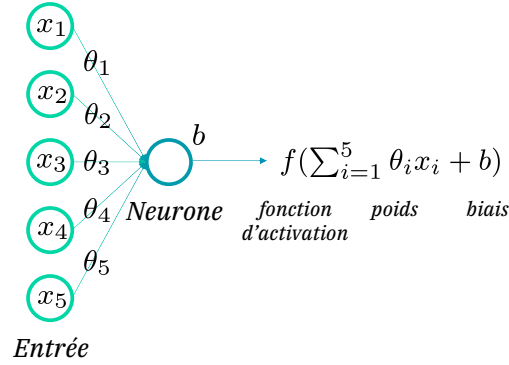


FIGURE 2.6 – **Neurone**. Fonctionnement d'un neurone qui se définit par ses poids  $\Theta = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5]$  et son biais  $b$ . La sortie consiste en une combinaison de l'entrée  $\mathbf{x}$  avec les poids et l'ajout du biais suivi de l'activation de la fonction d'activation  $f$  qui est non linéaire.

### 2.3.1.b Réseaux de neurones

Dans le domaine de l'apprentissage automatique, les réseaux de neurones sont devenus des outils importants ces dernières années.

Le modèle le plus simple, appelé perceptron, est une unité de calcul correspondant à un classifieur linéaire suivi généralement de l'application d'une fonction appelée fonction d'activation. Ce type de neurone artificiel a pour paramètres un vecteur de poids  $\Theta$  dont la dimension est égale à celle de l'entrée et d'un biais (ou seuil)  $b$  (voir Figure (2.6)). Ce modèle peut être étendu en utilisant plusieurs neurones en parallèle travaillant sur les données d'entrée et dont les sorties peuvent ensuite être combinées à l'aide de l'application d'une fonction d'activation ou de sortie donnée. On parle alors d'une couche de neurones. Ce modèle peut encore être étendu en combinant plusieurs couches successives, les sorties des neurones d'une couche devenant les entrées de la couche suivante. On obtient alors un perceptron multicouches, voir la Figure (2.7) pour une illustration. L'action d'un tel modèle sur un vecteur  $\mathbf{v} \in \mathbb{R}^q$  s'écrit en notant  $\mathbf{v}^{(k)}$  le vecteur de dimension  $q^{(k)}$  des sorties des  $q^{(k)}$  neurones de la  $k$ -ième couche, avec  $\mathbf{v}^{(0)} = \mathbf{v}$  et  $q^{(0)} = q$  et  $K$  la profondeur du réseau et correspondant au nombre de couches :

$$\mathbf{v}^{(K)} = F_{\theta}(\mathbf{v}^{(0)}) = \Theta^{(K-1)} f(\dots f(\Theta^{(1)} f(\Theta^{(0)} \mathbf{v}^{(0)} + \mathbf{b}^{(0)}) + \mathbf{b}^{(1)}) \dots) + \mathbf{b}^{(K-1)} \quad (2.43)$$

Où, on a noté  $F_{\theta}$  la fonction que réalise le perceptron multicouches et  $\theta = \{\Theta^{(0)}, \mathbf{b}^{(0)}, \dots, \Theta^{(K-1)}, \mathbf{b}^{(K-1)}\}$  l'ensemble de ses paramètres, qui sont pour tout  $k \in \{0, \dots, K-1\}$  les matrices  $\Theta^{(k)} \in \mathbb{R}^{q^{(k+1)} \times q^{(k)}}$  et les vecteurs  $\mathbf{b}^{(k)} \in \mathbb{R}^{q^{(k+1)}}$ .  $f$  est une non-linéarité (non polynomiale) appliquée coefficient par coefficient. En général, on choisit  $f = \text{ReLU}$ , c'est-à-dire que  $\forall x, f(x) = \max(x, 0)$ . Les couches après la couche d'entrée et avant la couche finale sont les couches cachées. Le théorème d'approximation universelle [Pinkus 1999] assure qu'un perceptron multicouche<sup>8</sup> avec une seule couche cachée permet d'approximer de manière arbitrairement proche (la valeur des

8. Les poids de biais  $\mathbf{b}^{(k)}$  ne sont pas nécessaires.

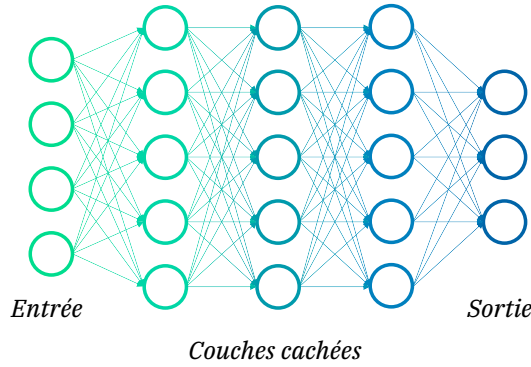


FIGURE 2.7 – **Perceptron multicouches.** Chaque alignement vertical de neurones (représentés par des cercles) correspond à une couche. Ce perceptron multicouches, possède 3 couches cachées et a une profondeur de 4. Chaque neurone de la couche  $k \in \{1, 2, 3, 4\}$  représente une ligne de la matrice de paramètre  $\Theta^{(k)}$  (voir Eq. (2.43)) et son action correspond à effectuer le produit scalaire de cette ligne avec les valeurs de sortie des neurones de la couche qui le précède, puis à envoyer la valeur résultante aux neurones de la couche qui le suit.

$q^{(k)}$  augmentant avec la précision voulue) toute fonction continue de  $\mathbb{R}^{q^{(0)}}$  dans  $\mathbb{R}^{q^{(k)}}$ . Ce pouvoir d'approximation permet d'assurer que ces modèles vont avoir une expressivité suffisante pour traiter des tâches complexes. La prochaine étape est la question de l'apprentissage de ces modèles que nous allons voir maintenant.

**Entraînement.** Supposons que l'on ait un jeu de données  $\{\mathbf{v}_i\}_{i=1}^n$  (où les  $\mathbf{v}_i$  appartiennent à un espace d'instances  $\mathbb{X}$ ) associé à un ensemble d'étiquettes  $\{\mathbf{y}_l\}_{l=1}^{|\mathbb{Y}|}$  possibles (où les  $\mathbf{y}_l$  appartiennent à un espace d'instance  $\mathbb{Y}$ ). On suppose qu'il existe une fonction  $\mathcal{E} : \mathbb{X} \rightarrow \mathbb{Y}$  qui à chaque élément de  $\mathbb{X}$  associe son étiquette correspondante dans  $\mathbb{Y}$ . L'idée de la classification avec le perceptron est simplement d'apprendre cette fonction, c'est-à-dire déterminer les paramètres  $\theta$  tel que  $F_\theta \approx \mathcal{E}$  de tel sorte qu'on puisse ensuite appliquer  $F_\theta$  sur des éléments dont on ne connaît pas les étiquettes. Pour ce faire, on définit une fonctionnelle  $\mathcal{F}$  (ou *loss*) qui va caractériser l'écart qu'il y a entre  $F_\theta$  et  $\mathcal{E}$ , elle s'écrit généralement sous cette forme<sup>9</sup> :

$$\mathcal{F}(F_\theta, \mathcal{E}) = \sum_{i=1}^n l(F_\theta(\mathbf{v}_i), \mathcal{E}(\mathbf{v}_i)) \quad (2.44)$$

où  $l$  mesure l'adéquation entre la prédiction  $F_\theta(\mathbf{v}_i)$  et  $\mathcal{E}(\mathbf{v}_i)$  en pénalisant les prédictions incorrectes ou imprécises. En pratique,  $\mathcal{E}$  associe à chaque vecteur  $\mathbf{v}_i$  un encodage *one-hot*<sup>10</sup> de l'une des étiquettes  $y_l$  plutôt que directement l'étiquette elle-même. On considère aussi  $g(F_\theta)$  où  $g$

9. Cette forme est spécifique à la tâche de classification, évidemment d'autres formes sont admissibles en fonction de la tâche souhaitée.

10. Il s'agit d'un encodage qui associe aux étiquettes qui peuvent prendre  $|\mathbb{Y}|$  état un vecteur de taille  $|\mathbb{Y}|$  avec des 0 sur chaque composante sauf une où on y trouve un 1 qui correspond à la présence de l'étiquette associée à l'exemple considéré.

est une fonction différentiable qui restreint l'ensemble d'arrivée de  $F_\theta$ , en l'occurrence dans ce cas-là il s'agit du softmax<sup>11</sup>, pour que les valeurs de  $F_\theta$  soit dans le simplexe<sup>12</sup>. Ceci permet notamment d'interpréter  $g(F_\theta(\mathbf{v}_i))$  et  $\mathcal{E}(\mathbf{v}_i)$  comme des probabilités et d'utiliser les outils de la théorie de l'information pour les comparer (voir remarque ci-dessous). Cette façon de procéder est très efficace et permet notamment d'avoir une capacité de généralisation bien meilleure. On pourra se référer à [Goodfellow et al. 2016] pour plus de détails à ce sujet. Quoiqu'il en soit pour déterminer nos  $\theta$  on procède à ce qu'on appelle l'entraînement, qui consiste à effectuer une succession de descentes de gradient qui met à jour les paramètres  $\theta$  (la mise à jour des paramètres se fait notamment grâce à la technique de *backpropagation*) de l'expression en (2.44) pour la minimiser :

$$\theta' = \theta - \alpha \Delta_\theta \mathcal{F}(g(F_\theta), \mathcal{E}) \quad (2.45)$$

Où  $\Delta_\theta$  est le gradient par rapport aux paramètres  $\theta$  et  $\alpha$  un paramètre strictement positif qui quantifie le « pas » effectué dans l'espace des paramètres  $\theta$ . Ce paramètre est ce qu'on appelle le *learning rate*, et doit être judicieusement choisi. Un *learning rate* trop faible impliquera un apprentissage long tandis qu'un *learning rate* trop grand rendra la convergence impossible. De nombreuses méthodes de mises à jour automatiques de ce *learning rate* existent, nous utiliserons dans nos méthodes l'algorithme Adam [Kingma and Ba 2014]. Comme les jeux de données peuvent être relativement volumineux, les descentes de gradient (2.45) sont effectuées seulement sur des sous-ensembles d'éléments du jeu de données (qui forment une partition de ce jeu de données), qu'on appelle des *batch*. Lorsque l'on a réalisé une descente de gradient sur chaque *batch* d'un jeu de données, on a effectué une *epoch*. L'entraînement se termine après un certain nombre d'*epochs*<sup>13</sup> déterminé à l'avance ou selon un critère donné qui peut par exemple consister : à arrêter l'entraînement lorsque le taux de variation moyen de la fonctionnelle (2.44) d'une *epoch* à une autre est inférieur à une certaine quantité ; à faire de l'*early stopping*, c'est-à-dire évaluer notre fonction  $g(F_\theta)$  sur un ensemble différent de l'ensemble de test et dont on connaît les étiquettes (appelé ensemble de validation), et arrêter l'entraînement dès que l'on constate que les performances de  $g(F_\theta)$  se dégradent sur cet ensemble.

Si l'entraînement décrit ci-dessus est aujourd'hui efficace, cela a été permis par plusieurs avancées : sur les méthodes d'optimisation mais également sur l'initialisation des paramètres qui a permis de résoudre certains problèmes d'instabilités de gradient [Glorot and Bengio 2010] (notamment lorsque le nombre de couches cachées est important) ; l'introduction de techniques de régularisation comme le *dropout*<sup>14</sup> [Srivastava et al. 2014] qui ont permis d'améliorer la capacité à généraliser des réseaux de neurones ; et les progrès technologiques qui ont permis de rendre les calculs matriciels extrêmement rapides.

11. Pour  $\mathbf{a} = [a_1, \dots, a_{|\mathcal{Y}|}]^T \in \mathbb{R}^{|\mathcal{Y}|}$ ,  $\text{softmax}(\mathbf{a}) = [\frac{e^{a_1}}{\sum_i e^{a_i}}, \dots, \frac{e^{a_{|\mathcal{Y}|}}}{\sum_i e^{a_i}}]$

12. Ensemble des vecteurs à coefficients positifs, qui somme à 1.

13. La partition en *batch* est aléatoire et refaite à chaque *epoch*.

14. Il s'agit d'une technique qui consiste à considérer un certain pourcentage des coefficients des paramètres  $\theta$  comme nuls (différents à chaque *batch*), lors du calcul de  $\mathcal{F}$ . Ceci force le réseau de neurones à réaliser sa tâche même lorsque certaines informations sont indisponibles.

**Architectures particulières.** Tels que définis, les perceptrons sont parfois trop expressifs, et cela peut limiter leur capacité de généralisation. Il est intéressant de proposer des architectures particulières, tenant compte de particularités du problème étudié. La prise en compte de tels particularités peut notamment aider à construire une approximation plus « générale » tout en réduisant le nombre de poids à ajuster dans le réseau<sup>15</sup>. Les architectures de réseaux de neurones convolutionnels, ou *Convolutional Neural Networks* (CNN) en anglais, (voir Fig. (2.8)) sont nées de la connaissance du fonctionnement du cortex visuel chez certains animaux [Fukushima et al. 1983] mais aussi de la prise en compte de certaines particularités propres aux tâches sur image : notamment la propriété d’invariance par translation, c’est-à-dire qu’une information sur image doit pouvoir être détectée indépendamment de sa position dans l’image. Cette propriété combinée à certaines des avancées sur l’entraînement des réseaux de neurones évoqués plus haut ont permis d’atteindre d’excellents résultats en classification d’images, mais aussi dans d’autres tâches relatives à la vision par ordinateur, ou encore en traitement du langage naturel. L’opération au centre de ces architectures est l’opération de convolution que la communauté de l’apprentissage automatique a voulu généraliser aux graphes et dont nous allons décrire dans la suite le fonctionnement.

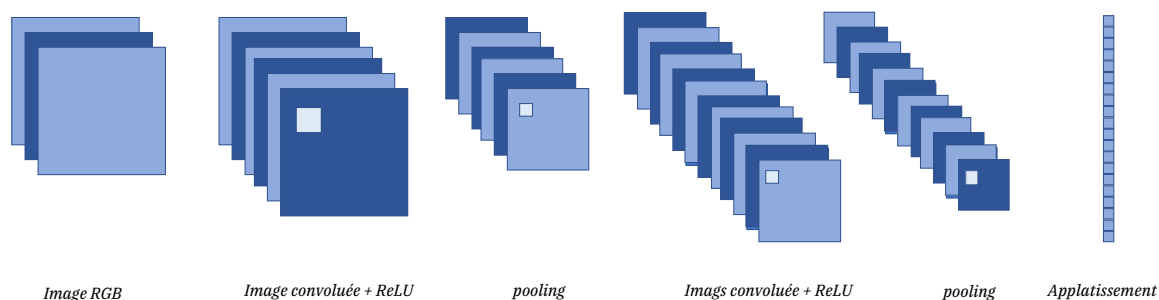


FIGURE 2.8 – **Architecture d’un CNN.** En partant d’une image couleur avec 3 canaux RGB, on convolue des filtres (voir Fig.(2.9)) avec chaque canal pour générer les canaux de l’image de la couche suivante après application d’une non-linéarité (voir Eq. (2.46)). On applique ensuite une opération de *pooling* (voir Fig.(2.10)) pour réduire la taille des images (pour contrebalancer l’effet de l’augmentation du nombre de canaux, notamment pour ne pas avoir des matrices trop grandes en mémoire). On procède ainsi de suite un certain nombre de fois, puis les images finales obtenues sont ”aplaties” dans un seul vecteur de sortie. Le vecteur ainsi obtenu, peut alors être utilisé en entrée d’un réseau de neurones comme le perceptron multicouches lui même connecté à une fonctionnelle adéquate pour entraîner l’ensemble de l’architecture. Tout au long de ce réseau, l’invariance par translation est respectée par construction (au sens où translater l’image revient à translater les caractéristiques intermédiaires du réseau) sauf au moment de l’aplatissement.

15. De la même manière une régression linéaire est parfois plus informative qu’une régression complexe.



### 2.3.2 Motivations

Les CNN sont basés sur un type particulier de filtres linéaires s'appliquant sur des images qui sont *spatialement localisées* et *entraînables*. Ces derniers sont depuis quelques années très présents dans l'état de l'art, permettant notamment d'atteindre des performances exceptionnelles dans certaines tâches de vision par ordinateur. Ces performances spectaculaires s'expliquent par le caractère entraînable de ces filtres qui permet d'automatiser le processus de sélection des filtres appropriés pour une tâche donnée mais surtout de l'architecture selon laquelle ces filtres sont agencés et la procédure d'entraînement associée [Goodfellow et al. 2016].

Dans les architectures de réseaux de neurones convolutionnels (voir Fig. (2.8)), les filtres sont empilés les uns après les autres. Chaque filtre extrait des informations pertinentes pour résoudre une tâche donnée : il se crée naturellement une hiérarchie dans laquelle les premiers filtres du réseau vont extraire des informations relativement simples<sup>16</sup> qui vont permettre aux couches suivantes d'élaborer des concepts de plus en plus complexes<sup>17</sup>. Cette habileté à construire des concepts complexes va croissante avec la quantité de données d'entraînement à disposition et la profondeur de l'architecture.

### 2.3.3 Problématique : des CNN aux GCN

Considérons un réseau de neurones convolutifs de profondeur  $K$  opérant sur des images, la  $k$ -ième couche de ce réseau traite une image  $\mathbf{I}^{(k)} \in \mathbb{R}_+^{h^{(k)} \times l^{(k)} \times a^{(k)}}$ <sup>18</sup>, où  $h^{(k)}$  est la hauteur de l'image en pixel,  $l^{(k)}$  sa largeur et  $a^{(k)}$  sa profondeur. Cette couche applique un nombre  $a^{(k)} * a^{(k+1)}$  de filtres linéaires de convolutions  $(\mathbf{H}_{a,b}^{(k)})_{\substack{a \in \{1, \dots, a^{(k)}\} \\ b \in \{1, \dots, a^{(k+1)}\}}}$  tel que  $\mathbf{H}_{a,b}^{(k)} \in \mathbb{R}^{(2*c^{(k)}+1) \times (2*c^{(k)}+1)}$

avec  $c^{(k)}$  entier, pour produire l'image  $\mathbf{I}^{(k+1)}$ . Ces filtres sont dits localisés spatialement car leur action est locale autour de chaque pixel. Dans l'exemple de la Figure 2.9 les filtres sont des carrés de côtés impairs pour simplifier la situation<sup>19</sup>. On pourra se référer à Goodfellow et al. [2016] pour le cas général. Le calcul de  $\mathbf{I}^{(k+1)}$  se déroule ainsi (voir illustration en Figure 2.9) :

- Les filtres sont appliqués sur chaque tranche de l'image d'entrée de la couche  $k$  pour construire le pixel  $(h, l, a)$  de l'image de la couche  $k + 1$  de la manière suivante<sup>20</sup> :

$$\mathbf{I}^{(k+1)}(h, l, a) = \sum_{b=1}^{a^{(k)}} \mathbf{H}_{a,b}^{(k)} * \mathbf{I}^{(k)}(:, :, b) = \sum_{b=1}^{a^{(k)}} \left( \sum_{h'=-c^{(k)}}^{c^{(k)}} \sum_{l'=-c^{(k)}}^{c^{(k)}} \mathbf{H}_{a,b}^{(k)}(h', l') \mathbf{I}^{(k)}(h+h', l+l', b) \right) \quad (2.46)$$

16. L'ensemble des frontières de l'image, par exemple.

17. Avec l'ensemble des frontières, on peut facilement définir l'ensemble des surfaces fermées de l'image, par exemple.

18.  $\mathbf{I}^{(k)}$  est un tenseur, il sera noté en gras.

19. Dans ce but, on considérera aussi que le pixel central de  $\mathbf{H}_{a,b}$  est situé à la coordonnée  $(0, 0)$ . L'image est indexée de manière usuelle.

20. Ici on choisit de sommer les résultats issus des différents filtres selon  $b$ , mais d'autres choix sont possibles. On peut notamment décider de conserver tous les résultats de filtrages avec  $\mathbf{H}_{a,b}^{(k)}$ , on remplace ainsi la somme sur  $b$  par une concaténation sur la profondeur. La profondeur de l'image  $\mathbf{I}^{(k+1)}$  est alors égale aux nombres de filtres  $\mathbf{H}_{a,b}^{(k)}$  appliqués.

\* désigne l'opération de convolution. Chaque tranche de  $\mathbf{I}^{(k+1)}(h, l, a)$  correspond à une caractéristique extraite par un filtre. On remarquera que la formule est mal définie au bord de l'image car les indices  $h + h'$  et  $l + l'$  peuvent excéder la taille de l'image. Pour pallier à cette problématique, on peut soit considérer que l'image se prolonge au bord avec des valeurs nulles, soit qu'elle est périodique, soit ne considérer que les couples  $(h, l)$  où cette situation ne se produit pas. Dans les deux premiers cas, l'image  $\mathbf{I}^{(k+1)}$  sera de même taille que l'image  $\mathbf{I}^{(k)}$ , alors qu'elle sera plus petite de  $c^{(k)}$  en hauteur et en largeur dans le dernier cas (ce cas est illustré en Figure (2.9)).

- Ensuite une étape de *pooling* (voir Fig.(2.10)) qui consiste essentiellement à réduire la taille de l'image  $\mathbf{I}^{(k+1)}$  est généralement effectuée puis une non-linéarité est appliquée à chaque composante de l'image, généralement une ReLU. Cette opération explique que les éléments de  $\mathbf{I}^{(k)}$  soient dans  $\mathbb{R}_+$ .

Le réseau procède ainsi de suite jusqu'à produire l'image finale  $\mathbf{I}^{(K)}$  qui est alors traitée par un réseau de neurones classique (voir Eq. (2.43)) connecté à une fonctionnelle d'entraînement adéquate. Le cœur de ce réseau est l'opération de convolution qu'on souhaite généraliser aux graphes<sup>21</sup>.

En s'appuyant sur la transformée de Fourier sur graphes, la transposition de l'Equation (2.46) est immédiate [Bruna et al. 2014]. On rappelle que le signal vectoriel sur un graphe peut être représenté par la matrice  $\mathbf{X}^{(k)} \in \mathbb{R}^{n \times a^{(k)}}$ . Les filtres quant à eux  $(\mathcal{H}_{a,b}^{(k)})_{\substack{a \in \{1, \dots, a^{(k)}\} \\ b \in \{1, \dots, a^{(k+1)}\}}}$  sont définis

dans le domaine de Fourier  $\mathcal{H}_{a,b}^{(k)} = \text{diag}(\mathbf{h}_{a,b}^{(k)})$ . Et  $\Phi$  est la matrice dont les colonnes sont les vecteurs propres du laplacien normalisé qui permet de réaliser la transformée de Fourier sur graphe et son inverse. Le signal filtré résultant  $\mathbf{X}^{(k+1)} \in \mathbb{R}^{n \times a^{(k+1)}}$  s'écrit ainsi en fonction de ces grandeurs :

$$\mathbf{X}^{(k+1)}(:, a) = \sum_{b=1}^{a^{(k)}} \mathcal{H}_{a,b}^{(k)} * \mathbf{X}^{(k+1)}(:, b) = \sum_{b=1}^{a^{(k)}} \Phi \text{diag}(\mathbf{h}_{a,b}^{(k)}) \Phi^T \mathbf{X}^{(k+1)}(:, b) \quad (2.47)$$

Cette première formulation naïve n'est pas satisfaisante notamment car :

- Par construction, les CNN sont des filtres localisés spatialement. Les filtres sur graphes étant définis dans le domaine spectral (voir en Section 2.2.2.b) il est difficile d'assurer leur localisation spatiale.
- Les CNN appris sur un lot d'images peuvent-être utilisés sur un autre lot d'images, tandis que les filtres sur graphes de (2.47) dépendent explicitement de la base  $\Phi$  du graphe sur laquelle ils sont appliqués. Il en découle qu'un filtre appris sur un graphe ne peut pas être réutilisé sur un autre graphe
- L'un des éléments importants participant au succès des CNN est leur faible coût de complexité de calcul. La nécessité d'effectuer une transformée de Fourier pour appliquer le filtre (2.47) et l'inexistence de la transformée de Fourier rapide sur graphes (malgré quelques avancées cf. [Le Magoarou et al. 2017]) font des filtres de l'Equation (2.47) des quantités coûteuses à calculer.

21. Une notion de *pooling* sur graphe sera introduite en Section 3.3.1.b.

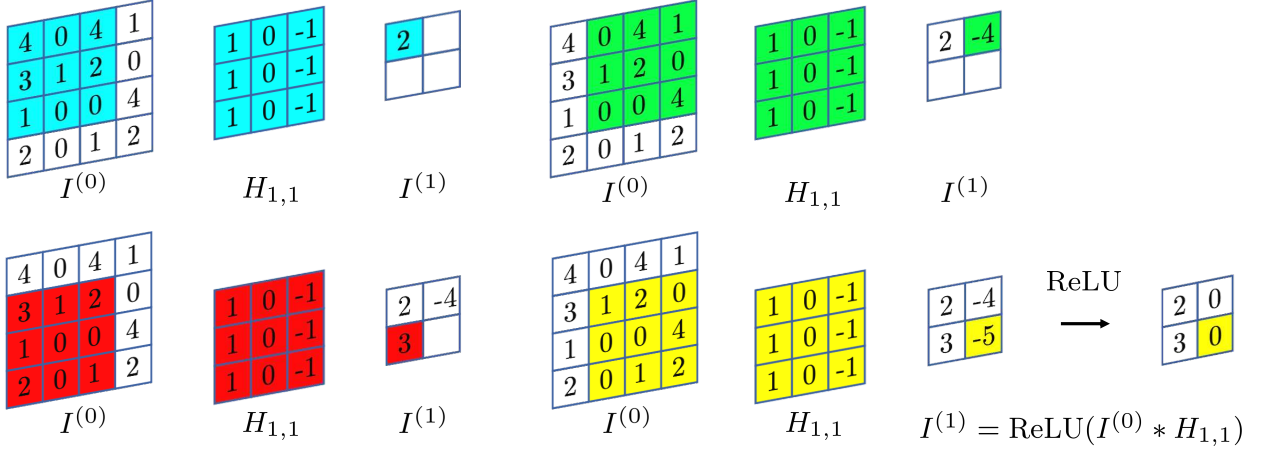


FIGURE 2.9 – **Application d'un filtre de convolution.** Pour simplifier, on considère une image  $I^{(0)}$  sans profondeur et un unique filtre  $H_{1,1}$ . Les coefficients de  $I^{(1)}$  sont obtenus en appliquant le filtre centré sur chaque pixel, ce qui consiste à : faire une multiplication termes à termes puis à sommer l'ensemble des termes. Quant aux effets de bord, la solution de cette illustration consiste à ne pas considérer les pixels qui feraient « déborder » le filtre, ce qui explique que l'image  $I^{(1)}$  soit plus petite. Une non linéarité comme la ReLU est ensuite appliquée sur chaque coefficient.

Il est donc nécessaire de procéder à des ajustements pour que les filtres sur graphes soient viables dans une démarche d'apprentissage automatique.

### 2.3.4 Approximation polynomiale des GCN

Defferrard et al. [2016] proposent de résoudre les problèmes évoqués plus haut en écrivant les filtres comme des fonctions polynomiales de l'ensemble des vecteurs propres du graphe ; et plus particulièrement via les polynômes de première espèce de Chebyshev<sup>22</sup>  $T$  définis tels que  $T_0 = 0$ ,  $T_1 = 1$  et  $\forall k > 2, T_k = 2xT_{k-1} - T_{k-2}$ . Ces polynômes forment une base des fonctions de carrés intégrables à valeur dans  $[-1, 1]$  ; et permettent de les approximer relativement finement d'où l'intérêt de rechercher notre filtre sous cette forme. En lieu et place du filtre diagonal  $\mathcal{H} = \text{diag}(\mathbf{h})$  caractérisé par des poids entraînaibles  $\mathbf{h}$ , ce filtre polynomial va se caractériser par un entier  $S$  qui est l'ordre de son développement et un vecteur entraînable  $\boldsymbol{\theta} \in \mathbb{R}^S$  qui contient les poids du développement. Soient deux signaux  $\mathbf{x} \in \mathbb{R}^n$  et  $\mathbf{y} \in \mathbb{R}^n$  tels que  $\mathbf{y} = \mathcal{H} * \mathbf{x}$  alors :

$$\mathbf{y} = \tilde{\Phi} \left( \sum_{s=1}^S \theta_s T_s(D^{\tilde{\Lambda}}) \right) \tilde{\Phi}^T \mathbf{x} = \sum_{s=1}^S \theta_s \tilde{\Phi} T_s(D^{\tilde{\Lambda}}) \tilde{\Phi}^T \mathbf{x}$$

où  $\tilde{\Phi}$  sont les matrices dont les colonnes sont constituées des vecteurs propres du laplacien  $\tilde{\mathcal{L}} = \frac{2}{\lambda_n^L} \mathbf{L} - \mathbf{I}_n$ ,  $D^{\tilde{\Lambda}}$  la matrice dont la diagonale est constituée de ses valeurs propres et  $\lambda_n^L$  la

22. Cette possibilité avait déjà été évoquée et appliquée par Hammond et al. [2011].

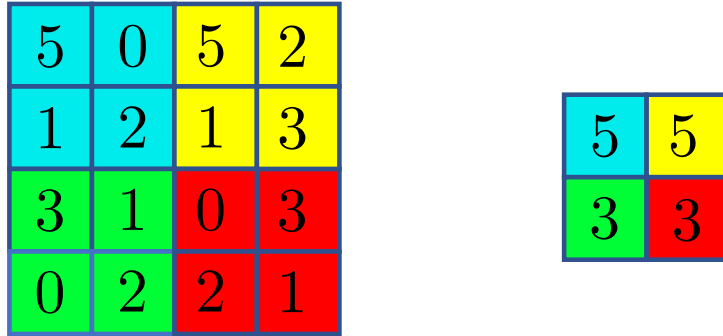


FIGURE 2.10 – **Application du max-pooling.** Le max-pooling est un type particulier de *pooling* courant dans la littérature. Ce *pooling* consiste à définir une forme ( $2 \times 2$  ici) qu'on balaye sur l'image et dont on prend le max pour chaque ensemble de pixels balayés. On peut considérer des fonctions autres que le max, comme la moyenne ou le min. On peut également considérer d'autres paramètres de balayage. Par exemple, on a décidé ici de ne pas se faire chevaucher les différents groupes de nœuds. Il peut en être autrement, on pourrait autoriser les juxtapositions des groupes de nœuds et même autoriser les débordements sur le bord de l'image. Cette opération a avant tout pour but de réduire la taille de l'image à traiter par la couche suivante notamment car le nombre de canaux (ou profondeur des images) augmente à mesure que l'on progresse dans les couches du réseau.

plus grande valeur propre de  $L$ , le laplacien combinatoire. Cela permet d'assurer que les valeurs propres dans  $D^{\tilde{\Lambda}}$  sur lesquelles agissent les polynômes de Chebyshev soient bien comprises dans l'intervalle  $[-1, 1]$ , i.e le domaine de validité des polynômes de Chebyshev. Ce choix particulier de filtre motivé par la nature des polynômes de Chebyshev (tout filtre peut se décomposer dans cette base) a plusieurs conséquences :

- Ces filtres sont localisés, la valeur en sortie de filtre en un nœud  $u$  ne dépend que des signaux portés par les nœuds  $\mathcal{V}_S(u)$ .
- La complexité de calcul de ces filtres est faible, et est égale à celle des CNN (modulo quelques constantes) car le filtre peut se réécrire :

$$\sum_s \theta_s \tilde{\Phi} T_s(D^{\tilde{\Lambda}}) \tilde{\Phi}^T = \sum_s \theta_s T_s(\tilde{\Phi} D^{\tilde{\Lambda}} \tilde{\Phi}^T) = \sum_s \theta_s T_s(\tilde{\mathcal{L}})$$

Cette expression peut se calculer récursivement par la définition des polynômes de Chebyshev et ne nécessite pas de calculer de transformée de Fourier sur graphe [Shuman et al. 2012b].

- L'expression de ce filtre est indépendante de la base choisie [Isufi et al. 2016; Tremblay et al. 2017].

Cette formulation est une solution à tous les points limitants relevés plus haut. Toutefois [Kipf and Welling 2016c] ont montrés que simplifier cette formulation en choisissant  $S = 2$  et en imposant  $\theta_0 = -\theta_1$  permet de fortement réduire la complexité (chaque filtre est alors

défini par un unique coefficient) sans dégrader les performances du GCN. L'expression scalaire de l'Equation (2.47) se simplifie donc en :

$$\mathbf{X}^{(k+1)}(:, a) = \sum_{b=1}^{a^{(k)}} \Theta_{a,b}^{(k)} (\mathbf{I}_n + \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}) \mathbf{X}^{(k+1)}(:, b)$$

Soit vectoriellement :

$$\mathbf{X}^{(k+1)} = (\mathbf{I}_n + \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}) \mathbf{X}^{(k)} \Theta^{(k)} \quad (2.48)$$

où  $\Theta^{(k)} \in \mathbb{R}^{a^{(k)} \times a^{(k+1)}}$  dont chaque coefficient représente un filtre. Cette expression souffre toutefois de problème de stabilité du gradient, Kipf and Welling [2016c] ont montré empiriquement qu'en la modifiant légèrement on peut résoudre ce problème et c'est sous cette forme que le premier GCN facilement utilisable et empilable comme un réseau profond est apparu dans la littérature :

$$\mathbf{X}^{(k+1)} = \text{ReLU}(\tilde{\mathbf{A}} \mathbf{X}^{(k)} \Theta^{(k)}) \quad (2.49)$$

Où  $\tilde{\mathbf{A}} = (\tilde{\mathbf{D}}^{-\frac{1}{2}} (\mathbf{I}_n + \mathbf{A}) \tilde{\mathbf{D}}^{-\frac{1}{2}})$  est le terme qui permet de stabiliser le gradient.  $\tilde{\mathbf{D}}$  est la matrice des degrés du graphe qui aurait  $\mathbf{I}_n + \mathbf{A}$  comme matrice d'adjacence. La ReLU est appliquée terme à terme.

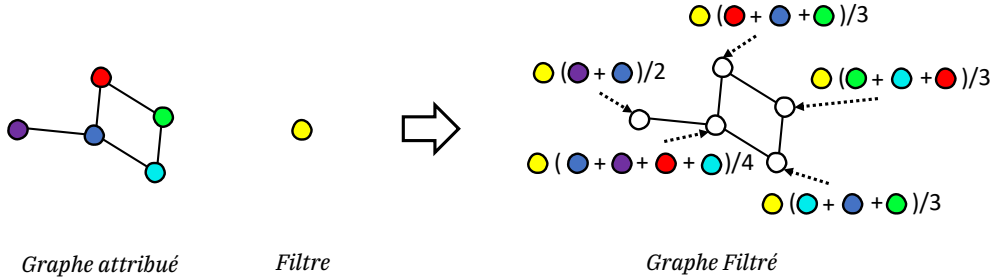


FIGURE 2.11 – *Illustration de l'application de GCN.* Pour simplifier, on illustre le cas où le graphe est attribué par des scalaires et on ne considère qu'un seul filtre (représenté par un nombre). L'application du filtre consiste pour chaque nœud à moyenner le signal dans son voisinage direct puis à le multiplier par le filtre.

### 2.3.5 Deux cas particuliers : Truncated Krylov et Simple GCN

Les GCN ainsi définis ont pu avantageusement être utilisés pour améliorer les résultats sur les *benchmarks* existants en apprentissage sur graphes. L'engouement pour ces réseaux a dynamisé tout un champ de recherche sur les GCN et plus généralement les GNN. De nombreux GNN ont alors émergé pour répondre à différentes problématiques et certaines lacunes des GCN. Dans notre travail, nous avons utilisé en particulier 2 GCN : **Truncated Krylov** et **Simple GCN**.

### 2.3.5.a Truncated Krylov

Bien que la forme du GCN (2.49) découle directement d’une analogie avec les CNN, certaines différences demeurent. Alors que pour le second, l’augmentation de la profondeur de l’architecture est souvent synonyme de meilleures performances (à condition que la quantité de données suive), pour le premier dans la pratique on observe une saturation des performances après 3 à 5 couches. Dans leur article Luan et al. montrent que l’association de GCN avec une ReLU a tendance à rendre les signaux stationnaires, à partir d’une certaine profondeur, favorisant notamment la colinéarité des composantes des signaux, ce qu’on peut interpréter comme une perte d’information dans le réseau. Ils montrent en revanche que l’utilisation de la fonction tangente hyperbolique favorise l’indépendance entre les composantes des signaux traversant le réseau. Ils proposent donc de remplacer la ReLU par la tangente hyperbolique. Dans ce même article Luan et al. montrent également que les GCN peuvent s’obtenir en développant le filtre en série de puissance du laplacien et en tronquant ce développement à l’ordre 1. Ils montrent qu’en prenant des ordres supérieurs de ce développement leur GCN a de meilleures propriétés de conservation de l’information à mesure que le signal traverse l’architecture. L’action de ce GCN à la couche  $K$  s’écrit ainsi :

$$\begin{aligned} \mathbf{X}^{(k+1)} &= \mathbf{Krylov}(\mathbf{X}^{(k)}, \tilde{\mathbf{A}}) \\ &= \tanh([\mathbf{X}^{(k)}, \tilde{\mathbf{A}}\mathbf{X}^{(k)}, \tilde{\mathbf{A}}^2\mathbf{X}^{(k)}, \dots, \tilde{\mathbf{A}}^p\mathbf{X}^{(k)}]\Theta^{(k)}) \end{aligned} \quad (2.50)$$

$p$  est la profondeur d’exploration de ce GCN autour d’un nœud,  $\Theta^{(k)} \in \mathbb{R}^{p \times a^{(k)} \times a^{(k+1)}}$  et  $[\dots]$  est l’opérateur de concaténation. Nous avons choisi ce GCN dans les tâches qui nécessitent de préserver de l’information notamment dans le Chapitre 3 et 4 où l’on construit des modèles qui essaient d’encoder de l’information dans un espace latent.

### 2.3.5.b SGCN

Si les nombreux GCN et GNN de la littérature ont leurs propres spécificités qui peuvent amener à choisir l’un ou l’autre, Wu et al. [2020] montrent qu’en moyenne ils ont des performances similaires. C’est pourquoi Wu et al. [2020] proposent de simplifier plus encore le GCN (2.49). Alors que ce dernier s’écrit à partir de  $\mathbf{X}^{(0)}$  jusqu’à la couche finale  $K$  de la façon suivante :

$$\mathbf{X}^{(K)} = \text{ReLU}(\tilde{\mathbf{A}}(\dots \tilde{\mathbf{A}}\text{ReLU}(\tilde{\mathbf{A}}\text{ReLU}(\tilde{\mathbf{A}}\mathbf{X}^{(0)}\Theta^{(0)})\Theta^{(1)})\Theta^{(2)} \dots)\Theta^{(K-1)}) \quad (2.51)$$

Wu et al. [2020] proposent de supprimer les non linéarités intermédiaires, ce qui revient à écrire :

$$\mathbf{X}^{(K)} = \text{ReLU}(\tilde{\mathbf{A}}^K \mathbf{X}^{(0)} \Theta^{(K)}) \quad (2.52)$$

Où on a posé  $\Theta^{(K)} = \Theta^{(0)}\Theta^{(1)} \dots \Theta^{(K-1)} \in \mathbb{R}^{a^{(0)} \times a^{(K)}}$ . Ce choix permet de réduire la taille des poids entraînés indépendamment de la profondeur  $K$  du réseau. Ils montrent dans leur article que les résultats de ce GCN sont tout aussi bons que ceux des autres GCN voire supérieurs sur certaines tâches. Nous avons choisi ce GCN au Chapitre 5 où le modèle défini a nécessité d’avoir le GCN le plus rapide possible.

### 2.3.6 Conclusion

Dans ce chapitre, nous avons présenté des notations et des notions qui seront rencontrées dans les chapitres suivants de ce manuscrit. Nous avons également eu un bref aperçu des difficultés inhérentes à la nature des graphes notamment pour faire de l'apprentissage. Dans les chapitres qui suivent, nous allons aborder des problématiques très précises qui nous amèneront à construire des modèles basés sur des GCN pour les solutionner. Le chapitre suivant va notamment aborder la problématique d'apprentissage non-supervisée de représentation de graphes.







## Chapitre 3

# Représentation hiérarchique non supervisée de graphes

*Dans ce chapitre, nous présentons un modèle non supervisé que nous appelons Hierarchical Graph2vec – HG2V – de représentations euclidiennes de graphes attribués. Ce modèle vérifie simultanément 3 propriétés importantes, qui en font une alternative crédible aux méthodes de représentations de graphes existantes dans la littérature. Il est inductif, et permet notamment (après entraînement) de calculer les représentations de nouveaux graphes sans nécessité de ré-entraînement. Il est également multi-échelle, et prend en compte à la fois les micro-structures et les macro-structures des graphes pour construire des représentations riches. Enfin, HG2V est différentiable de bout en bout et peut-être utilisé au sein d’un modèle plus large pour construire des représentations spécifiques à la réalisation d’une tâche. Ce modèle se base principalement sur des méthodes issues du traitement du langage naturel, en particulier le negative sampling et sur une méthode de pooling de graphe (Loukas’s coarsening). La combinaison de ces deux outils permet à notre modèle de tenir compte des différentes échelles présentes dans les graphes ; et d’ainsi produire des représentations de qualité, que nous évaluerons dans un nombre varié de tâches. Ce chapitre est basé sur les publications suivantes : [Béthune et al. 2020a,b] faites conjointement à part égale avec Louis Béthune.*

### 3.1 Introduction

Comme nous l’avons vu précédemment, les graphes sont des moyens usités pour représenter des objets et des relations entre ces derniers. Ce sont en effet de bons modèles pour décrire de nombreux types de données, notamment en chimie, biologie, ou encore en informatique. Dans ces domaines, les informations relationnelles sont souvent complétées par des caractéristiques discrètes (appelées aussi étiquettes) ou continues sur les nœuds : on dit alors de ces graphes qu’ils sont attribués. Alors que l’apprentissage automatique propose de nombreuses méthodes pour résoudre des problèmes de classification, de *clustering* ou d’inférence, dès lors qu’un nombre suffisant de données d’entraînement est disponible ; ces méthodes sont essentiellement construites pour traiter des données globalement structurées (i.e euclidiennes) et sont donc incompatibles avec des jeux de données de graphes attribués. Une manière de résoudre cette difficulté consiste

à encoder les graphes attribués en vecteurs euclidiens de manière à ce que les méthodes classiques d'apprentissage soient efficaces sur ces représentations. En d'autres mots, il s'agit de faire de l'apprentissage de représentations de graphes (*graph representation learning* [Hamilton et al. 2017b]) : les graphes sont alors plongés dans un espace euclidien de dimension fixée de telle sorte que les graphes similaires partagent des représentations similaires. Parmi les nombreuses propriétés nécessaires pour qu'une méthode réalisant cette tâche soit efficace, on peut mettre en évidence 3 propriétés importantes :

- **Non-supervisée.** De manière générale que ce soit pour les graphes ou tout autre type de données, obtenir des étiquettes est souvent très coûteux en temps et en argent, elles sont donc régulièrement non disponibles. Il est donc souhaitable d'une méthode de représentation de graphes qu'elle puisse opérer sans étiquette (sur le graphe), cela permet également d'avoir des représentations relativement *générales* et non spécifiques à la réalisation d'une seule tâche.
- **Inductive.** Les bases de données sont dynamiques ; et elles sont de plus en plus volumineuses, de nouveaux éléments viennent s'y ajouter continuellement. Il est impératif qu'on ne soit pas contraint à ré-entraîner le modèle sur l'entièreté de la base de données à chaque nouvel élément entrant, de par le coût temporel, énergétique et environnemental que cela implique. On souhaite donc que la méthode soit inductive et en mesure de calculer les représentations de nouveaux graphes (i.e jamais vus pendant l'entraînement) sans ré-entraînement.
- **Hiérarchique.** Certaines propriétés des graphes peuvent dépendre de certaines échelles en particulier. On souhaite donc que la méthode tienne compte à la fois des échelles locales, globales et intermédiaires présentes au sein des graphes ; lui permettant ainsi d'avoir des représentations riches en informations sur chacune de ces échelles, et utilisables dans des tâches diverses et variées.

Le travail présenté dans ce chapitre est une tentative de conciliation de toutes ces propriétés dans un seul modèle. Nous y présentons une nouvelle méthode d'apprentissage de représentation de graphes *Hierarchical Graph2vec* – HG2V. Ce modèle s'inspire principalement de *Graph2vec* [Narayanan et al. 2017] un modèle de représentation de graphes qui est construit en maximisant l'information mutuelle (via le *negative sampling*) portée par les nœuds des graphes et les représentations en question. Notre contribution consiste en la combinaison de ce principe avec une utilisation judicieuse d'une méthode de *pooling* de graphe (voir Fig. (3.6)) proposée par Loukas and Vandergheynst [2018] qui permet à HG2V de travailler avec une pyramide de graphes attribués représentant le graphe à différentes échelles et ainsi d'incorporer des informations relatives à toutes ces échelles (de la plus petite - nœuds, etc. – à la plus grande – communauté, ponts, etc.) dans la représentation finale.

### 3.1.1 Contexte

Il existe de nombreuses familles de méthodes de représentations de graphes qui s'appuient sur des outils provenant de divers horizons. On peut toutefois distinguer en particulier deux familles de méthodes : celles s'appuyant sur des noyaux et celles s'appuyant sur des réseaux de neurones qui ont connu une certaine popularité ces dernières années, comme nous l'avons déjà mentionné dans les chapitres précédents.

**Méthodes à noyau.** Les méthodes à noyau sont devenues des méthodes de référence [Togninalli et al. 2019; Vishwanathan et al. 2010] (voir Section 2.1.3) dans l'apprentissage de représentations de graphes. Ces méthodes s'appuient sur des mesures de similarités entre les graphes construites *à la main* mais souvent assez génériques. Si certaines de ces mesures de similarités sont limitées aux graphes dont les nœuds portent des caractéristiques discrètes tandis que d'autres peuvent opérer sur des graphes avec des caractéristiques continues, elles sont généralement très performantes et font parties du sommet de l'état de l'art. Toutefois, elles souffrent toutes des mêmes inconvénients : le calcul et le stockage en mémoire de la matrice du noyau est coûteux et limite donc également la scalabilité de ces méthodes, bien qu'il existe des méthodes d'approximation permettant de réduire le coût de calcul [Rahimi and Recht 2008].

**Information mutuelle et *negative sampling*.** Ces dernières années, les méthodes se basant sur des réseaux de neurones [Goodfellow et al. 2016] et un principe de maximisation de l'information mutuelle (*Mutal Information* - MI) [Vergara and Estévez 2015] sont devenus très populaires. L'idée principale exploitée par ces méthodes est qu'une bonne représentation d'un graphe doit maximiser l'information mutuelle entre cette dernière et les composantes du graphe (le plus souvent les nœuds et leurs voisinages); l'objectif de ces méthodes est donc tout simplement de rechercher la représentation qui maximise cette quantité. Des méthodes supervisées populaires comme *Deep Graph Infomax* [Veličković et al. 2019] ou *GrapheSAGE* [Hamilton et al. 2017a] s'appuient sur ce principe pour calculer des représentations de nœuds de graphes. Dans ces articles, les auteurs s'appuient notamment sur le *negative sampling* pour construire un estimateur de la MI, et ainsi la maximiser. Néanmoins, ces articles se limitent à la représentation de nœuds. Pour ce qui est de la représentation de graphes, on peut notamment citer *Graph2vec* [Narayanan et al. 2017] qui est probablement la plus célèbre des méthodes (non-supervisées pour les graphes) s'appuyant sur le *negative sampling* et la MI. Ce modèle est directement inspiré du modèle de représentation des mots du langage naturel, *Word2vec* [Mikolov et al. 2013].

**Extraction de caractéristiques.** *Graph2vec* combine le principe de maximisation de la MI avec la procédure de Weisfeiler-Lehman – WL – (voir Section 3.2.1), qui permet de caractériser les sous-arbres attribués d'un graphe : cette méthode recherche les représentations qui maximisent l'information mutuelle avec les représentations extraites par WL dans le graphe. Ce modèle présente certaines limites; il est non inductif et dévolué uniquement à des graphes dont les nœuds ont des attributs discrets. Malgré ces limites, ce modèle a inspiré de nombreux modèles, on peut en particulier citer *Infograph* [Sun et al. 2020] une méthode semi-supervisée qui se sert également de la MI mais cette fois-ci couplée avec un GCN en lieu et place de WL pour être en mesure de traiter les attributs continus.

Il a été constaté empiriquement que la MI permet d'avoir des représentations de qualité à condition d'avoir une fonction d'extraction de l'information dans les graphes efficace : WL dans la cas de *Graph2vec* ou encore des GCN pour *Infograph*, *Deep Graph Infomax* et *GrapheSAGE*. Le choix de la manière dont on effectue l'extraction d'information est donc crucial pour avoir des représentations de qualité. Si les GCN sont une réponse à certaines des limitations de WL, ils ont également leurs propres limitations. Ces réseaux souffrent de problèmes d'acquisition de l'information à grande échelle dans le graphe. En effet, bien qu'ils soient très efficaces pour résoudre

de nombreuses tâches d'apprentissage relatives aux graphes [Zhou et al. 2021 ; Wu et al. 2020], il a été démontré que certains problèmes *faciles*<sup>1</sup> d'algorithmie classique (comme le calcul du diamètre d'un graphe, la recherche de plus court chemin, la détection de cycle, etc.) ne peuvent pas être résolus par des réseaux de neurones avec une profondeur trop faible [Loukas 2020]. Toutefois, l'empilement de trop nombreuses couches de ces réseaux de neurones peuvent dégrader la capacité d'un modèle à être *scalable*. De plus, on constate expérimentalement qu'après 3 ou 4 couches les performances des GNN plafonnent voire se dégradent [Luan et al.] sur de nombreuses tâches. Pour résoudre ces problèmes d'acquisitions d'informations à large échelle, dans HG2V nous utilisons un GCN que nous combinons à une méthode de *pooling* qui, comme nous allons le voir, permet d'obtenir un modèle plus efficace notamment sur des graphes de grandes tailles.

**Pooling de graphes.** Il existe souvent plusieurs façons de procéder au *pooling* d'un graphe (voir illustration en Figure (3.6) et définition formelle en Section 3.3.1.b). L'algorithme de Loukas and Vandergheynst [2018] que nous avons choisi procède à cette opération en tentant de préserver le spectre des basses fréquences, ce qui permet de préserver les informations topologiques à grande échelle du graphe (voir Section 2.2.1). Toutefois, il existe d'autres principes directeurs pour effectuer cette opération, on peut notamment citer le principe de décimations de sommets liés à la réduction de Kron utilisé par Bianchi et al. [2019] et Dorfler and Bullo [2013]. Ce principe peut notamment être combiné avec d'autres principes d'éliminations de nœuds dans le but de dé-densifier des graphes [Dorfler and Bullo 2013 ; Bravo Hermsdorff and Gunderson 2019]. Les approches automatiques ont également émergées ces dernières années, *Diffpool* Ying et al. [2018] (méthode supervisée) comme son nom l'indique utilise une telle approche, durant l'entraînement de ce modèle un *clustering* automatique de nœuds le long de ses couches est effectué dans le but d'apprendre les fonctions de *pooling* adaptées à la tâche pour laquelle la méthode est entraînée. De même *MinCutPool* Bianchi et al. [2020a] une autre méthode supervisée, s'appuie sur une version relaxée (et différentiable) de la fonction objectif du *MinCut* qui lui permet d'automatiser et adapter le *pooling*. Elle produit ainsi des partitions de (nœuds de) graphes de très bonne qualité.

Dans la suite, nous allons présenter notre méthode HG2V qui se base également sur un principe de maximisation de l'information mutuelle, entre l'information extraite de nos graphes et leurs représentations. Notre méthode s'appuiera notamment sur une méthode efficace d'extraction de l'information à toutes les échelles du graphe grâce à l'utilisation conjointe d'un GCN et de la technique de *pooling* de Loukas. Ensuite, le *negative sampling* sera alors utilisé pour incorporer ces informations dans les représentations des graphes. Dans une première section, nous présenterons les travaux sur lesquels s'appuient directement notre travail, notamment WL, la MI, le *negative sampling* et *Word2vec* et nous y introduirons quelques notations supplémentaires. Ensuite, une section sera dédiée à la présentation de notre modèle et ses propriétés. Puis une section finale détaillera les différentes expériences effectuées pour démontrer l'efficacité de notre modèle sur diverses tâches.

---

1. Soluble en temps polynomial.

Méthodes	Attributs (continus)	Complexité (entraînement)	Complexité (inférence)	Différentiable (de bout-en-bout)	Supervisée
WL-OA, WWL	✓	$O(n^2)$ <sup>2</sup>	$O(n)$	✗	✗
<i>Graph2vec</i>	✗	$O(n)$	✗	✗	✗
GIN, <i>DiffPool</i> <i>MinCutPool</i>	✓	$O(n)$	$O(1)$	✓	✓
HG2V, <i>Infograph</i>	✓	$O(n)$	$O(1)$	✓	✗

TABLE 3.1 – *Propriétés principales des méthodes de représentations de graphes*. Le symbole ✗ pour la colonne **complexité (inférence)** signifie que la méthode est transductive<sup>4</sup> (et non inductive) et le temps pour produire les représentations est celui de l’entraînement. Le symbole ✓ pour la colonne **supervisée** signifie que les étiquettes sont nécessaires pour apprendre les représentations (il s’agit essentiellement de faire de la rétro-propagation arrière à partir d’une fonctionnelle de classification).

## 3.2 Définitions et contexte

Nous allons ici redonner les éléments et méthodes de l’état de l’art à la fois pour présenter la méthode et discuter le tableau 4.3.2 qui présente les points forts et faibles de l’état de l’art. Notre méthode s’inspire fortement de *Graph2vec* [Narayanan et al. 2017] qui s’appuie sur l’algorithme de Weisfeiler-Lehman [?] qui est une manière efficace de produire des descripteurs de la topologie d’un graphe. Ces descripteurs permettent de produire des représentations de qualités pour les graphes à travers le principe de maximisation de l’information mutuelle entre les descripteurs et les représentations des graphes associés. Pour maximiser cette information mutuelle, on s’appuie sur le *negative sampling* qui permet de construire un estimateur de cette grandeur. Cette section expose les idées fondamentales à l’origine de Weisfeiler-Lehman et du *negative sampling*.

Les notations utilisées pour décrire les graphes seront les mêmes que celles définies en Section 2.1.1. On travaillera dans ce chapitre avec des graphes **attribués et pondérés**  $\mathcal{G} = (V, E)$  associés à des matrices d’adjacences  $\mathbf{A} \in \mathbb{R}^{n \times n}$ . On rappelle que  $n$  est la taille du graphe, un nœud du graphe  $\mathcal{G}$  est noté  $u_i \in \mathcal{V}$ ; et l’attribut qu’il porte est noté  $x(u_i) \in \mathbb{R}^q$ . On note  $q$  la dimension de ces attributs. Notre modèle se caractérise par une certaine profondeur  $L$ , à chaque étape  $l$  est associé un graphe (qui découle de  $\mathcal{G}$  via le *pooling*) qui sera noté  $\mathcal{G}^{(l)} = (V^{(l)}, E^{(l)})$  et la fonction attributrice  $x^{(l)}$ . De manière générale, toutes grandeurs évoluant en fonction de l’étape  $l$  seront mises à l’exposant  $(l)$ , pour spécifier à quelle étape elles se réfèrent. On remarquera que  $\mathcal{G}^{(0)} = \mathcal{G}$ .

On notera  $\mathbb{G}_x$ , l’ensemble des graphes attribués d’un jeu de données quelconque. L’objectif de notre travail est de déterminer une fonction de représentation  $\mathcal{R} : \mathbb{G}_x \rightarrow \mathbb{R}^d$  qui à chaque graphe attribué associe un vecteur dans l’espace latent euclidien  $\mathbb{R}^d$  pour un entier  $d$  strictement positif, la fonction étant telle que ces vecteurs facilitent les tâches d’apprentissage sur ces graphes (voir Fig. (3.1)).

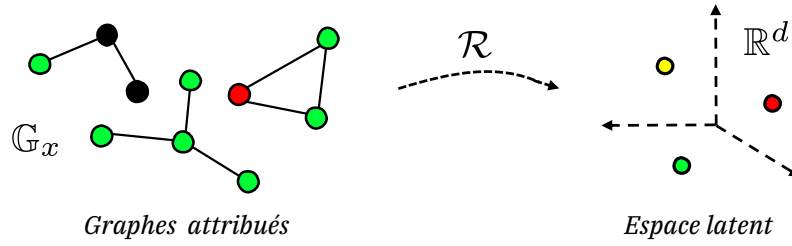


FIGURE 3.1 – Schéma de l'apprentissage de représentations de graphes.

### 3.2.1 Algorithme de Weisfeiler-Lehman - WL

La procédure de Weisfeiler-Lehman [?] a été pensée à l'origine pour résoudre le problème difficile de l'isomorphisme de graphes (voir Section 2.1.2). Cette procédure consiste, étant donné un graphe avec des nœuds portant des attributs discrets (si le graphe n'en comporte pas, on peut donner le même attribut à tous) à générer toute une série de graphes étiquetés via l'application successive du même opérateur déterministe (voir Fig. (3.2)). Les histogrammes d'étiquettes ainsi générés pour les deux graphes permettent de distinguer si les deux graphes sont similaires. Toutefois, il existe des exemples où des graphes différents produisent les mêmes histogrammes (voir Annexe A.2.1). Malgré ces contre-exemples, c'est une méthode très efficace pour discriminer les graphes, elle peut également être employée pour construire des noyaux pour l'apprentissage sur graphes très efficaces [Shervashidze et al. 2011]. Formellement, la procédure pour générer les étiquettes de chaque nœud  $u_i$  s'écrit :

$$\mathbf{x}^{(0)}(u_i) = \mathbf{x}(u_i) \quad (3.1)$$

$$\mathbf{x}^{(l+1)}(u_i) = \text{hash}\left(\{\mathbf{x}^{(l)}(u_j) | u_j \in \mathcal{V}(u_i)\}\right) \quad (3.2)$$

où on le rappelle  $\mathcal{V}(u_i)$  est le voisinage des plus proches voisins de  $u_i$  ( $u_i$  inclu). Cette procédure est itérée jusqu'à une profondeur  $L$  (à définir par l'utilisateur). Plus  $L$  est choisi grand, plus large est le voisinage considéré pour distinguer les graphes<sup>5</sup> car la  $l$ -ième étiquette de  $u_i$ ,  $\mathbf{x}^{(l)}(u_i)$  dépend uniquement des étiquettes à une distance au plus  $l$ . En d'autres termes, l'étiquette  $\mathbf{x}^{(l)}(u_i)$  caractérise le sous-arbre de sommet  $u_i$  et de profondeur  $l$ . La fonction *hashing* de cette procédure est injective notamment pour pouvoir distinguer les différents arbres enracinés que constituent les voisinages. On remarquera en particulier que c'est une fonction de l'ensemble des étiquettes du voisinage. Cette fonction produit donc un résultat indépendant de l'ordre de ces derniers, comme deux graphes isomorphes ont les mêmes sous-arbres à une permutation près, cette procédure produira les mêmes étiquettes pour de tels graphes.

**Remarque 9 (Nombre d'étiquettes.)** Cette procédure produit après  $L$  itérations,  $nL$  caractéristiques.

5. Toutefois, à partir d'un certain nombre d'itérations ( $\leq n$ ), le processus sature et ne donne pas plus d'informations.

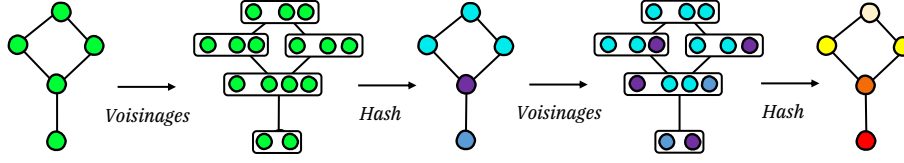


FIGURE 3.2 – **Weisfeiler-Lehman**. Deux itérations de la procédure de Weisfeiler-Lehman, à partir d'un graphe avec un étiquetage uniforme.

### 3.2.2 *Negative Sampling* et Information Mutuelle

De nombreux problèmes d'apprentissage se réduisent à déterminer une distribution  $p(B|A)$  en ayant seulement des exemples  $(\mathbf{a}, \mathbf{b}) \sim p(A, B)$  échantillonnés selon la distribution jointe. Ici typiquement,  $A$  est un ensemble de données quelconques et  $B$  l'ensemble de leurs étiquettes. En général, ce type de problème requiert l'usage de la fonction *softmax*, ce qui implique le calcul coûteux de la constante de normalisation de cette dernière. C'est impossible lorsque le domaine de  $B$  est trop grand (par exemple dans le contexte du langage naturel, où  $B$  est un vocabulaire). Pour surmonter cette difficulté, *Word2vec* [Mikolov et al. 2013] remplace ce problème par un problème de classification binaire, où le classifieur apprend à distinguer entre des paires réelles  $(\mathbf{a}, \mathbf{b}) \sim p(A, B)$  et de fausses paires  $(\mathbf{a}, \mathbf{b}) \sim p(A) \otimes p(B)$  (i.e  $\mathbf{a} \sim p(A)$  et  $\mathbf{b} \sim p(B)$ ) échantillonnées selon la distribution produit. Ainsi, au lieu de modéliser la distribution conditionnelle  $p(B|A)$ , un discriminateur binaire apprend à distinguer les paires provenant de la distribution  $p(B, A)$  et celles provenant de la distribution  $p(B) \otimes p(A)$ . Ce changement de paradigme qui *débloque* le problème se fait au coût d'un modèle de classification moins puissant mais très efficace pour faire de l'apprentissage de représentations. Formellement, ce principe se réalise en maximisant la quantité suivante :

$$\widehat{\mathcal{I}}^{(\text{JSD})}(A, B) = -\mathbb{E}_{(\mathbf{a}, \mathbf{b}) \sim P(A, B)} \log \tau(T_{\theta}(\mathbf{a}, \mathbf{b})) - \mathbb{E}_{(\mathbf{a}, \mathbf{b}) \sim P(A) \otimes P(B)} \log \tau(-T_{\theta}(\mathbf{a}, \mathbf{b})) \quad (3.3)$$

où  $T_{\theta}$  est le discriminateur à valeur dans  $\mathbb{R}$ , i.e une fonction paramétrée par  $\theta$  et où typiquement  $\tau : y \mapsto \frac{1}{1+e^{-y}}$  représente la fonction sigmoïde.  $\tau(T_{\theta}(\mathbf{a}, \mathbf{b}))$  est la probabilité paramétrée que la paire  $(\mathbf{a}, \mathbf{b})$  soit issue d'un échantillonnage de  $p(A, B)$ . Comme on peut le voir, pour calculer cette quantité, il faut tirer de fausses paires selon  $p(A) \otimes p(B)$ , d'où le nom de *negative sampling* donné à cette méthode.

La quantité définie en Equation (3.3), est en fait l'estimateur de Jensen-Shannon de l'information mutuelle  $\mathcal{I}$  entre  $A$  et  $B$  (voir [Hjelm et al. 2019] pour le premier usage dans ce contexte de cet estimateur, voir [Nowozin et al. 2016] pour sa construction et pour d'autres détails [Melamud and Goldberger 2017]). Cette quantité est issue de la théorie de l'information, c'est la divergence de Kullback-Leibler entre la distribution jointe et la distribution produite, ce qui s'écrit dans le cas de distributions discrètes :

$$\mathcal{I}(A, B) = \sum_{\mathbf{a} \in A, \mathbf{b} \in B} P(\mathbf{a}, \mathbf{b}) \log \frac{P(\mathbf{a}, \mathbf{b})}{P(\mathbf{a})P(\mathbf{b})} \quad (3.4)$$



Elle quantifie la dépendance statistique entre deux variables. Ainsi, si on paramétrise les éléments de  $B$  par des paramètres  $\lambda$  et qu'on note cet ensemble  $B_\lambda$ , alors rechercher la paramétrisation des éléments de  $B_\lambda$  qui maximise l'information mutuelle entre  $A$  et  $B_\lambda$  revient à résoudre le problème suivant :

$$\max_{\lambda} \mathcal{I}(A, B_\lambda) \approx \max_{\lambda} \widehat{\mathcal{I}}^{(\text{JSD})}(A, B_\lambda) \quad (3.5)$$

Ce principe est utilisé dans *Word2vec* pour déterminer les représentations euclidiennes de mots d'un corpus de texte. Dans ce cadre, on constitue à partir d'un corpus de textes, deux jeux de données ;  $B$  qui sont les mots du vocabulaire de ce corpus et  $A$  les *sacs de mots* [Mikolov et al. 2013] entourant les mots de  $B$  dans le corpus, i.e., il s'agit du contexte du mot. Ici, une paire positive signifie qu'un mot est bien associé à son contexte (voir Fig. (3.3)). Pour calculer les représentations des mots dans  $B$ , on introduit l'ensemble des représentations  $B_\theta$  où chaque mot  $m$  de  $B$  est représenté par un vecteur de poids  $\theta_m \in \mathbb{R}^d$  entraînable. Et de même, on introduit  $A_\theta$  où chaque contexte  $c$  de  $A$  est représenté par un vecteur de poids  $\theta_c \in \mathbb{R}^d$ . On souhaite alors que l'information mutuelle entre les représentations des mots  $B_\theta$  et leur contexte  $A_\theta$  soit maximale i.e :

$$\max_{\theta} \widehat{\mathcal{I}}^{(\text{JSD})}(A_\theta, B_\theta) \quad (3.6)$$

L'idée brillante sous-jacente à ce modèle c'est de comprendre que les mots et les ensembles de mots (le contexte) peuvent être représentés dans le même espace. En effet, ce n'est pas tant le mot en lui-même qu'on souhaite représenter mais son sens. Or, le sens peut être porté par un mot tout comme par toute une série de mots. En maximisant la MI, on force le modèle à déterminer un espace latent cohérent dans lequel le sens peut-être porté par un ou plusieurs mots. Ce principe fonctionne excessivement bien, notamment car ce modèle est naturellement fortement contraint : le sens des mots dépend de leur contexte d'utilisation et inversement le sens du contexte dépend des mots qui le constituent. En définitif, *Word2vec* est un très bon modèle de représentation de mots, qu'on peut partiellement transposer pour les graphes, comme nous allons le voir.

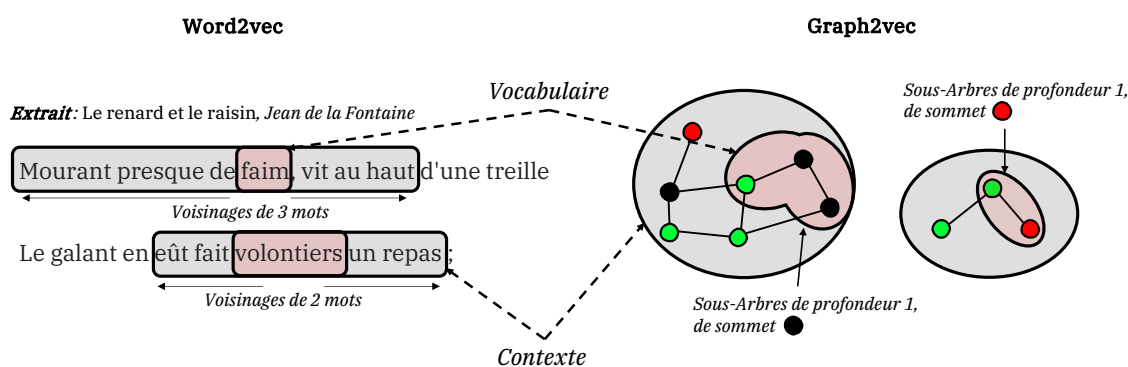


FIGURE 3.3 – *Analogie entre Word2vec et Graph2vec.*

**Remarque 10 (Divergence de Kullback-Leibler.)** *La divergence de Kullback-Leibler est très utilisée en apprentissage automatique. Elle permet étant donné deux distributions  $p, q \in \mathcal{P}(\mathbb{X})$  d'évaluer la similarité entre les deux distributions. Lorsque  $\mathbb{X}$  est un espace discret, elle s'exprime ainsi :*

$$D_{\text{KL}}(p\|q) = \sum_{x \in \mathbb{X}} p(x) \log \frac{p(x)}{q(x)}. \quad (3.7)$$

Et lorsque  $\mathbb{X}$  est continue :

$$D_{\text{KL}}(p\|q) = \int_{\mathbb{X}} p(x) \log \frac{p(x)}{q(x)} dx. \quad (3.8)$$

Cette grandeur possède d'excellentes propriétés qui en font un outil de choix pour construire des fonctionnelles d'apprentissage. Elle est positive et ne s'annule que lorsque que les deux distributions sont égales. Dans l'Equation (2.44) on peut notamment l'utiliser comme fonction de comparaison  $l$ . On pourra se référer à [Goodfellow et al. 2016] pour plus de détails à son sujet.

**Remarque 11 (État de l'art en traitement du langage naturel.)** *Cependant ce type de technique ne constitue plus l'état de l'art du traitement du langage naturel. L'état de l'art actuel consacre les transformeurs [Vaswani et al. 2017] qui sont une évolution des RNN ; permettant notamment de paralléliser l'entraînement (ce qui n'est pas possible avec les RNN qui s'appliquent séquentiellement) et qui grâce à des mécanismes d'attentions permettent une analyse plus fine du contexte d'un mot et in fine à de meilleures représentations pour les mots.*

### 3.2.3 Graph2vec

Les deux idées présentées plus haut WL et le *negative sampling* sont combinées par Narayanan et al. [2017], fortement inspiré par *Word2vec*, pour produire des représentations de graphes. Dans ce cadre là, le contexte évoqué plus haut dans la description de *Word2vec* correspond aux graphes i.e  $A = \mathbb{G}_x$  (voir Fig. (3.3)). Ce *contexte* est composé des mots du vocabulaire des graphes, c'est à dire l'ensemble des sous-arbres  $B$  (de profondeur  $\leq L$ ) des graphes dans  $A$ . Ici, une paire positive correspond donc à un graphe de  $A$  associé à l'un de ses sous-arbres dans  $B$ . Tout comme dans *Word2vec*, on introduit  $A_\theta = \mathcal{R}_\theta(\mathbb{G}_x)$  un ensemble dans lequel on a associé à chaque graphe  $\mathcal{G}$  de  $A$ , un vecteur de poids entraînable  $\theta_{\mathcal{G}} \in \mathbb{R}^d$ . Par ailleurs, les *mots*, c'est-à-dire les sous-arbres, eux sont tout simplement associés à leurs caractéristiques données par Weisfeiler-Lehman. En effet, l'arbre de sommet  $u_i$  et de profondeur  $l$  est caractérisé par WL avec  $\mathbf{x}^{(l)}(u_i)$  (voir Eq. (3.2)), alors on note l'ensemble de ces caractéristiques  $B = \text{WL}(\mathbb{G}_x) = \{\mathbf{x}^{(l)}(u_i) | \forall u_i \in V, l \in \{0, \dots, L-1\}\}$  et  $B_\theta = \text{WL}_\theta(\mathbb{G}_x) = \{\text{encodage}(\mathbf{x}) | \forall \mathbf{x} \in B\}$  l'ensemble des encodages<sup>6</sup> de ces caractéristiques.

6. Par exemple, comme les caractéristiques de WL sont discrètes. On peut utiliser un encodage *one-hot*, il s'agit d'un encodage qui associe à une variable qui peut prendre  $c$  états un vecteur de taille  $c$  avec des 0 sur chaque composante sauf une où on y trouve un 1. Il est en général utilisé pour encoder des caractéristiques discrètes dans un jeu de données, dans ce cadre  $c$  correspond au nombre de caractéristiques discrètes différentes extraites par WL soit le nombre de sous-arbres différents. De plus, comme en général  $d \neq c$ , on peut ensuite utiliser une projection aléatoire pour que les éléments de  $B$  soit dans  $\mathbb{R}^d$ , cela marche assez bien en pratique.

Pour calculer les représentations des graphes, on maximise alors l'information mutuelle entre les représentations des sous-graphes et les représentations des graphes auxquelles ils appartiennent, ce qui nous donne le problème suivant :

$$\max_{\theta} \widehat{\mathcal{I}}^{(\text{JSD})}(\mathcal{R}_{\theta}(\mathbb{G}_x), \text{WL}_{\theta}(\mathbb{G}_x)) = \mathbb{E}_{(\mathcal{G}, t) \sim P(\mathbb{G}_x, \text{WL}(\mathbb{G}_x))} \log \tau(\boldsymbol{\theta}_{\mathcal{G}} \cdot \boldsymbol{\theta}_t) \quad (3.9)$$

$$+ \mathbb{E}_{(\mathcal{G}, t) \sim P(\mathbb{G}_x) \otimes P(\text{WL}(\mathbb{G}_x))} \log \tau(-\boldsymbol{\theta}_{\mathcal{G}} \cdot \boldsymbol{\theta}_t) \quad (3.10)$$

Le discriminateur  $T_{\theta}(\mathcal{G}, t) = \boldsymbol{\theta}_{\mathcal{G}} \cdot \boldsymbol{\theta}_t$  est choisi comme étant le produit scalaire entre les représentations de graphes  $\boldsymbol{\theta}_{\mathcal{G}} \in \mathbb{R}^d$  qui sont des vecteurs initialisés aléatoirement et  $\boldsymbol{\theta}_t \in \mathbb{R}^d$  un encodage des caractéristiques produites par WL. A la fin de l'optimisation,  $\mathcal{R}(\mathcal{G}) = \boldsymbol{\theta}_{\mathcal{G}}^*$ . L'optimisation de cette fonction nous assure que l'information mutuelle entre les étiquettes de graphes produites par WL et entre les représentations de graphes est maximale ce qui correspond en quelque sorte, à compresser l'information de la distribution des étiquettes de WL dans la représentation.

### 3.3 Hierarchical Graph2vec - HG2V

Dans cette partie, nous allons présenter notre modèle et ses propriétés.

- Comme nous allons le montrer, la procédure de WL a des difficultés à capturer des informations à l'échelle globale du graphe, ce qui constitue une limite pour de nombreuses tâches. Cette limite peut-être surmontée à l'aide du *pooling* de graphes. Nous avons fait le choix d'utiliser la méthode de Loukas pour effectuer cette opération.
- Dans notre modèle, nous montrons aussi l'avantage qu'il y a à remplacer WL par un GNN. Cela permet de traiter les graphes avec des caractéristiques continues et d'avoir une sensibilité moindre aux petites variations non significatives entre les graphes d'un même jeu de données, grâce aux propriétés de continuité de ces derniers. Ainsi, deux graphes proches en termes de structure et d'attributs auront des caractéristiques extraites par le GNN proches et *in fine* des représentations proches dans l'espace latent.

En se basant sur ces observations, nous allons proposer une nouvelle méthode construite autour de la méthode de *pooling* de Loukas et du principe de maximisation de l'information mutuelle entre représentations et caractéristiques extraites par le GNN ; qu'on nommera *Hierarchical Graph2vec* - HG2V. Une version *haut-niveau* de cet algorithme peut être trouvée en [3.3.4.a](#). Comme on pourra le constater, *Hierarchical Graph2vec* s'appuie fortement sur Graph2vec, mais il corrige nombre de ces défauts ; aussi il admet les propriétés suivantes :

- L'entraînement de ce modèle est non supervisé, et ne nécessite donc pas d'étiquette sur les graphes. Les représentations qu'il permet d'obtenir peuvent donc être utilisées dans de nombreuses tâches différentes.
- Ce modèle est inductif, on peut l'entraîner en temps linéaire, puis calculer les représentations de nouveaux graphes jamais vus pendant l'entraînement avec une complexité temporelle quasiment constante.
- Le modèle est capable de traiter les graphes avec des attributs continus, notamment via le remplacement de la fonction de *hashing* dans la procédure de WL (Eq. (3.2)) par un GNN.

- Le modèle est différentiable de bout en bout. Son entrée et sa sortie peuvent être connectées à d'autres réseaux de neurones, HG2V peut ainsi être utilisé au sein d'un modèle plus large. Le gradient peut notamment être rétro-propagé de la fonctionnelle du modèle jusqu'aux couches extractrices des caractéristiques. Ce qui permet d'envisager l'utilisation d'étiquettes ; pour les graphes lorsqu'elles sont disponibles, la possibilité de faire du *transfer learning* ou encore l'utilisation des couches d'extractions de caractéristiques plus performantes (juste avant l'application des GNN) lorsque les caractéristiques des nœuds sont d'un type particulier (images, sons, vidéos, etc.) comme des CNN par exemple.
- Le modèle est multi-échelle, et tient compte des informations à toutes les échelles du graphe notamment grâce à la réduction de Loukas et la MI, lui permettant ainsi de générer des représentations relativement *générales*, adaptées à plusieurs types de tâches.

La Table 3.1.1 reprend certaines des propriétés de la méthode vis-à-vis d'autres méthodes de la littérature. Dans la suite de cette section, nous allons d'abord présenter la méthode de *pooling* de Loukas [2020] et comment elle est utilisée et justifiée dans notre travail. Ensuite, nous entamerons une discussion sur les propriétés de continuité des GNN et leurs intérêts dans le cadre de ce travail, puis nous concluons cette section en décrivant précisément comment notre modèle intègre ces éléments et comment il est entraîné.

### 3.3.1 Méthode de *pooling* de Loukas

Dans cette partie, on discutera les principaux défauts de la procédure de Weisfeiler-Lehman et l'intérêt de la réduction de graphes dans ce cadre. Dans un souci de simplification, nous allons pour le moment ignorer les attributs de nos graphes et nous intéresser exclusivement à leurs structures. Nous montrerons que même dans ce scénario simplifié, WL n'est pas capable d'évaluer quantitativement la proximité entre graphes.

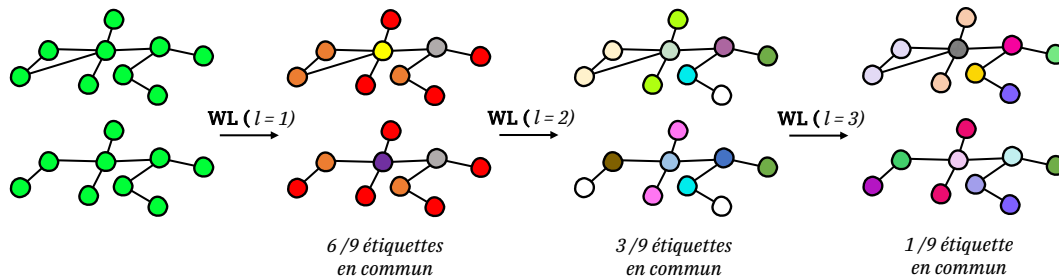


FIGURE 3.4 – *Weisfeiler-Lehman sur deux graphes structurellement proches.*

#### 3.3.1.a Sensibilité de Weisfeiler-Lehman aux perturbations locales

L'excellente capacité de WL à discriminer les graphes, est aussi la cause de son incapacité à reconnaître que deux graphes sont très proches. Dans la pratique, dans un jeu de données de graphes, on trouvera très rarement des graphes possédant la même structure mais on pourra retrouver des graphes similaires, sensiblement différents et partageant donc les mêmes propriétés. Or, un seul sommet retiré ou ajouté peut très fortement changer l'histogramme des étiquettes

générees par Weisfeiler-Lehman (voir Fig. (3.4)). Autrement dit, WL n'est pas une méthode adaptée pour quantifier la proximité entre des graphes. Pour étayer notre propos, nous avons fait quelques expériences pour évaluer l'effet de la suppression de quelques nœuds sur 4 familles de graphes : des cycles, des arbres, des roues, et des échelles (définies en Section 2.1.1.a). Il s'agit de 4 familles de graphes qui sont représentatives de structures qu'on peut retrouver dans les jeux de données usuels notamment les cycles et les arbres, mais aussi des roues pour représenter des nœuds très fortement connectés, ou des échelles pour représenter des groupements de nœuds reliés de manière non dense. Étudier l'impact de la suppression de nœuds sur ces structures nous donnerons une indication de l'impact de cette action sur des jeux de données usuels. On notera que les échelles et cycles présentent le diamètre le plus important (de l'ordre du nombre de nœuds), alors que les roues présentent un diamètre faible (toujours égale à 3). Quant aux arbres leurs diamètres dépendent de leur nature ; les arbres construits pour cette expérience ont un diamètre de l'ordre du logarithme de leur nombre de nœuds.

Pour réaliser notre expérience, on a aléatoirement construit 100 graphes de 500 nœuds et pour chacune de ces familles<sup>7</sup>, on a progressivement (et aléatoirement) retiré 1 à 10 nœuds. Ensuite, on a comparé les étiquettes générées par l'application de la procédure de WL pour chacun de ses graphes partiellement amputé (pour toutes les étapes de WL jusqu'à 5) à son graphe d'origine. Les histogrammes d'étiquettes sont ensuite comparés via le score de similarité suivant qui correspond à un pourcentage de similarité entre les histogrammes :

$$\mathcal{S}^{(l)}(\mathcal{G}, \mathcal{G}') = 100 \frac{|\text{histogramme}(WL(\mathcal{G}^{(l)})) \cap \text{histogramme}(WL(\mathcal{G}'^{(l)}))|}{|\text{histogramme}(WL(\mathcal{G}^{(l)})) \cup \text{histogramme}(WL(\mathcal{G}'^{(l)}))|} \quad (3.11)$$

Lorsque les histogrammes sont strictement identiques (resp. différents en tout point), cette quantité vaut 1 (resp. 0). Les scores moyens  $\mathcal{S}^{(l)}(\mathcal{G}, \mathcal{G}')$  mesurés sur les 100 graphes (par famille de graphes) sur lesquels cette expérience a été menée peuvent être observés en Figure 3.5. On peut y voir que le score de similarité décroît avec le nombre d'arêtes retirés ; et ce dès la première étape de WL. La dégradation de cette similarité est variable selon le type de graphe considéré, mais pour tous l'effet est d'autant plus marqué que le nombre d'étapes de WL augmente. A la 5e étape, on obtient avec surprise des scores de similarités allant de 70% à 20%, voire 0% alors que les expériences mettent en jeu des graphes de plus de 500 nœuds qui diffèrent de moins d'une dizaine de nœuds. Une petite perturbation locale change donc totalement la nature de l'histogramme des caractéristiques produites par WL. S'il existe des cas où l'absence d'un nœud justifie une forte dissimilarité entre des graphes, par exemple lorsque ce nœud connecte deux communautés ; dans ces expériences on a mesuré l'impact moyen du retrait d'un nœud *quel qu'il soit*, les résultats obtenus prouvent que la méthode n'est pas suffisamment fine pour distinguer les nœuds et les éléments d'importances dans un graphe. On peut remarquer que les effets d'un changement local, sont d'autant plus impactant que le diamètre du graphe est faible, i.e les nœuds sont proches les uns des autres. En effet, pour ces graphes, une perturbation locale se

7. Les cycles, échelles et roues sont strictement déterminés par leur taille ; les mêmes graphes sont donc générés à chaque fois. La variabilité dans l'expérience provient des différentes configurations de nœuds retirés à chaque essai. Quant aux arbres, ils sont générés en prenant un nœud puis en lui ajoutant 3 nœuds fils, auxquels on ajoute de nouveau 3 autres nœuds fils et ainsi de suite jusqu'à obtenir un graphe de taille 500. Le même arbre est donc généré à chaque fois. La librairie networkx a été utilisée pour générer ces graphes.

propagera beaucoup plus rapidement à tous les sommets du graphe. C'est pourquoi les familles de graphes telles que les roues et les arbres sont les plus impactées dans nos expériences. En résumé, cette expérience permet d'illustrer l'incapacité de la procédure de WL à caractériser les informations à grande échelle et à évaluer justement la proximité de deux graphes l'un par rapport à l'autre.

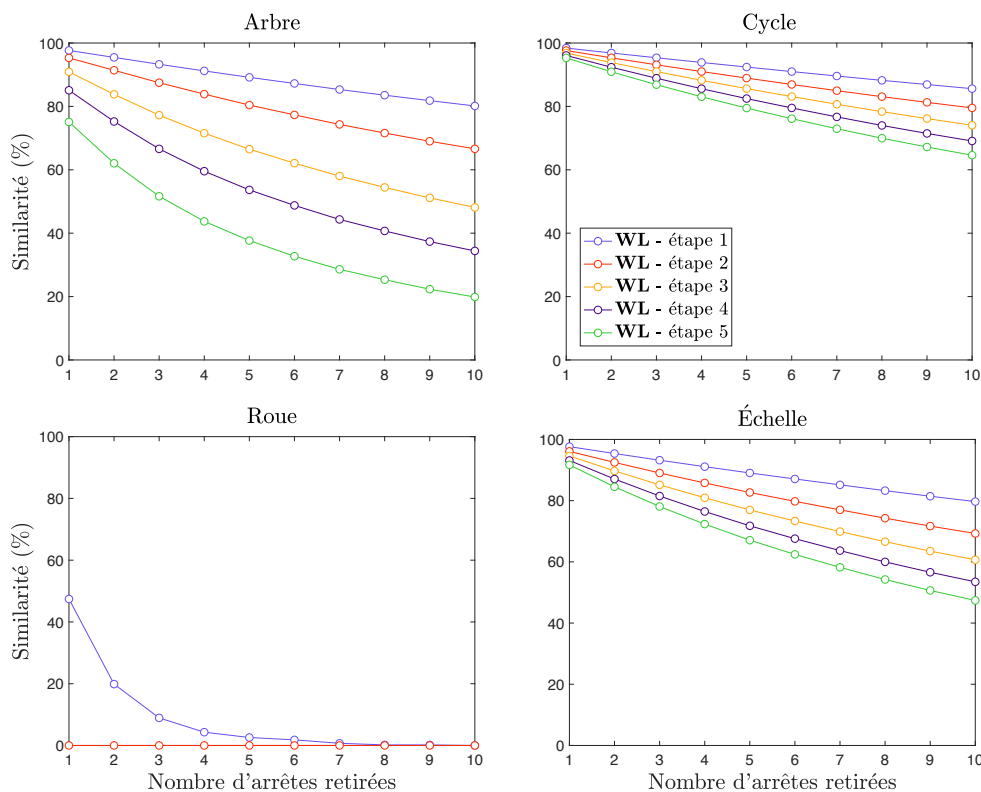


FIGURE 3.5 – *Weisfeiler-Lehman face aux bruits structurels*. Les résultats expérimentaux relatés sur cette figure sont décrits en Section 3.3.1.a.

### 3.3.1.b Robustesse aux perturbations locales avec le *pooling* de Loukas

Le *pooling* consiste à regrouper des nœuds adjacents dans un graphe de sorte à former des *super* nœuds reliés entre eux si et seulement si, il existait un lien entre les nœuds dont ils sont issus. L'intérêt de cette opération est qu'elle permet de travailler avec des graphes beaucoup plus petits avec une perte minimale d'information, les méthodes de *pooling* sur les données euclidiennes associées aux CNN ont connu de nombreux succès l'avènement des GCN a donc été de paire avec un intérêt accru de la communauté d'apprentissage pour les techniques de *pooling* sur graphes. La méthode de Loukas est une de ces techniques (dont la définition formelle peut être trouvée ci-dessous), elle permet notamment de travailler avec des versions réduites des graphes

qui conservent certaines propriétés globales des graphes comme la présence de communauté, de *bridges*, etc.

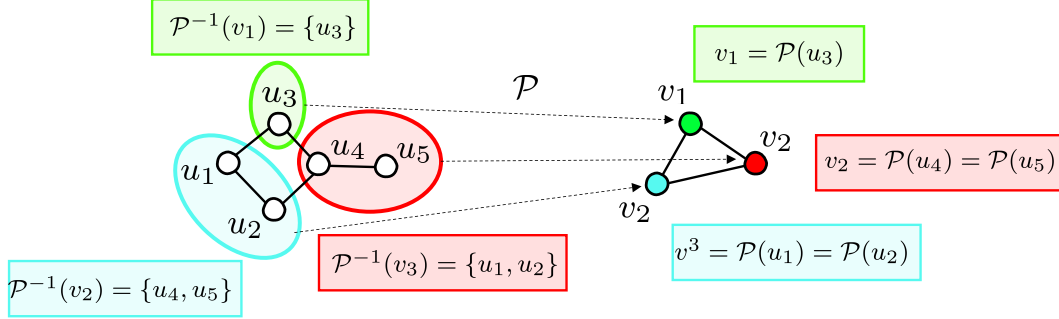


FIGURE 3.6 – Une itération de pooling.

**Definition 1 (Pooling de graphes)** *Le pooling de graphes [Loukas and Vandrgheynst 2018] est une opération qui à un graphe  $\mathcal{G}_1 = (V_1, E_1)$  associe un autre graphe  $\mathcal{G}_2 = (V_2, E_2)$  vérifiant  $|V_2| < |V_1|$ , en usant d'un homomorphisme (voir Section 2.1.1.c) de graphe surjectif  $\mathcal{P} : V_1 \rightarrow V_2$  nommé **fonction de pooling** :*

$$\begin{aligned} (u_i, u_j) \in E_1 &\implies (\mathcal{P}(u_i), \mathcal{P}(u_j)) \in E_2 \text{ or } \mathcal{P}(u_i) = \mathcal{P}(u_j) \\ (\mathcal{P}(u_i), \mathcal{P}(u_j)) \notin E_2 &\implies (u_i, u_j) \notin E_1 \end{aligned} \quad (3.12)$$

Comme on peut le voir, la définition ci-dessus donne une certaine liberté d'action pour la fonction de  $\mathcal{P}$ . La méthode de réduction de Loukas choisit une **fonction de pooling** parmi celles admissibles, qui offre des garanties théoriques sur le spectre des graphes produits lors du *pooling*. Elle est construite de telle sorte à préserver les basses fréquences du spectre ; ainsi les basses fréquences du spectre du graphe réduit sont *proches* de celles du graphe initial, ou du moins encadrées par celles-ci. Or, comme évoqué en Section 2.2.1, ces quantités caractérisent les propriétés topologiques globales des graphes. La méthode de Loukas a donc tendance à préserver les propriétés globales des graphes. La Figure 3.7 permet de constater visuellement l'efficacité de la méthode sur un exemple.

Comme on a pu le voir, une différence d'un petit nombre de nœuds ou d'arêtes entre graphes peut complètement bouleverser les caractéristiques extraites par Weisfeiler-Lehman. C'est parce que WL caractérise les graphes comme une collection de sous-arbres de profondeurs inférieures à  $L$  d'importances égales. Le moindre changement local implique qu'une plus ou moins grande partie des sous-arbres soient modifiée ce qui induit des caractéristiques globales différentes. A contrario, la méthode de *pooling* de Loukas permet de décrire le graphe comme une suite de graphes (décroissant en la taille des nœuds), dont les nœuds ont été fusionnés, ce qui équivaut à supprimer certains arbres dans notre graphe. Ainsi à chaque étape de *pooling*, les arbres les moins importants du point de vue de la structure globale sont supprimés. On s'attend donc à ce qu'une petite perturbation concernant un des ces arbres ne change pas fortement la pyramide de sous-graphes qui décrit notre graphe, les graphes proches avec des différences locales minimales, seront décrits par une pyramide de graphes semblables. Seules certaines perturbations

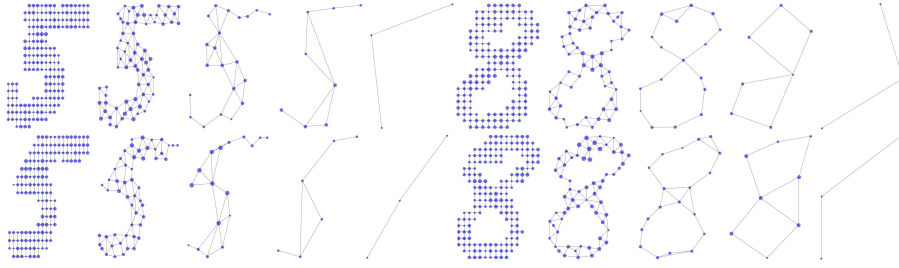


FIGURE 3.7 – *Pyramide de graphes poolés via la méthode de Loukas*. Les 4 graphes sont issus du jeu de données d’images représentant des chiffres écrits à la main convertis en graphes (voir Section 3.4.1). Plus un nœud est volumineux, plus il provient d’un regroupement important de nœuds.

locales impactant la structure globale sont susceptibles de changer la pyramide de sous-graphe (par exemple des arbres connectant des communautés). Cette pondération de l’importance des sous-arbres est un comportement beaucoup plus désirable que celui de WL pour faire de la représentation de graphes.

De manière plus formelle, on souhaite qu’étant donné deux graphes  $\mathcal{G}^{(l-1)}$  et  $\mathcal{H}^{(l-1)}$  structurellement *proches* à une étape  $l - 1$  quelconque de la pyramide des graphes, on s’attend à ce que  $\mathcal{G}^{(l)}, \mathcal{H}^{(l)} \in \mathbb{G}_x$  soient également structurellement *proches*. On a testé cette intuition expérimentalement sur 3 jeux de données réelles (voir Section 3.4.1 pour leurs descriptions). Dans cette expérience, pour mesurer la proximité structurelle entre graphes, on a utilisé la distance de 2-Wasserstein entre les spectres des graphes. Si les spectres sont caractéristiques de la structure d’un graphe, la taille de ces derniers diffère d’un graphe à l’autre et rend la comparaison difficile, néanmoins la distance de 2-Wasserstein permet de les comparer malgré cette difficulté :

**Definition 2 (distance de 2-Wasserstein spectrale)** Soient  $\lambda = \{\lambda_0, \lambda_2, \dots, \lambda_{n-1}\}$  et  $\mu = \{\mu_0, \mu_1, \dots, \mu_{m-1}\}$  les spectres de deux graphes. La distance de 2-Wasserstein entre ces deux quantités s’écrit :

$$d_{\mathcal{W}}^{spec}(\lambda, \mu) = \min_{\pi \in \Pi_{n,m}} \sum_{i,j} \pi_{i,j} |\lambda_i - \mu_j| \quad (3.13)$$

où  $\Pi$  est l’ensemble des mesures sur  $\lambda \times \mu$  dont les marginales sont des distributions uniformes respectivement sur  $\lambda$  et  $\mu$ . Voir le Chapitre 5 pour plus d’informations sur le transport optimal.

Notre expérience est simple, on a échantillonné des graphes  $\mathcal{G}^{(0)}$  et  $\mathcal{H}^{(0)}$  dans 3 jeux de données MUTAG, ENZYMES et DD [Kersting et al. 2016] ; on a mesuré les distances entre ces graphes et entre leurs versions *poolées*  $\mathcal{G}^{(1)}$  et  $\mathcal{H}^{(1)}$  mais aussi entre  $\mathcal{G}^{(2)}$  et  $\mathcal{H}^{(2)}$ . Les résultats des ces expériences peuvent être trouvés en Figure 3.8. Comme on peut le voir, ces figures corroborent l’intuition donnée plus haut, il y a une bonne corrélation entre les distances entre graphes et leurs graphes *poolés* ; on peut également remarquer que la corrélation entre les deux distances dépend du jeu de données.



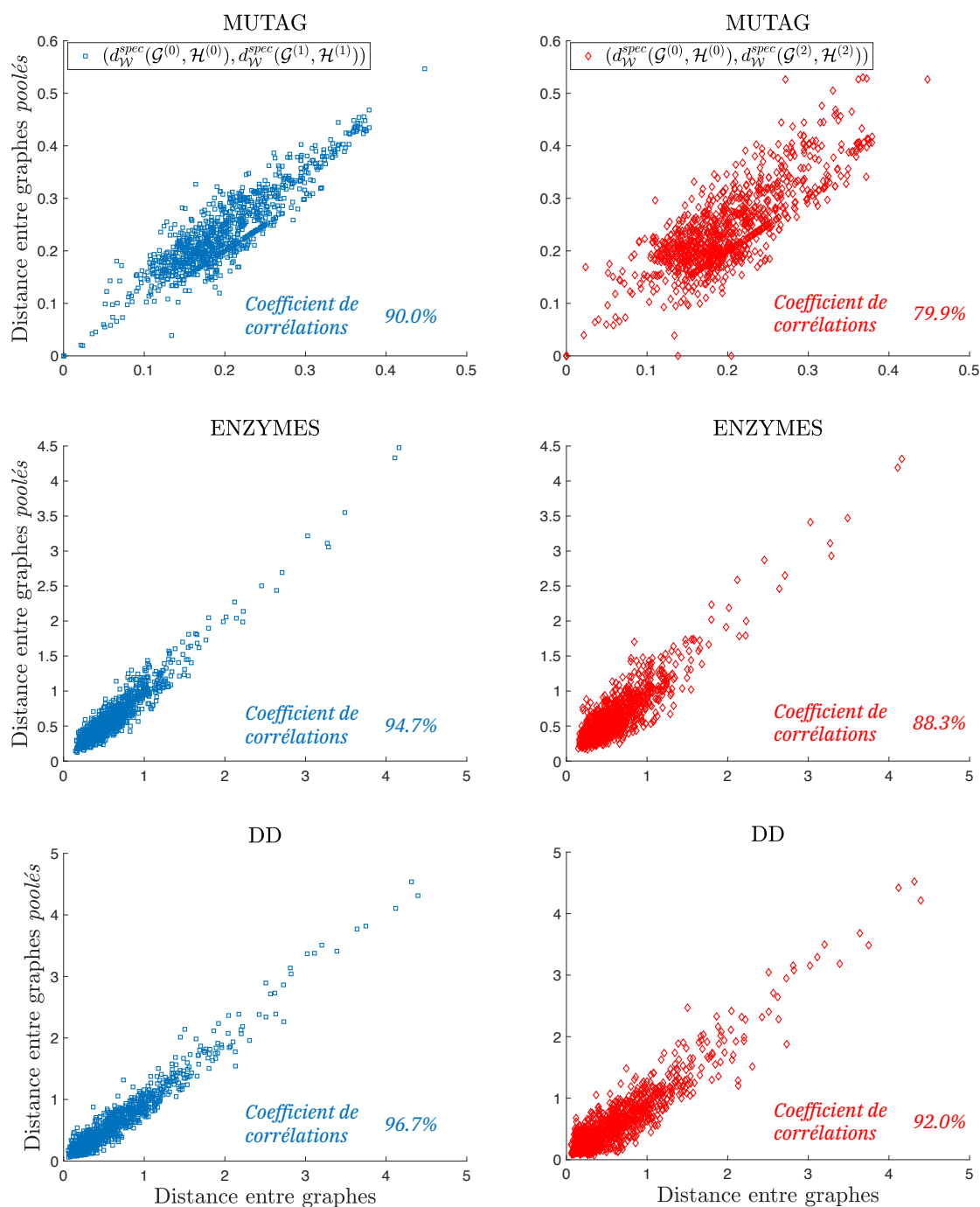


FIGURE 3.8 – *Influence du pooling sur la distance de 2-Wasserstein spectrale.* Le nuage de points bleus (resp. rouges) compare la distance de 2-Wasserstein spectrale (de la matrice d'adjacence) entre les graphes  $\mathcal{G}^{(0)}$  et  $\mathcal{H}^{(0)}$  et entre les graphes *poolés* correspondant  $\mathcal{G}^{(1)}$  et  $\mathcal{H}^{(1)}$  (resp.  $\mathcal{G}^{(2)}$  et  $\mathcal{H}^{(2)}$ ). On a également relevé les coefficients de corrélations entre les distances entre graphes et leurs versions *poolés* qu'on peut retrouver sur les figures. Comme on peut s'y attendre, ce coefficient décroît avec le nombre d'étapes de *pooling* puisque cette dernière perd de l'information, mais on peut voir que les distances restent assez corrélées montrant que les graphes proches auront tendance à avoir des versions *poolées* proches.

### 3.3.2 Construction hiérarchique du modèle

En tenant compte des constats précédents, et dans le but de tenir compte des informations présentes à chaque échelle, on construit pour chaque graphe  $\mathcal{G}^{(0)} = \mathcal{G}$  du jeu de données une pyramide hiérarchique de graphes *poolés* (voir Fig. 3.7)  $\mathcal{G}^{(l)} = (V^{(l)}, E^{(l)})$  pour  $l \in \llbracket 0, L-1 \rrbracket$ , à l'aide de la méthode de Loukas. Chaque nœud  $u_i \in V$  est progressivement regroupé selon la suite de nœuds :  $u_i, \mathcal{P}(u_i), \mathcal{P}(\mathcal{P}(u_i)), \dots, \mathcal{P}^{(L-1)}(u_i)$ . Notre algorithme consiste à apprendre des représentations de chacun de ces nœuds mais aussi une représentation pour chacun de leurs voisinages de profondeur  $p$  fixée. Ces représentations sont apprises en maximisant à chaque étape  $l$ , l'information mutuelle entre la représentation des nœuds  $\mathcal{P}^{(l)}(u_i)$  et les représentations des voisinages des nœuds de l'étape  $l-1$  dont il est issu  $\mathcal{P}^{-1}(\mathcal{P}^{(l)}(u_i))$ . Une différence majeure avec *Graph2vec* est qu'on utilise exclusivement des fonctions pour nos représentations et non pas des poids entraînaibles, ce qui constitue une première forme de contrainte.

La représentation du nœud  $\mathcal{P}^{(l)}(u_i)$  de l'étape  $l$  sera notée  $\mathbf{x}^{(l)}(u_i) = \mathbf{x}(\mathcal{P}^{(l)}(u_i))$  alors que la représentation du voisinage de profondeur  $p$  ( $\mathcal{V}_p^{(l)}(\mathcal{P}^{(l)}(u_i))$  noté  $\mathcal{V}_p^{(l)}(u_i)$ ) de ce même nœud sera notée  $\mathbf{x}_v^{(l)}(u_i) = \mathbf{x}_v(\mathcal{P}^{(l)}(u_i))$  (on n'indique pas la dépendance en  $p$  puisque c'est une grandeur fixée). On remarquera que cette notation hérite de la non injectivité de  $\mathcal{P}$  et que donc si deux nœuds différents  $u_i \neq u_j$  sont regroupés à une étape  $l$  dans le même nœud i.e  $\mathcal{P}^{(l)}(u_i) = \mathcal{P}^{(l)}(u_j)$  alors  $\mathbf{x}^{(l)}(u_i)$  et  $\mathbf{x}^{(l)}(u_j)$  (et  $\mathbf{x}_v^{(l)}(u_i)$  et  $\mathbf{x}_v^{(l)}(u_j)$ ) désignent les mêmes représentations.

La représentation du voisinage de profondeur  $p$  de  $\mathcal{P}^{(l)}(u_i) : \mathbf{x}_v^{(l)}(u_i)$  dépend des représentations des nœuds qui le composent via la fonction  $H_\theta^{(l)}$ . Et de même, la représentation du nœud  $\mathcal{P}^{(l+1)}(u_i) : \mathbf{x}^{(l+1)}(u_i)$  dépend des représentations des nœuds dont il est issu via la fonction  $F_\theta^{(l)}$ . Ce qui s'écrit formellement :

$$\mathbf{x}^{(0)}(u_i) = M_\theta(\mathbf{x}(u_i)) \quad (3.14)$$

$$\mathbf{x}_v^{(l)}(u_i) = H_\theta^{(l)}(\{\mathbf{x}^{(l)}(u_j) | \mathcal{P}^{(l)}(u_j) \in \mathcal{V}_p^{(l)}(u_i)\}) \quad (3.15)$$

$$\mathbf{x}^{(l+1)}(u_i) = F_\theta^{(l)}(\{\mathbf{x}^{(l)}(u_j) | \mathcal{P}^{(l)}(u) \in \mathcal{P}^{-1}(\mathcal{P}^{(l+1)}(u_i)) \text{ et } \mathcal{P}^{(l)}(u_j) \in \mathcal{V}_p^{(l)}(u)\}) \quad (3.16)$$

Cette procédure est illustrée en Figure 3.9. Dans la pratique,  $M_\theta$  est un perceptron multi-couche sans couche cachée et avec un biais tel que défini en Section 2.3.1.a dont la dimension de sortie est  $q_s$ . Les fonctions  $H_\theta^{(l)}$  et  $F_\theta^{(l)}$  sont des réseaux de neurones sur graphes paramétrés par  $\theta$  et dont la forme exacte sera précisée dans la section suivante. On notera  $\mathbb{X}_v^{(l)}$  l'ensemble des représentations des voisinages des nœuds de l'étape  $l$  et  $\mathbb{X}^{(l)}$  l'ensemble des représentations de ces nœuds à la même étape. On notera également  $\mathbf{X}_v^{(l)}$  la matrice dont les lignes sont les éléments de  $\mathbb{X}_v^{(l)}$  (i.e  $\mathbf{X}_v^{(l)}(i, :) = \mathbf{x}_v^{(l)}(u_i)^T$ ). Et on notera  $\mathbf{X}^{(l)}$  la matrice dont les lignes sont les éléments de  $\mathbb{X}^{(l)}$  (i.e  $\mathbf{X}^{(l)}(i, :) = \mathbf{x}^{(l)}(u_i)^T$ ).

### 3.3.3 Attributs continus et *truncated Krylov*

L'algorithme de WL utilise une fonction de *hashing* qui agit sur des caractéristiques discrètes et renvoie des caractéristiques discrètes. Cela limite donc l'applicabilité de *Graph2vec*

aux graphes portant des attributs discrets ; de plus, comme on a pu le voir la procédure d'extraction d'information de WL n'est pas la plus adaptée lorsque la tâche d'apprentissage visée requiert des informations sur des propriétés globales du graphe. Pour surmonter ce problème, nous abandonnons la fonction de *hashing* (et donc la propriété d'injectivité) au profit d'une fonction continue des voisinages, et naturellement les GNN sont de bons candidats puisqu'ils ont été spécifiquement créés pour extraire de l'information de ce type de structure. Par ailleurs, il a été prouvé que les GNN ont un pouvoir discriminatif au moins équivalent à celui de WL [Xu et al. 2019 ; Morris et al. 2019 ; Maron et al. 2019]. L'abandon de l'injectivité pour la continuité permet aux voisinages présentant des structures et des attributs proches d'avoir des représentations proches et donc *in fine* des représentations de graphes proches lorsque les graphes sont proches. Nous avons démontré la continuité des GNN, par rapport à *Gromov-Wassertsein* (en annexe A.2.2). Il s'agit d'une distance sur **les graphes attribués** (la mesure de similarité précédente ne fonctionne que sur les graphes sans attributs) s'appuyant également sur la théorie du transport optimal qui présente notamment d'excellentes propriétés (voir [Titouan et al. 2019] et le Chapitre 5).

L'autre avantage des GNN par rapport à une fonction de *hashing* est qu'il s'agit généralement de fonctions paramétrées pouvant être apprises par descente de gradient stochastique. Cela permet à la méthode de pouvoir ajuster la manière dont elle extrait les informations du voisinage, et donc d'optimiser la propagation de l'information (mutuelle) entre les couches. Cette propriété est particulièrement importante pour que le modèle soit d'une part différentiable de bout en bout et puisse être potentiellement utilisé de manière supervisée.

Parmi les nombreux GNN à disposition dans la littérature, les GCN se sont imposés comme des méthodes de référence (voir Sections 1.2 et 2.3). C'est donc tout naturellement que nous avons choisi de paramétrer notre modèle avec ces derniers. S'il existe de nombreux GCN différents, il a été démontré que la plupart de ces méthodes ont des performances très similaires, le choix du GCN n'est donc pas un élément de *design* critique [Wu et al. 2020]. Toutefois, chaque GCN présente ses propres spécificités qui peuvent le rendre plus à même de faire une tâche spécifique. Ici, nous avons porté notre choix sur le GCN nommé **truncated Krylov**, proposé par Luan et al. qui ont prouvé que leur GCN avait un comportement légèrement meilleur notamment lorsque le nombre de couches cachées augmente, avec un effet de dilution de l'information moindre par rapport aux autres GCN. Nous avons donc choisi cette méthode, compte tenu de notre objectif qui est de transporter de l'information le long de notre réseau. Le fonctionnement de Krylov peut être trouvé en Section 2.3.5. Ce choix se traduit donc sur les Equations (3.15) et (3.16) de la manière suivante (écrit vectoriellement) :

$$\mathbf{X}_v^{(l)} = \tanh([\mathbf{X}^{(l)}, \tilde{\mathbf{A}}^{(l)} \mathbf{X}^{(l)}, \tilde{\mathbf{A}}^{(l)^2} \mathbf{X}^{(l)}, \dots, \tilde{\mathbf{A}}^{(l)^p} \mathbf{X}^{(l)}] \Theta_1^{(l)}) \quad (3.17)$$

$$\mathbf{X}^{(l+1)}(i, :) = \sum_{u_j \in \mathcal{V}(u_i)} \left( \left( \tanh([\mathbf{X}^{(l)}, \tilde{\mathbf{A}}^{(l)} \mathbf{X}^{(l)}, \tilde{\mathbf{A}}^{(l)^2} \mathbf{X}^{(l)}, \dots, \tilde{\mathbf{A}}^{(l)^p} \mathbf{X}^{(l)}] \Theta_2^{(l)}) \Theta_3^{(l)} \right) (j, :) \right) + \theta_4^{(l)T} \quad (3.18)$$

Les matrices  $\Theta_1^{(l)} \in \mathbb{R}^{pq_s \times q_s}$ ,  $\Theta_2^{(l)} \in \mathbb{R}^{pq_s \times q_s}$ ,  $\Theta_3^{(l)} \in \mathbb{R}^{q_s \times q_s}$  sont des paramètres entraînaibles du modèle, avec  $\theta_4^{(l)} \in \mathbb{R}^{q_s}$  est un vecteur de biais ; et la notation [...] correspond à la conca-

tération. On remarque que la représentation du nœud  $\mathcal{P}^{(l+1)}(u) : \mathbf{x}^{(l+1)}(u)$  est la somme des caractéristiques extraites par le GCN des nœuds dont il est issu, et non une moyenne comme il est coutume. Il a été montré que la somme avait un meilleur pouvoir discriminant sur les ensembles de caractéristiques et préservait mieux l'information que la moyenne (voir [Xu et al. \[2019\]](#)).

**Remarque 12 (GNN.)**  $F_\theta^{(l)}$  et  $H_\theta^{(l)}$  ont comme entrée les mêmes grandeurs et explorent à priori le même voisinage autour de  $u$  (jusqu'à une distance  $p$ ). Toutefois,  $F_\theta$  explore également les voisins de profondeur  $p$  des voisins directs du nœud sur lesquels il est centré (il explore donc les  $p+1$  voisins). Ces voisinages sont ensuite sommés après application d'une transformation affine supplémentaire. Cela permet à  $F_\theta^{(l)}$  d'avoir un pouvoir descriptif bien supérieur à  $H_\theta^{(l)}$ . C'est attendu car  $F_\theta$  doit construire une représentation qui est censée être à la fois un descripteur du nœud issu de l'opération de pooling mais aussi le vocabulaire décrivant le contexte qui doit servir à construire les représentations des nœuds de l'étape  $l+1$ ; d'où l'intérêt d'une bien meilleure expressivité de cette fonction [[Hamilton et al. 2017a](#)]. Par ailleurs, la moindre expressivité de  $H_\theta^{(l)}$  réduit les risques de sur-apprentissage.

### 3.3.4 Negative Sampling hiérarchique

En suivant le principe de *Graph2vec*, on souhaite maximiser l'information mutuelle entre les représentations des voisinages des nœuds  $\mathcal{P}^{(l)}(u) : \mathbf{x}_v^{(l)}(u) \in \mathbb{X}_v^{(l)}$  et la représentation du nœud  $\mathcal{P}^{(l+1)}(u) : \mathbf{x}^{(l+1)}(u) \in \mathbb{X}^{(l)}$ . On aura donc un estimateur de la MI à chacune des  $L$  étapes de notre modèle soit autant de classifieurs binaires qui vont distinguer les paires positives :

$$\{(\mathbf{x}_v^{(l)}(u), \mathbf{x}^{(l+1)}(w)) \in \mathbb{X}_v^{(l)} \times \mathbb{X}^{(l+1)} | \mathcal{P}^{(l)}(u) \in \mathcal{P}^{-1}(\mathcal{P}^{(l+1)}(w))\}$$

qui sont le support de la distribution jointe (uniforme)  $P(\mathbb{X}_v^{(l)}, \mathbb{X}^{(l+1)})$  et les paires négatives :

$$\{(\mathbf{x}_v^{(l)}(u), \mathbf{x}^{(l+1)}(w)) | \mathcal{P}^{(l)}(u) \notin \mathcal{P}^{-1}(\mathcal{P}^{(l+1)}(w)) \text{ ou } \mathcal{P}^{(l)}(u) \in \mathcal{P}^{-1}(\mathcal{P}^{(l+1)}(w))\}$$

qui sont le support de la distribution produit (uniforme)  $P(\mathbb{X}_v^{(l)}) \otimes P(\mathbb{X}^{(l+1)})$ . Ce qui nous donne exactement  $L$  fonctions objectifs  $\mathcal{L}^{(l)}$  à maximiser, via la somme :

$$\begin{aligned} \mathcal{L}^{(l)} = & \mathbb{E}_{(\mathbf{x}_v^{(l)}(u), \mathbf{x}^{(l+1)}(w)) \sim P(\mathbb{X}_v^{(l)}, \mathbb{X}^{(l+1)})} \log \tau(\mathbf{x}_v^{(l)}(u) \cdot \mathbf{x}^{(l+1)}(w)) \\ & + \mathbb{E}_{(\mathbf{x}_v^{(l)}(u), \mathbf{x}^{(l+1)}(w)) \sim P(\mathbb{X}_v^{(l)}) \otimes P(\mathbb{X}^{(l+1)})} \log \tau(-\mathbf{x}_v^{(l)}(u) \cdot \mathbf{x}^{(l+1)}(w)) \end{aligned} \quad (3.19)$$

L'algorithme complet est décrit dans l'Algorithme [3.3.4.a](#). Évidemment, il y a bien plus de paires négatives que de paires positives, on utilise donc un coefficient de normalisation pour rééquilibrer la fonction objectif. Lorsque l'entraînement du modèle est achevé, on a appris des représentations pour chaque nœud de nos pyramides de graphes. La représentation pour chaque graphe de ces pyramides est calculé en sommant les représentations de chacun de leurs sommets (de dimension  $q_s$ ) et la représentation finale des graphes initiaux du jeu de données est une concaténation de leurs représentations intermédiaires, i.e :

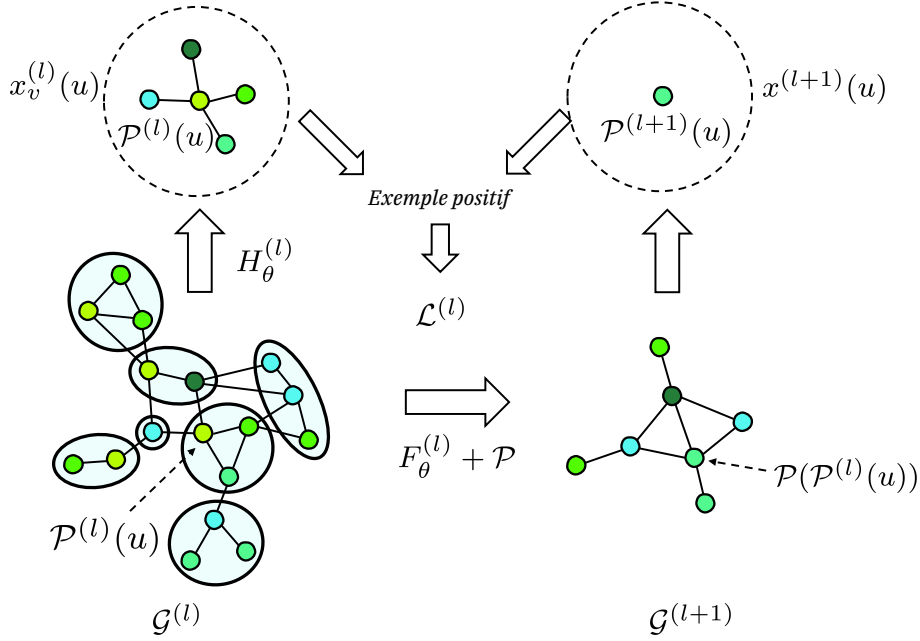


FIGURE 3.9 – *Une transition d'étape de notre algorithme.* L'information locale  $\mathbf{x}_v^{(l)}(u)$  centrée autour du nœud  $u$  est extraite du graphe  $\mathcal{G}^{(l)}$ . Le graphe est *poolé* pour former un nouveau graphe  $\mathcal{G}^{(l+1)}$ . Alors, la représentation  $\mathbf{x}^{(l+1)}(\mathcal{P}(u))$  contient les informations d'une échelle plus grande centrée autour du nœud  $\mathcal{P}(u)$ . La paire  $(\mathbf{x}_v^{(l)}(u), \mathbf{x}^{(l+1)}(v))$  est un exemple positif (resp. négatif) si  $\mathcal{P}(\mathcal{P}^{(l)}(u)) = \mathcal{P}^{(l+1)}(v)$  (resp.  $\mathcal{P}(\mathcal{P}^{(l)}(u)) \neq \mathcal{P}^{(l+1)}(v)$ ) pour la fonctionnelle de *negative sampling* de cette étape  $l$ . Cette fonctionnelle maximise l'information entre les informations locales portées par les représentants des nœuds de l'étape  $l + 1$  et les informations globales portées par les représentations de l'étape  $l$ . **Note** : Les paires positives sont aussi des paires négatives par définition du *negative sampling*.

$$\mathcal{R}^{(l)}(\mathcal{G}) = \left( \sum_{u \in V} \mathbf{x}^{(l)}(u) \right)^T \in \mathbb{R}^{1 \times q_s} \quad (3.20)$$

$$\mathcal{R}(\mathcal{G}) = \left[ \mathcal{R}^{(0)}(\mathcal{G}), \mathcal{R}^{(1)}(\mathcal{G}), \dots, \mathcal{R}^{(L-1)}(\mathcal{G}) \right]^T \in \mathbb{R}^{Lq_s} \quad (3.21)$$

Le vecteur résultant (de dimension  $d = Lq_s$ ) peut être utilisé pour faire de la visualisation (PCA ou t-SNE) ou encore directement faire de la classification de graphes.

**Remarque 13 (Taille de la pyramide.)** Dans la méthode de Loukas, le nombre de graphes dans la pyramide n'est pas prédéterminé à l'avance, seule la taille du graphe réduit peut être choisie. Lorsque le nombre de graphes produit par la méthode de Loukas n'est pas aligné avec la profondeur de notre algorithme, on a deux solutions possibles : soit la pyramide de graphes poolés générés est trop grande alors on la restreint au nombre d'étapes  $L$  choisies dans notre

modèle, soit au contraire elle est trop courte, on la complète donc en répétant le dernier graphe de cette dernière (ce qui correspond au *pooling identité*).

### 3.3.4.a Complexité

**En temps.** Les pyramides de graphes *poolés* peuvent être pré-calculées en temps linéaire en le nombre de nœuds et d’arêtes [Loukas and Vandergheynst 2018]. A priori, la principale contribution à la complexité temporelle provient de l’entraînement de la fonction objective. On appelle  $n$  le nombre maximal de nœuds dans les graphes du jeu de données,  $B$  la taille du *batch*,  $L$  le nombre d’étapes de notre algorithme et  $p$  l’ordre d’exploration de **Krylov** et  $q_s$  la dimension des représentations des sommets. La complexité exacte par *batch* est donnée par  $\mathcal{O}(BLpn^3 + BLpn^2q_s + BLpnq_s^2 + B^2Lq_s)$ . Dans l’ordre d’apparition des termes, les coûts sont dus à : l’exponentiation des matrices d’adjacences (**Krylov**), la diffusion des informations le long des arêtes (**Krylov**), la propagation avant dans les couches du réseau de neurones et le produit cartésien pour créer les paires positives et négatives. La complexité moyenne est donc  $\mathcal{O}(Lpn(n + q_s)^2 + BLq_s)$ . Les éléments qui contribuent le plus à cette complexité sont le nombre de nœuds, suivi par la dimension des caractéristiques puis la taille du *batch*. La méthode peut traiter des graphes avec des centaines de nœuds efficacement notamment sur des GPU modernes avec  $q_s = 512$  et au plus un *batch* de 8 graphes. En pratique, l’étape limitante est en fait la réduction de Loukas qui ne bénéficie pas de l’accélération via GPU.

**En espace.** L’opération la plus coûteuse en mémoire est la création des paires, notamment négatives qui dépend de la taille des graphes mais surtout de la taille du *batch*, d’où la limite à 8 introduite plus haut.

## 3.4 Expériences

Dans cette section, on pourra retrouver les différentes expériences effectuées pour évaluer notre modèle<sup>8</sup>. Les expériences sont conduites sur de nombreux jeux de données de graphes attribués. La qualité des représentations de notre méthode est d’abord évaluée sur une tâche de classification en Section 3.4.2. Ensuite, la capacité de généralisation de notre méthode est évaluée en l’entraînant sur un jeu de données puis en calculant les représentations des graphes d’un autre jeu de données similaire en Section 3.4.3. Pour conclure, nous ferons une expérience dans laquelle nous comparerons notre modèle à *Graph2vec*, et dans laquelle nous évaluerons chacun de nos choix de *design*, notamment l’influence de l’introduction du GNN par rapport à WL et de l’introduction de la méthode de *pooling* de Loukas par rapport à son absence en Section 3.4.4.

8. Toutes les expériences de l’article peuvent être trouvées ici : <https://github.com/Algue-Rythme/GAT-Skip-Gram> à l’exception de la Figure 3.8 dont le code ainsi qu’une seconde implémentation de HG2V peuvent être trouvées là : <https://github.com/Yacnmn/HG2V>.

---

**Algorithm 1** Version *haut-niveau* de l'algorithme de HG2V

---

**Objectif :** Calculer les représentations  $\mathcal{R}(\mathcal{G})$  pour  $\mathcal{G} \in \mathbb{G}_x$ .

**Entrée :** Jeu de données :  $\forall \mathcal{G} \in \mathbb{G}_x$ ; nombre d'étapes :  $L$ ; GNN :  $F_\theta^{(l)}$  et  $H_\theta^{(l)}$  avec des poids  $\theta$  initialisés aléatoirement,  $l \in \{1, \dots, L\}$ ; nombre maximal d'*epoch* :  $E$ .

**Pour** chaque *epoch*  $e \in \{1, \dots, E\}$  :

Créer une partition  $\cup_k^B B_k = \mathbb{G}_x$  avec  $B_k \cap B_{k'} = \emptyset$ .

**Pour** chaque  $B_k$  :

**Pour** chaque  $\mathcal{G} \in B_k$  :

Créer la pyramide de graphe  $\mathcal{G}^{(l)} \in B_k^{(l)}$  *poolé* avec l'algorithme de Loukas pour  $l \in \{0, \dots, L-1\}$  (voir Equation (3.15) et (2.50) en Section 3.3.1.b).

**Pour** chaque étape  $l \in \{0, \dots, L-1\}$  :

**Pour** chaque graphe  $\mathcal{G}^{(l)} \in B_k^{(l)}$  :

**Pour** chaque nœud  $\mathcal{P}^{(l)}(u) \in V^{(l)}$  :

On génère les représentations locales  $\mathbf{x}_v^{(l)}(u) = H_\theta^{(l)}(\cdot)$  (voir Equation (3.16) et (3.18) en Section 3.3.2);

**Pour** chaque nœud  $\mathcal{P}^{(l+1)}(v) \in V^{(l+1)}$  tel que  $\mathcal{P}^{(l)}(u) \in \mathcal{P}^{-1}(\mathcal{P}^{(l+1)}(v))$  :

On génère la représentation du nœud  $\mathcal{P}^{(l+1)}(v)$  :  $\mathbf{x}^{(l+1)}(v) = F_\theta^{(l)}(\cdot)$  (voir Section 3.3.2);

On crée la paire positive  $(\mathbf{x}^{(l)}(u), \mathbf{x}_v^{(l+1)}(v))$ .

**Pour** chaque paire de graphes  $(\mathcal{G}, \mathcal{G}') \in \mathbb{G}_x^2$  :

**Pour** chaque paire de nœuds  $(\mathcal{P}^{(l)}(u), \mathcal{P}^{(l+1)}(v)) \in V^{(l)} \times V^{(l+1)}$  :

On crée la paire d'exemples négatifs  $(\mathbf{x}_v^{(l)}(u), \mathbf{x}^{(l+1)}(v))$

Avec ces exemples positifs et négatifs on construit la fonctionnelle  $\mathcal{L}^{(l)}$  selon les modalités de l'Equation (3.19).

On effectue une itération de l'algorithme de descente de gradient (*Adam*) sur  $\sum_{l=0}^{L-1} \mathcal{L}^{(l)}$ .

---

### 3.4.1 Jeux de données

Pour nos expériences, nous avons utilisé un large panel de jeux de données de la littérature [Kersting et al. 2016] :

- PROTEINS, ENZYMES, D&D, NCI1, NCI109, MUTAG et PTC\_FR qui sont des jeux de données issus de la biologie et de la chimie qui représentent en général des molécules.
- IMDB-Binary est un jeu de données composé de graphes dont les nœuds représentent des acteurs, ces acteurs sont liés si et seulement s'ils ont fait une apparition dans le même film. Ces graphes sont construits à partir de films d'action ou de romance. La tâche de classification consiste à déterminer à partir de quel genre un de ces graphes d'acteurs a été construit.
- REDDIT-Binary, ces graphes correspondent à des discussions en ligne sur le site communautaire *Reddit*. Les nœuds de chacun de ces graphes représentent un utilisateur de la plateforme, deux utilisateurs sont liés si l'un d'eux répond aux commentaires de l'autre. Ces graphes sont construits à partir des conversations de 4 forums (*appelés sub-reddit*) ; *IamA*, *AskReddit* qui sont des forums de questions/réponses et *TrollXChromosomes*, *atheism* qui sont des forums de discussions. Les graphes sont étiquetés en fonction de leur appartenance à un forum de discussion ou de questions/réponses. *Reddit-5K* est un jeu de données similaires, mais comportant 5 classes.

Nous avons également travaillé avec des jeux de données moins connus :

- Frankenstein, créé par Orsini et al. [2015] à partir de BURSI qui est un jeu de données moléculaire, dont les étiquettes ont été remplacées par des images de MNIST dont les chiffres correspondent aux étiquettes originales.
- *Diffusion Limited Aggregation* – DLA – est un nouveau jeu de données que nous introduisons ici, et que nous avons construit spécifiquement pour illustrer certaines propriétés de notre algorithme. Dans ce jeu de données, les graphes sont construits via un processus aléatoire d'agrégation mettant en jeu des particules. Le processus est tel que les graphes formés sont invariants d'échelle, i.e la distribution des degrés suit une loi puissance. L'intérêt de notre modèle vis-à-vis du modèle de Barabasi-Albert (qui produit ce type de graphe) est qu'on obtient des graphes avec des attributs naturels (les positions dans l'espace des particules) alors que le modèle de Barabási-Albert en est dépourvu. Ce jeu de données permet donc d'évaluer l'habileté de notre modèle à distinguer des propriétés globales de graphes attribués. Les détails de sa construction peuvent être retrouvés en annexe A.2.3.
- MNIST & USPS en tant que graphes : on a converti les éléments de MNIST et USPS des jeux de données de chiffres manuscrits. Chaque pixel de ces images devient un nœud, dont les caractéristiques sont la luminosité du pixel d'origine et la position de ce dernier. Les nœuds provenant de pixels pas suffisamment (voir Annexe A.2.4) lumineux sont supprimés. Les nœuds qui proviennent de pixels qui étaient adjacents sont reliés. Pour MNIST, étant donné la taille du jeu de données, et des graphes considérés, nous nous sommes restreints à 10 000 éléments dans le jeu de données. Voir l'annexe A.2.4 pour la visualisation de ce jeu de données et des détails supplémentaires sur sa construction.
- FASHIONMNIST est un jeu de données similaire à MNIST. Mais, ses images représentent des accessoires de mode. Ce jeu de données est composé de 60 000 images, représentant



Jeu de données	# graphe	# classe	# nœud moy.	# arête moy.
<b>PROTEIN</b>	1113	2	39.06	72.82
<b>ENZYMES</b>	600	6	32.63	64.14
<b>DD</b>	1178	2	284.32	715.66
<b>NCI1</b>	4110	2	29.87	32.30
<b>NCI109</b>	4127	2	29.68	32.13
<b>MUTAG</b>	188	2	17.93	19.79
<b>PTC_FR</b>	351	2	14.56	15
<b>IMDB BINARY</b>	1000	2	19.77	96.53
<b>IMDB MULTI</b>	1500	3	13.00	65.94
<b>REDDIT BINARY</b>	2000	2	429.63	497.75
<b>REDDIT 5K</b>	4999	5	508.82	594.87
<b>FRANKENSTEIN</b>	4337	2	16.90	17.88
<b>DLA</b>	1000	2	500	499
<b>MNIST</b>	10000	10	144	237.76
<b>USPS</b>	9298	10	89.04	141.35
<b>FASHION-MNIST</b>	10000	10	366.13	672.36

TABLE 3.2 – *Quelques caractéristiques des jeux de données considérés.* # : nombre. moy. : moyen.

10 types d’accessoires différents d’où les 10 étiquettes. La même procédure qu’à MNIST et USPS est appliquée pour transformer ces images en graphes. Seules 10 000 d’entre elles sont conservées. Voir également l’annexe A.2.4 pour la visualisation de ce jeu de données et des détails supplémentaires sur sa construction.

Tous les attributs continus ont été centrés normés. Les étiquettes discrètes sont encodées en *one-hot*. Lorsqu’il n’y pas de caractéristiques sur les nœuds, on utilise les degrés comme tels.

### 3.4.2 Classification supervisée

Dans cette première tâche, on entraîne le modèle sur l’ensemble des graphes disponibles dans le jeu de données, puis on détermine les représentations de ces graphes. La qualité de ces représentations est évaluée via une tâche de classification supervisée. Le classifieur utilisé est une machine à vecteur de support (SVM) à noyau gaussien (RBF), provenant de la librairie de scikit-learn et qui est appliqué sur les représentations obtenues pour chacune des méthodes comparées.

**Paramètres & Modalité d’évaluations.** Le modèle a été entraîné pendant 10 *epochs*. Durant ces *epochs*, à chaque itération de l’algorithme d’optimisation, 8 graphes (la taille du *batch*) sont choisis aléatoirement dans le jeu de données et tout le vocabulaire produit par ces graphes est utilisé pour créer des paires négatives et positives. L’optimiseur choisi est Adam [Kingma and Ba 2014], qui est censé mettre à jour les poids automatiquement. Toutefois, on a constaté

expérimentalement que forcer la décroissance du *learning rate* de sorte à le diviser par 1000 (avec une loi exponentielle) à la fin des 10 epochs, permet d’obtenir de meilleurs résultats. Les hyperparamètres pertinents dans ce modèle sont la dimension de la représentation finale qui est indirectement choisie à travers la dimension de la représentation des nœuds. On a soumis la sélection de cette grandeur à une grille de recherche dont les valeurs potentielles sont  $d_n \in \{16, 128, 256\}$ . D’autres hyperparamètres sont sélectionnés via cette grille tels que la distance d’exploration de **Krylov**  $p \in \{2, 4\}$  et la profondeur du réseau  $L \in \{3, 5\}$ . Les paramètres du modèle ainsi que ceux du classifieur sont choisis en effectuant une validation croisée à 5 blocs sur 80% de notre jeu de données en utilisant l’*accuracy* de la SVM comme métrique. Une fois la meilleure combinaison de paramètres déterminée, le classifieur est entraîné sur l’ensemble des 80% des représentations qui ont servis durant la validation et est ensuite évalué sur les 20% de données restantes. **Note** : HG2V est entraîné sur 100% du jeu de données à chaque fois. On peut voir en annexe A.2.5 le résultat de cette expérience en ne s’entraînant que sur 20% du jeu de données.

**Autres méthodes.** Les résultats de nos expériences sont comparés à un large panel de méthodes de la littérature en Table 3.3; d’abord, les méthodes à noyau évoquées en début de chapitre, dont les résultats sont issus des articles de Kriege et al. [2016] et de Nikolentzos et al. [2019]. Ces articles rapportent les résultats obtenus avec WL-Optimal Assignment [Kriege et al. 2016] et Wasserstein Weisfeiler-Lehman [Togninalli et al. 2019]. Comme on peut le voir, ces méthodes présentent presque toujours de meilleures performances que les méthodes inductives basées sur un réseau de neurones. Toutefois, ces méthodes ont un coût prohibitif lorsqu’il s’agit de traiter des graphes de grandes tailles. Les résultats de *Diffpool* Ying et al. [2018] et *GIN* [Xu et al. 2019] sont extraits de l’article de Errica et al. [2020]. Ces algorithmes sont supervisés. Diffpool s’appuie également sur du *pooling*, mais ce dernier est entraînable. Les résultats de *MinCutPool* et des autres méthodes supervisées sont également disponibles et extraits de l’article de Bianchi et al. [2020b]. Dans cet article, seuls les degrés des nœuds sont utilisés, ainsi que les coefficients de *clustering* comme attributs de nœuds. Enfin, les résultats d’*Infograph* extraits de l’article de Sun et al. [2020] sont également consultables sur cette table. C’est la méthode la plus proche de notre méthode en termes de propriétés, elle est non supervisée, différentiable de bout en bout et s’appuie également sur le principe de la maximisation de la MI. Toutefois, elle n’utilise pas de *pooling* de graphe. Cette méthode est une référence dans le calcul de représentation non supervisée de graphe attribué.

**Evaluation** Dans les résultats observés, on constate une amélioration substantielle des résultats par rapport à *Graph2vec* et ce pour de nombreux jeux de données : plus spécifiquement, lorsque les graphes sont de grandes tailles et leurs caractéristiques. Pour les jeux de données plus complexes comme REDDIT-B et REDDIT-5K, avec des graphes relativement grands, on obtient de très bons résultats, proche de l’état de l’art (même comparativement aux méthodes supervisées). Toutefois sur les jeux de données avec des graphes de taille modeste, les résultats sont moins significatifs, bien que positifs.

Frankenstein est un autre jeu de données complexe qui illustre la capacité de notre modèle à traiter des jeux de données plus difficiles. Les résultats sont bons comparativement aux autres

Jeu de données	HG2V	Graph2Vec	Infograph	DiffPool (supervisé)	GIN (supervisé)	MinCutPool (supervisé)	WL-OA (noyau)	WWL (noyau)
IMDB-m	47.9 ± 1.0	<b>50.4 ± 0.9</b>	49.6 ± 0.5	45.6 ± 3.4	48.5 ± 3.3	✗	✗	✗
PTC_FR	<b>67.5 ± 0.5</b>	60.2 ± 6.9	✗	✗	✗	✗	<b>63.6 ± 1.5</b>	✗
FRANK.	<b>65.3 ± 0.7</b>	60.4 ± 1.3	✗	✗	✗	✗	✗	✗
MUTAG	81.8 ± 1.8	83.1 ± 9.2	<b>89.0 ± 1.1</b>	✗	✗	✗	84.5 ± 1.7	<b>87.3 ± 1.5</b>
IMDB-b	71.3 ± 0.8	63.1 ± 0.1	<b>73.0 ± 0.9</b>	68.4 ± 3.3	71.2 ± 3.9	✗	✗	<b>74.4 ± 0.8</b>
NCI1	76.3 ± 0.8	73.2 ± 1.8	✗	76.9 ± 1.9	<b>80.0 ± 1.4</b>	✗	<b>86.1 ± 0.2</b>	85.8 ± 0.2
NCI109	<b>75.6 ± 0.7</b>	74.3 ± 1.5	✗	✗	✗	✗	<b>86.3 ± 0.2</b>	✗
ENZYMES	<b>66.0 ± 2.5</b>	51.8 ± 1.8	✗	59.5 ± 5.6	59.6 ± 4.5	✗	59.9 ± 1.1	<b>73.3 ± 0.9</b>
PROTEINS	75.7 ± 0.7	73.3 ± 2.0	✗	73.7 ± 3.5	73.3 ± 4.0	<b>76.5 ± 2.6</b>	76.4 ± 0.4	<b>77.9 ± 0.8</b>
MNIST	<b>96.1 ± 0.2</b>	56.3 ± 0.7	✗	✗	✗	✗	✗	✗
D&D	79.2 ± 0.8	58.6 ± 0.1	✗	75.0 ± 3.5	75.3 ± 2.9	<b>80.8 ± 2.3</b>	79.2 ± 0.4	<b>79.7 ± 0.5</b>
REDDIT-b	91.2 ± 0.6	75.7 ± 1.0	82.5 ± 1.4	87.8 ± 2.5	89.9 ± 1.9	<b>91.4 ± 1.5</b>	89.3	✗
DLA	<b>99.9 ± 0.1</b>	77.2 ± 2.5	✗	✗	✗	✗	✗	✗
REDDIT-5K	55.5 ± 0.7	47.9 ± 0.3	53.5 ± 1.0	53.8 ± 1.4	<b>56.1 ± 1.7</b>	✗	✗	✗

TABLE 3.3 – *Expériences de transfert.* L’*accuracy* dans la tâche de classification. HG2V est entraîné sur l’ensemble d’entraînement et de test, sans en utiliser les étiquettes car la méthode est non supervisée. La sélection des hyperparamètres de la SVM-RBF est effectuée via une validation croisée à 5 blocs. Le tableau présente les résultats de classification pour l’ensemble de test, pour les meilleurs paramètres issus de cette validation croisée (moyennés) sur 10 exécutions de la procédure d’entraînement et de validation des hyperparamètres. Les écarts types sont également disponibles. Le signe ✗ indique que les résultats ne sont pas disponibles pour cette méthode.

méthodes inductives. Comme les caractéristiques des nœuds des graphes de Frankenstein sont des images, on a refait une expérience dans laquelle on a intercalé un CNN entre les caractéristiques des graphes et notre modèle. Cette opération nous a permis d’améliorer les résultats pour atteindre une performance en *accuracy* de  $66,5 \pm 0,4$ . Si l’amélioration est faible, elle permet de témoigner de la pertinence de la différentiabilité de bout en bout de notre modèle, qui permet de construire des modèles plus complexes autour de HG2V.

**Remarque 14 (Temps de calcul.)** *En s’entraînant seulement sur 1 seule epoch, on a en réalité déjà des résultats de classification supervisée solides. L’entraînement sur une seule epoch, notamment sur les jeux de données constitués de molécules, prend moins d’une minute sur une GTX 1080 GPU. Pour les jeux de données les plus éprouvants comme reddit-multi-5k, le simple pré-calcul des graphes réduit avec la méthode de Loukas demande près de 40 min (ce temps pourrait être significativement amélioré en parallélisant cet algorithme). Après cette étape, par contre le temps de calcul n’excède pas 190s par epoch, ce qui fait un temps total d’entraînement de 70 minutes.*

### 3.4.3 Inductivité

HG2V est une méthode inductive et permet d’entraîner HG2V sur une portion d’un jeu de données et ensuite d’inférer les représentations de l’entièreté de ce jeu de données. Nous avons conduit les mêmes expériences décrites plus haut, en nous entraînant sur seulement 20% de

chacun des jeux de données. Les résultats de cette expérience sont satisfaisants et peuvent être retrouvés en Annexe [A.2.5](#).

**Expériences.** Toutefois, nous avons souhaité aller plus loin pour évaluer la capacité de généralisation de notre modèle. Nous avons notamment évalué la capacité de notre modèle à pouvoir transférer les poids appris sur un jeu de données sur un autre jeu de données. Les techniques de transferts d'apprentissage sont très en vogue actuellement [[Courty et al. 2016](#); [Murez et al. 2018](#)]. Les réseaux de neurones sont très gourmands en temps et en capacité de calcul, pouvoir réutiliser certains poids est donc très avantageux. Dans notre cas, les poids n'ayant pour but que de transférer de l'information d'une couche à l'autre, on peut s'attendre à ce que pour deux jeux de données partageant des caractéristiques communes, les poids optimisés pour l'un devraient permettre de calculer une bonne représentation pour l'autre. C'est un avantage de l'aspect non supervisé de la méthode, on peut s'attendre à avoir de bons résultats sur ce type de tâche sans avoir recours à des techniques notamment issues du transport optimal, par exemple, pour aligner les représentations. L'expérience menée ici a pour but de tester cette intuition.

Les meilleurs hyperparamètres trouvés dans la section précédente sont gardés tel qu'ils sont sans recherche supplémentaire. Le but n'étant pas d'atteindre le meilleur résultat possible, mais de démontrer la réusabilité des poids de la méthode.

**Évaluation.** Comme on peut le voir [Table 3.4](#), il y a une perte d'*accuracy* à chaque transfert, c'est attendu puisque le modèle n'a pas été entraîné spécifiquement pour le jeu de données sur lequel il est testé. Toutefois, on peut voir que pour de nombreux transferts, la perte d'information est minime allant de 2% à 7%. Notre méthode est tout à fait crédible dans ce type de tâche et utilisable dans un contexte réel. Même pour des graphes dont les caractéristiques sont de même nature mais qui conceptuellement représentent des choses différentes (Mnist vs FASHIONMNIST), le transfert est relativement fonctionnel. Les représentations obtenues ne sont pas beaucoup moins performantes que celles obtenues par apprentissage directement sur le jeu de données.

On conclut donc que notre algorithme a de bonnes propriétés de généralisation sans avoir la nécessité de faire du paramétrage d'hyperparamètres fin ou à procéder à une procédure d'adaptation coûteuse.

**Remarque 15 (Asymétrie.)** *On peut remarquer que les résultats de transfert ne sont pas symétriques, par exemple MNIST et USPS qui sont deux jeux de données de chiffres écrits. Le transfert de Mnist à USPS est plus performant que l'inverse. Ce comportement a déjà été observé dans d'autres articles de transfer learning (sur les images), notamment sur ces deux jeux de données [[Tzeng et al. 2017](#)].*

**Remarque 16 (Comparaison avec la littérature.)** *Cette expérience n'est pas possible avec Graph2vec (transductive) et les modèles entraînés avec des fonctions objectifs supervisées tel que GIN et Diffpool. Au moment de l'écriture de ce manuscrit, nous n'avions pas trouvé d'article dans lequel de telles expériences avaient été faites, cette expérience n'a donc pas de point de comparaison.*

Ensemble d'entraînement	Ensemble inféré	Accuracy (inféré)	Différentielle avec les résultats de la table 3.3
MNIST	USPS	94.86	<b>X</b>
USPS	MNIST	93.68	-2.40
REDDIT-b	REDDIT-5K	55.00	-0.48
REDDIT-5K	REDDIT-b	91.00	-0.15
REDDIT-b	IMDB-b	69.00	-2.25
REDDIT-5K	IMDB-b	69.50	-1.75
MNIST	FASHION MNIST	83.35	<b>X</b>

TABLE 3.4 – *Table de résultat de l'étude ablative.* Accuracy de la classification obtenues en s'entraînant sur un jeu de données et en utilisant les poids obtenus sur un autre jeu de données. Les hyper-paramètres sélectionnés sont identiques à ceux de la Table 3.3.

### 3.4.4 Étude ablative

**Expérience.** Pour confirmer expérimentalement les choix de *design* de notre méthode, on a réalisé des études ablatives en retirant séparément la réduction de Loukas et le GNN de notre modèle. Lorsqu'on retire ces deux spécificités de notre modèle, on retombe sur *Graph2vec*. Dans ces expériences, les dimensions des représentations sont choisies égales à 1024.

- **Graph2vec + GCN** Dans ce modèle, on retire la réduction de Loukas. La seule différence avec *Graph2vec* est le remplacement de Weisfeiler-lehman par un GNN. Tous les attributs connus pour les graphes sont utilisés (en les concaténant).
- **Graph2vec + Loukas** Dans ce modèle, on retire le GCN de HG2V, l'algorithme correspond donc à *Graph2vec* appliqué à une séquence de graphe réduit. L'étiquette du nœud  $\mathcal{P}^{(l)}(u)$  est générée en concaténant et en hashant les étiquettes de  $\mathcal{P}^{-1}(\mathcal{P}^{(l)}(u))$  (comme une itération de WL le ferait). La séquence des graphes ainsi générée est utilisée dans *Graph2vec* puis la représentation finale d'un graphe correspond à la concaténation des représentations de ses graphes réduits. Les attributs continus sont ignorés car WL ne peut pas les traiter.

**Évaluation.** Les résultats de ces expériences se trouvent en Table 3.5. Dans les jeux de données avec des petits graphes (moins de 30 nœuds en moyenne), l'usage de la réduction n'est pas indispensable. Mais dès que les graphes atteignent des tailles significatives, l'apport de l'utilisation de la réduction est flagrant. Cela confirme le propos introductif de HG2V, le *pooling* aide le processus d'extraction à récupérer de l'information à grande échelle ; ce qui bien évidemment est d'autant plus important sur les grands graphes. Quant à l'utilisation du GNN, son habilité à tenir compte des attributs continus permet une amélioration significative des résultats dans le jeu de données. La réduction de Loukas, et le GNN pris séparément augmentent bien les performances de *Graph2vec* mais présentent des résultats moindres par rapport à HG2V, preuve de la pertinence de ces deux outils améliorant d'une part *Graph2vec* et permettant d'autre part de proposer une méthode compétitive avec l'état de l'art.

Jeu de données	HG2V		Graph2vec+GNN		Graph2vec+Loukas		Graph2vec
	Accuracy	Diff.	Accuracy	Diff.	Accuracy	Diff.	
IMDB-b	70.85	+7.75	70.70	+7.60	57.5	-5.60	63.10
NCI1	77.97	+4.75	75.40	+2.18	65.45	-7.77	73.22
MNIST	95.83	+39.56	91.05	+34.78	72.5	+16.23	56.27
D&D	78.01	+19.37	79.26	+13.16	66.10	+7.45	58.64
REDDIT-B	91.95	+16.23	OOM	✗	82.50	+6.78	75.72

TABLE 3.5 – *Étude ablative*. L’accuracy de l’ensemble de tests est dans la colonne correspondante et la colonne *Diff.* correspond à la différence de performance avec *Graph2vec*. L’indication **OOM** est relative à un déficit de mémoire empêchant la méthode d’être exécutée.

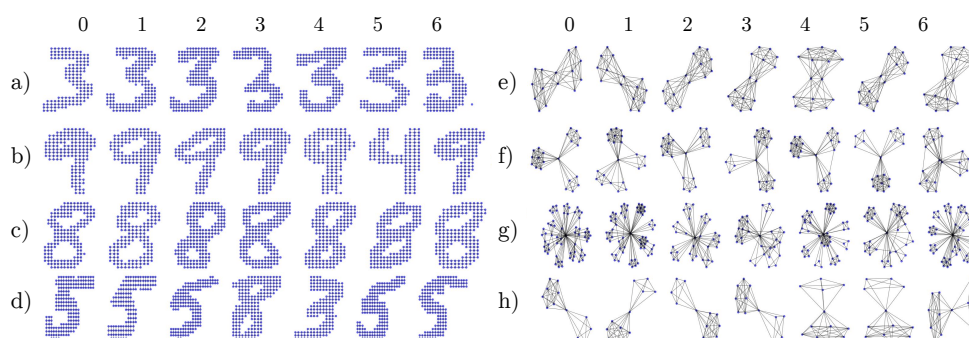


FIGURE 3.10 – *Six plus proches voisins dans l’espace latent pour MNIST et IMDB-b.* La colonne 0 correspond aux graphes choisis aléatoirement et les 6 plus proches voisins (dans l’ordre) correspondent aux colonnes 1 à 6.

### 3.4.5 Espace latent

Nous avons fait quelques expériences pour visualiser le voisinage d’une représentation dans l’espace latent. Ces expériences peuvent être observées en Figure 3.10. On peut y voir certains graphes choisis aléatoirement (sur la colonne la plus à gauche) ainsi que leurs 6 plus proches voisins dans l’espace latent (de la colonne 1 à 6). Ces graphes sont issus de MNIST et d’IMDB-b. On peut aisément constater que les graphes proches dans l’espace latent ont une structure proche. D’autres visualisations de ce type peuvent être trouvées en Annexe A.2.6.

## 3.5 Conclusion

Nous avons proposé une nouvelle méthode pour l’apprentissage de représentation de graphes. Cette méthode est totalement non-supervisée, et peut apprendre à extraire des caractéristiques via l’information mutuelle. L’utilisation d’une méthode de réduction comme celle de Loukas permet de tenir compte de plusieurs échelles en même temps, ceci grâce à sa capacité de préserver le spectre du graphe et donc sa forme. Cette méthode est inductive et peut être entraînée

---

sur seulement une fraction d'un jeu de données. La méthode produit des représentations de qualité qui nous permettent d'atteindre des résultats compétitifs dans la tâche de classification notamment face aux méthodes supervisées. Dans le prochain chapitre, nous nous intéresserons à la représentation de nœuds, et plus particulièrement de nœuds portant plusieurs caractéristiques.







## Chapitre 4

# Analyse des corrélations canoniques appliquée aux graphes

*Dans ce chapitre nous présentons, un modèle pour l'analyse de données multivues, c'est-à-dire d'objets représentés par plusieurs caractéristiques appelées vues. Nous nous intéressons plus particulièrement au cas où ces objets sont structurés par un graphe. Classiquement pour traiter ces données, on réduit les différentes caractéristiques de ces objets en une seule qu'on peut ensuite traiter par des méthodes usuelles d'apprentissage. Une méthode classique de la littérature pour effectuer cette réduction consiste à rechercher des projections de ces caractéristiques dans un espace de petite dimension (par rapport aux dimensions de celles-ci) de sorte à maximiser les corrélations croisées entre ces projections. Une de ces projections est alors utilisée comme caractéristique unique. Cette manière de procéder est appelée analyse des corrélations canoniques (canonical correlation analysis - CCA), et donne son nom à toute une classe de méthodes qui se basent sur le même principe. Notre travail s'inscrit dans la continuité de ces méthodes et s'appuiera notamment sur une interprétation probabiliste de la CCA. Les deux travaux suivants ont été publiés sur ce sujet : [Kaloga et al. 2021a,b] et servent de fondation à ce chapitre.*

### 4.1 Introduction

Les jeux de données multivues, i.e les ensembles de données pour lesquelles un même objet est décrit par plusieurs caractéristiques, sont des sujets d'intérêt pour la communauté de l'apprentissage automatique. Les acteurs publics et privés de la société sont friands d'informations, les moyens d'en acquérir se multiplient notamment via une recrudescence de capteurs, détecteurs et traqueurs divers. Il n'est donc pas rare d'avoir des informations provenant de plusieurs canaux mesurant chacun une caractéristique différente d'un même objet. On retrouve ces données dans de nombreux domaines tels que la bio-informatique [Yamanishi et al. 2003], l'acoustique [Arora and Livescu 2013], la surveillance [Collins et al. 2001] ou encore les réseaux sociaux [Benton et al. 2016] pour n'en citer que quelques-uns. Dans l'optique de traiter ces données dans un contexte *réel*, un modèle d'apprentissage doit vérifier un certain nombre de propriétés. Il doit tout d'abord être *scalable*, c'est-à-dire pouvoir s'appliquer à un grand volume de données compte tenu notamment de la taille des jeux de données actuels. Ce modèle doit également être *robuste*,

à savoir s'accommoder de l'existence d'objets avec des vues manquantes car les jeux de données sont sujets aux erreurs d'acquisitions, des vues peuvent donc être manquantes pour certains objets. Pour traiter ces cas problématiques, on pourrait simplement supprimer les objets concernés mais dans un jeu de données multivues une telle opération nous priverait de précieuses informations apportées par les autres vues disponibles. La capacité d'un modèle multivues à pouvoir traiter ces données incomplètes est donc cruciale pour une utilisation en condition réelle. Par ailleurs, certains jeux de données multivues sont localement structurés, on peut les voir comme des graphes dont les nœuds sont décrits par plusieurs caractéristiques. Cette structure apporte une information supplémentaire sur les objets du jeu de données, qui peut être exploitée pour faciliter certaines tâches d'apprentissage ; c'est pourquoi les modèles multivues capables d'intégrer ces informations lorsqu'elles existent ont un intérêt certain. S'il existe actuellement de nombreuses méthodes permettant l'analyse de données multivues [Li et al. 2019a], des méthodes capables de traiter à la fois un large volume de données, d'être robustes à l'existence d'éléments incomplets et capables d'exploiter des informations apportées par une structure locale de graphe n'existent pas. L'objectif de ce travail est de répondre à cette problématique.

**Contexte.** Pour réaliser des tâches d'apprentissage sur les données multivues, on peut s'appuyer sur l'analyse des corrélations canoniques (*Canonical Correlation Analysis* - CCA) [Hotelling 1936 ; Kettenring 1971] qui consiste à rechercher une représentation en petite dimension de chacune des vues des objets étudiés. Le premier modèle de CCA, appelé<sup>1</sup> CCA linéaire [Hotelling 1936] se limite à des jeux de données comportant 2 vues. Dans ce modèle, les représentations sont calculées par les projecteurs linéaires des 2 vues qui maximisent conjointement les corrélations croisées entre les projections. L'une ou l'autre des représentations ainsi obtenues contient alors des informations provenant des deux caractéristiques et peut donc être utilisée pour faciliter certaines tâches d'apprentissage. La CCA linéaire a servi d'inspiration à de nombreux modèles d'analyses multivues, dont on peut distinguer deux familles ; les méthodes *algébriques* et celles *probabilistes* :

- D'une part, les approches *algébriques* de la CCA sont essentiellement des variations de la CCA linéaire. Ces approches sont versatiles et possèdent de bonnes propriétés qui les ont rendu populaires ; elles sont en général non paramétriques en plus de posséder pour la plupart des solutions quasi explicites. Leur utilisation est donc facilitée et ne nécessite pas d'ajustement à effectuer par rapport au jeu de données utilisé. L'intérêt pour ces méthodes a permis l'extension de la CCA linéaire, d'abord à des jeux de données possédant plus de deux vues [Kettenring 1971] puis à ceux comportant de fortes non-linéarités (voir Kernel CCA [Akaho 2001 ; Ling and Fyfe 2000], Deep CCA [Andrew et al. 2013 ; Benton et al. 2019] ou encore Autoencoder CCA [Wang et al. 2015]). Malgré tout, ces méthodes restent limitées, elles souffrent notamment d'un problème de scalabilité [Chang et al. 2018 ; Lopez-Paz et al. 2014] ; car pour résoudre ces méthodes, on est amené à effectuer une décomposition en éléments propres sur une matrice de la taille du carré du nombre de données, la construction de cette matrice est souvent coûteuse et de l'ordre de  $O(n^2)$ , où  $n$  est le nombre d'éléments (objets) du jeu de données (voir Table. 4.3.2).

---

1. En réalité, ce modèle s'appelle CCA, mais le terme CCA désigne aujourd'hui également la classe des méthodes inspirées de ce problème, pour éviter la confusion on appellera donc le modèle originel CCA linéaire.

- D'autre part, une approche *probabiliste* à la CCA a été développée, elle s'appuie essentiellement sur une équivalence<sup>2</sup> entre la CCA linéaire et un problème d'inférence Bayésien [Bach and Jordan 2005]. Cette approche a gagné en popularité ces dernières années notamment du fait de l'introduction des auto-encodeurs variationnels [Kipf and Welling 2016a] (*Variational autoencoders* - VAE) qui permettent à la fois de résoudre des problèmes d'inférences bayésiennes sur de grands volumes de données et la réalisation de tâches d'inférence inaccessible aux méthodes algébriques. Cette approche résout donc le problème de scalabilité des extensions algébriques de la CCA [Wang et al. 2016 ; Karami and Schuurmans 2020], mais en contrepartie on perd en versatilité et facilité d'utilisation : les modèles probabilistes sont des modèles plus difficiles à ajuster du fait de leurs nombreux hyperparamètres.

En parallèle de ces avancés sur la CCA, il a été démontré que lorsqu'une information de structure entre les objets (qui sont à l'origine des vues) existe et qu'elle est intégrée dans l'apprentissage des représentations de ces objets (à partir des vues), elle permet d'améliorer significativement les performances de nombreuses tâches d'apprentissage notamment de *clustering* [Chen et al. 2018, 2019]. Actuellement seules certaines méthodes algébriques [Chen et al. 2018, 2019] permettent de tenir compte d'une telle information, mais elles souffrent des mêmes problèmes de scalabilité que certaines méthodes algébriques.

En résumé, dans ce chapitre nous présenterons un modèle d'analyse de données multivues en nous appuyant sur une formulation variationnelle de la CCA linéaire et notamment sur les VAE. Grâce à cette formulation, le modèle sera scalable, et en mesure d'intégrer des informations de nature structurelle dans le calcul des représentations des données multivues. Plus encore, la méthode que nous développerons ici, a des propriétés uniques parmi toutes les autres méthodes CCA, elle est notamment en capacité de traiter des données multivues incomplètes. Sa nature variationnelle lui permet également d'être en mesure d'exécuter des tâches d'inférence qui seront mises à profit pour notamment reconstruire les vues de données endommagées. Nous introduirons dans une première section des notations spécifiques à ce chapitre, puis dans la section qui suivra nous ferons un bref état de l'art des méthodes de la CCA, ces auto-encodeurs variationnels sur lesquels se basent notre modèle y seront également introduits, ensuite nous construirons notre modèle (*Multiview Variational Graph Canonical Correlation Analysis* - MVGCCA) et nous en présenterons les principales propriétés. Enfin, dans une dernière partie nous présenterons diverses expériences qui permettront de valider les propriétés de notre modèle MVGCCA.

## 4.2 Définitions

Avant d'aller plus en avant, il est important de comprendre la distinction faite dans ce chapitre entre *objets* et *éléments*. Ce qu'on appelle un *objet* c'est la source des différentes vues observées tandis que l'*élément* fait référence à l'ensemble des vues associées à l'*objet* dans le jeu de données. En d'autres mots, on parlera donc d'objet lorsqu'on se référera à ce qui est à l'origine des vues et on parlera d'élément lorsqu'on désignera l'ensemble des vues de cet objet dans le jeu de données (voir Fig. (4.1)).

---

2. Au sens où les deux problèmes ont des solutions identiques modulo une opération mineure.

### 4.2.1 Jeux de données multivues ...

Un jeu de données multivues qu'on notera  $\mathbf{X}$  est constitué de  $n$  éléments. Pour  $i = 1, \dots, n$  on note  $\mathbf{x}^i$ , la collection des caractéristiques (ou vues) du  $i$ -ième élément de  $\mathbf{X}$ . Chaque élément  $i$  a, a priori  $M$  vues, notés  $\mathbf{x}_m^i \in \mathbb{R}^{q_m}$  avec  $m = 1, \dots, M$  et  $q_m$  la dimension de la  $m$ -ième vue. Ainsi  $\mathbf{x}^i = \{\mathbf{x}_1^i, \dots, \mathbf{x}_M^i\}$ <sup>3</sup>. Par commodité de calcul, on introduit les matrices  $\mathbf{X}_m \in \mathbb{R}^{n \times q_m}$  relatives à chaque vue  $m$ , dont la  $i$ -ième ligne contient la vue  $m$  du  $i$ -ième élément du jeu de données, i.e  $\mathbf{X}_m(i, :) = \mathbf{x}_m^{iT}$ . On pourra également écrire que  $\mathbf{X} = \{\mathbf{X}_1, \dots, \mathbf{X}_M\}$ <sup>4</sup>. Une illustration d'un jeu de données multivues avec les notations introduites ci-dessus peut être trouvé Figure 4.1.

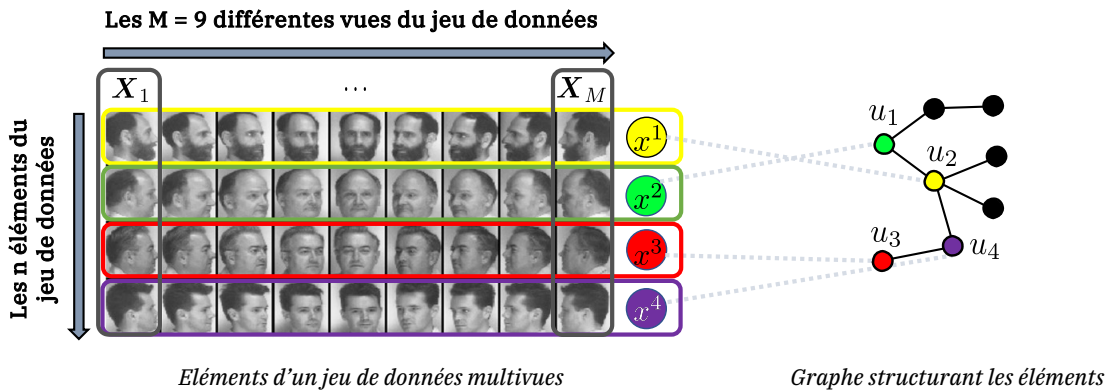


FIGURE 4.1 – *Jeu de données multivues structuré*. Ici représentée à gauche, une fraction d'un jeu de données multivues constituée des portraits de plusieurs individus capturés selon différents angles. Les différents angles de prises de vues correspondent aux vues. Les objets qui sont la source des vues sont les individus (eux et non pas leurs photos). Pour chacun d'entre eux il y a des photos prises selon  $M$  angles  $\theta_m$ . Pour l'individu  $i$ , le portrait capturé selon l'angle  $\theta_m$  est une de ses caractéristiques et est noté  $\mathbf{x}_m^i$ . L'ensemble des portraits d'un individu  $i$  est un élément du jeu de données multivues et correspond à l'ensemble  $\mathbf{x}^i = \{\mathbf{x}_1^i, \dots, \mathbf{x}_M^i\}$ . Pour un angle de prise de vue fixé  $\theta_m$ , l'ensemble des portraits (aplatis pour être mis sous forme de vecteur) pris selon cet angle constitue les lignes de la matrice  $\mathbf{X}_m$ , avec  $\mathbf{X}_m(i, :) = \mathbf{x}_m^{iT}$ . Les objets sous-jacents à l'origine des vues (i.e les individus) peuvent être liés via une structure qu'on peut voir à droite. Dans cet exemple, on peut imaginer, les différents individus sont en relation au sein d'un réseau social. Si par exemple on souhaite déterminer leurs âges, leurs portraits mais aussi leurs liens dans le réseau sont des informations pertinentes car les membres d'un réseau social tendent à être liés aux membres d'un âge similaire.

3.  $\mathbf{x}^i$  n'est pas un vecteur mais une collection de vecteurs mais par souci d'harmonisation, il sera noté en gras.

4. De même  $\mathbf{X}$ , n'est pas une matrice mais un ensemble de matrices mais par souci d'harmonisation, il sera aussi noté en gras.

### 4.2.2 ... et localement structurés

L'une des hypothèses de ce travail est que les objets qui sont à la source des données multivues sont localement structurés, et ces relations peuvent être mis à profit dans diverses tâches d'apprentissage (voir Fig. (4.1)). Cette structure sera décrite comme un graphe pondéré. Les mêmes notations introduites dans le Chapitre 2 seront utilisées pour le décrire mathématiquement. Par ailleurs, pour faciliter la discussion mathématique sur les jeux de données multivues structurés, on introduit des notations supplémentaires qui permettent de désigner les éléments ou spécifiquement des vues dans les voisinages. Le voisinage multivues qui contient toutes les vues d'un sommet et de ses voisins de profondeur inférieure à  $p$  relatif à une matrice d'adjacence  $\mathbf{A}$  s'écrit :

$$\mathcal{V}_p(\mathbf{x}^i) = \{\mathbf{x}^j | u_j \in \mathcal{V}_p(u_i)\} \quad (4.1)$$

et de même pour une vue particulière  $m$  :

$$\mathcal{V}_p(\mathbf{x}_m^i) = \{\mathbf{x}_m^j | u_j \in \mathcal{V}_p(u_i)\} \quad (4.2)$$

## 4.3 Etat de l'art

Dans cette section, nous présenterons la CCA linéaire et nous donnerons des éléments de compréhension pour s'en forger une intuition. Nous poursuivrons avec les principaux développements de la CCA, en nous attardant d'abord sur les approches algébriques puis sur les approches probabilistes. Nous terminerons cette partie en présentant les auto-encodeurs variationnels qui sont au cœur de notre modèle.

### 4.3.1 CCA linéaire

Considérons un ensemble multivues  $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2\}$  avec  $M = 2$  vues. Alors  $\mathbf{X}_1 \in \mathbb{R}^{n \times q_1}$  et  $\mathbf{X}_2 \in \mathbb{R}^{n \times q_2}$  sont les matrices correspondant aux vues de dimension  $q_1$  et  $q_2$  pour les  $n$  éléments du jeu de données. Pour appliquer la CCA linéaire, on choisit une dimension  $q$  tel que  $d \ll \min(q_1, q_2)$ . On cherche alors les meilleurs *projecteurs linéaires*  $\mathbf{U}_1 \in \mathbb{R}^{q_1 \times q}$  et  $\mathbf{U}_2 \in \mathbb{R}^{q_2 \times q}$  tels que les corrélations entre les projections  $\mathbf{X}_1 \mathbf{U}_1$  et  $\mathbf{X}_2 \mathbf{U}_2$  soient maximales. Intuitivement, l'information en commun entre les différentes vues caractérise les propriétés intrinsèques de l'objet à la source des vues, cette information est donc la plus à même d'être discriminante entre ces objets. Ces représentations sont donc pertinentes pour effectuer des tâches d'apprentissage sur ces objets. Pour réaliser cette maximisation, on agit composante par composante, ce qui veut dire qu'on maximise les corrélations entre les composantes des deux projections à travers l'ensemble des éléments du jeu de données ce qui s'écrit :

$$\max_{\mathbf{U}_1, \mathbf{U}_2} \sum_{p=1}^q \text{corr}(\mathbf{X}_1 \mathbf{U}_1(:, p), \mathbf{X}_2 \mathbf{U}_2(:, p)) \quad (4.3)$$

i.e,

$$\max_{\mathbf{U}_1, \mathbf{U}_2} \sum_{p=1}^q \frac{\mathbf{U}_1^T(p, :)\mathbf{X}_1^T\mathbf{X}_2\mathbf{U}_2(:, p)}{\sqrt{\mathbf{U}_1^T(p, :)\mathbf{X}_1^T\mathbf{X}_1\mathbf{U}_1(:, p)\mathbf{U}_2^T(p, :)\mathbf{X}_2^T\mathbf{X}_2\mathbf{U}_2(:, p)}} \quad (4.4)$$

On reconnaît les matrices de corrélations (régularisées<sup>5</sup>) empiriques des deux vues,  $\Sigma_{11}$ ,  $\Sigma_{22}$  :

$$\Sigma_{mm} = \frac{1}{n-1}\mathbf{X}_m^T\mathbf{X}_m + r_m\mathbf{I}_{q_m} \quad (r_m > 0) \quad (m \in \{1, 2\}) \quad (4.5)$$

et la matrice de corrélation croisée empirique  $\Sigma_{12} = \frac{1}{n-1}\mathbf{X}_1^T\mathbf{X}_2$ . On peut remarquer que la fonction objectif de l'Equation (4.4) reste invariante par multiplication des lignes de  $\mathbf{U}_1$  et  $\mathbf{U}_2$ , on peut donc imposer librement :

$$\forall m \in \{1, 2\}, \forall p \in \llbracket 1, q \rrbracket, \mathbf{U}_m^T(p, :)\mathbf{X}_m^T\mathbf{X}_m\mathbf{U}_m(:, p) = 1 \quad (4.6)$$

Pour tout  $p \in \llbracket 1, q \rrbracket$ ,  $\mathbf{X}_1\mathbf{U}_1(p, :)$  et  $\mathbf{X}_2\mathbf{U}_2(p, :)$  forment la  $p$ -ième paire de *variables canoniques*. Chacune de ces paires explique une partie de la corrélation croisée entre les deux vues, la quantification de cette explicabilité est appelée *corrélation canonique* et vaut  $\mathbf{U}_1^T(p, :)\mathbf{X}_1^T\mathbf{X}_2\mathbf{U}_2(:, p)$ . Ces corrélations canoniques sont en ordre décroissant, en effet chaque nouvelle paire de variables canoniques explique une part supplémentaire des corrélations entre les vues mais toujours une part inférieure à celle de la paire de variables canoniques précédente.

$$\forall p, p' \in \llbracket 1, q \rrbracket, p < p' \implies \mathbf{U}_1^T(p, :)\mathbf{X}_1^T\mathbf{X}_2\mathbf{U}_2(:, p) > \mathbf{U}_1^T(p', :)\mathbf{X}_1^T\mathbf{X}_2\mathbf{U}_2(:, p') \quad (4.7)$$

On peut faire un parallèle entre la CCA et l'analyse en composantes principales (*Principal Component Analysis* - PCA) qui étant donné un jeu de données (avec une seule vue) cherche des variables (dites composantes principales) non corrélées permettant d'expliquer au mieux la variabilité i.e la variance dans un jeu de données. Chaque composante principale explique ainsi une part maximale de la variance qui est *indépendante* des composantes principales précédentes (cette part est donc inférieure aux précédentes). De manière analogue, on souhaite que les variables canoniques satisfassent la propriété de non-corrélation les unes des autres, pour éviter notamment la redondance d'informations, ce qui se traduit par les conditions suivantes :

$$\forall m \in \{1, 2\}, \forall p, p' \in \llbracket 1, q \rrbracket, p \neq p', \mathbf{U}_m^T(p, :)\mathbf{X}_m^T\mathbf{X}_m\mathbf{U}_m(:, p') = 0 \quad (4.8)$$

Sous ces conditions (Eq. (4.7) et (4.8)), le problème de l'Equation (4.4) prend une forme simple sous laquelle on peut la retrouver dans la littérature :

---

5.  $r_1$  et  $r_2$  sont des paramètres de régularisation qui permettent d'éviter des matrices de corrélations dégénérées (i.e non inversibles) ou non pertinentes (i.e du bruit) [Bie et al. 2002].

$$\max_{\mathbf{U}_1, \mathbf{U}_2} \text{tr}(\mathbf{U}_1^T \boldsymbol{\Sigma}_{12} \mathbf{U}_2) \quad \text{t.q.} \quad \mathbf{U}_m^T \boldsymbol{\Sigma}_{mm} \mathbf{U}_m = \mathbf{I}_{q_m} \text{ pour } m \in \{1, 2\}. \quad (4.9)$$

Ce problème est équivalent à une autre forme qu'on peut également trouver dans la littérature et qui donne une meilleure intuition géométrique de la CCA :

$$\min_{\mathbf{U}_1, \mathbf{U}_2} \|\mathbf{X}_1 \mathbf{U}_1 - \mathbf{X}_2 \mathbf{U}_2\|_{\mathcal{F}}^2 \quad \text{t.q.} \quad \mathbf{U}_m^T (\mathbf{X}_m^T \mathbf{X}_m) \mathbf{U}_m = \mathbf{I}_{q_m} \text{ pour } m \in \{1, 2\}. \quad (4.10)$$

Maximiser les corrélations croisées entre les deux vues équivaut donc à minimiser la somme des distances au carré entre les points projetés des deux vues dans l'espace de projection, avec comme contrainte l'orthogonalité de ces projections. La résolution de ce problème est simple, il suffit de réaliser une décomposition en éléments propres [Andrew et al. 2013], le couple solution  $(\mathbf{U}_1^*, \mathbf{U}_2^*)$  est ainsi donné par :

$$(\mathbf{U}_1^*, \mathbf{U}_2^*) = (\boldsymbol{\Sigma}_{11}^{-\frac{1}{2}} \mathbf{U}_d, \boldsymbol{\Sigma}_{22}^{-\frac{1}{2}} \mathbf{V}_d) \quad (4.11)$$

où  $\mathbf{U}_d \in \mathbb{R}^{q_1 \times q}$  (resp.  $\mathbf{V}_d \in \mathbb{R}^{q_2 \times q}$ ) sont les  $q$  premières valeurs propres à gauche (resp. à droite) de la matrice  $\mathbf{T} = \boldsymbol{\Sigma}_{11}^{-\frac{1}{2}} \boldsymbol{\Sigma}_{12} \boldsymbol{\Sigma}_{22}^{-\frac{1}{2}}$ .

En conclusion, la CCA linéaire présente deux volets, d'une part via le calcul des *corrélations canoniques*, cette méthode permet de quantifier les liens existant entre deux jeux de données (en l'occurrence  $\mathbf{X}_1$  et  $\mathbf{X}_2$ ) et d'évaluer leurs degrés de dépendance linéaire. Des corrélations canoniques faibles rendent compte de l'absence d'information commune entre les deux vues et permettent donc de se rendre compte qu'on ne peut se passer de l'une ou l'autre des vues. A l'inverse, des corrélations canoniques importantes témoignent d'une information commune importante et donc de la possibilité de se passer de l'une ou l'autre des vues, en utilisant l'une des représentations obtenues appelés *variables canoniques*. Ces variables expliquent au mieux la variabilité à la fois *au sein* des deux vues et *entre* les vues, leur permettant d'avoir un pouvoir prédictif renforcé par rapport aux vues elles-mêmes. Ces améliorations ont pu être constatées empiriquement (voir Fig. (4.2) [Chen et al. 2018, 2019]). Ces propriétés font de la CCA linéaire une méthode très intéressante pour traiter les jeux de données multivues, c'est pourquoi de nombreuses extensions ont été proposées pour étendre cette méthode notamment à des jeux de données avec plus de vues ( $M > 2$ ) et mettant en jeu des relations plus complexes i.e non-linéaires.

### 4.3.2 Méthodes Algébriques

**Extension non linéaire.** Les formulations de la CCA linéaire peuvent facilement s'étendre pour capturer des relations non linéaires. Il suffit pour cela de remplacer les vues  $\mathbf{X}_1$  et  $\mathbf{X}_2$  dans l'Equation (4.10) par des fonctions non linéaires dépendant de ces vues.

- Kernel CCA [Akaho 2001 ; Ling and Fyfe 2000], l'astuce du noyau peut être avantageusement utilisée car la fonction objectif de la CCA linéaire (voir Eq. (4.4)) ne dépend que des produits scalaires entre les vues. La complexité de cette méthode est quadratique.



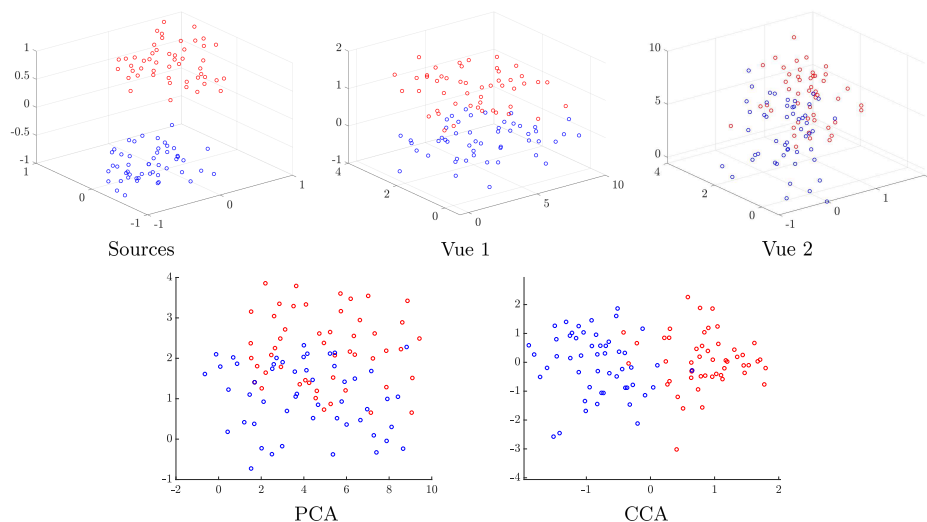


FIGURE 4.2 – **PCA vs CCA**. On peut voir sur cette figure comment la CCA permet de retrouver partiellement la structure de l'espace de la source. **En haut à gauche**, on a généré deux nuages de points dans  $\mathbb{R}^3$ . En bleu (resp. rouge) les coordonnées ont été tirées selon une loi uniforme  $\mathcal{U}([-1, 0])$  (resp.  $\mathcal{U}([0, 1])$ ). **En haut au milieu**, la première vue de la source, est générée en ajoutant un bruit uniforme différent à chaque composante du nuage de points de la source (dans l'ordre des composantes les bruits sont tirés selon  $\mathcal{U}([-9, 9])$ ,  $\mathcal{U}([-3, 3])$  et  $\mathcal{U}([-1, 1])$ ). **En haut à droite**, *idem* que pour la vue 1, les mêmes bruits ont été utilisés mais dans l'ordre inverse des composantes. **En bas à gauche**, visualisation des composantes principales (centrées normées) de la vue 2. On retrouve la séparation des éléments de la source mais elle n'est pas aussi nettement marquée que dans l'espace de la source. **En bas à droite**, visualisation des variables canoniques de la vue 2. On voit clairement que la séparation entre les deux groupes de la source est bien plus marquée que pour la PCA. Lorsque les bonnes hypothèses sont vérifiées, les variables canoniques représentent de bien meilleures représentations de la source que les vues elles-mêmes.

- Deep CCA [Andrew et al. 2013; Benton et al. 2019], la fonction non linéaire utilisée est un réseau de neurones. Cette approche permet un apprentissage par *batch* de la fonction objectif et rend la méthode scalable. Cela se fait au coût d'un certain nombre d'hyperparamètres à gérer. De plus, durant l'entraînement à chaque itération, on doit effectuer une décomposition en éléments propres sur une matrice de la taille du *batch* au carré et rétropropager le gradient à travers cette opération, ce qui rend la méthode numériquement instable.
- Autoencoder CCA [Wang et al. 2015], cette méthode est similaire à DeepCCA et a donc les mêmes avantages et les mêmes désavantages. Dans ce modèle, on souhaite que la fonction non linéaire (encore un réseau de neurones) conserve un maximum d'information sur les vues initiales, ce qui se traduit dans le modèle par une contrainte qui impose à celui-ci d'être en capacité de retrouver les vues à partir des représentations.

Méthode	Complexité	Non Linéaire	>2 vues	Graphe	Robustesse
CCA	$O(n)$	✗	✗	✗	✗
Kernel CCA	$O(n^2)$ <sup>6</sup>	✓	✗	✗	✗
Deep CCA	$O(n)$	✓	✗	✗	✗
GMCCA	$O(n^2)$	✗	✓	✓	✗
VCCA(p)	$O(n)$	✓	✗	✗	✗
VPCCA	$O(n)$	✗	✓	✗	✗
MVGCCA	$O(n)$	✓	✓	✓	✓

TABLE 4.1 – *Propriétés principales des méthodes dérivées de la CCA.*  $n$  est le nombre d'éléments dans le jeu de données. La colonne *Graphe* indique si le modèle est capable de tenir compte d'une potentielle structure entre les éléments. La colonne *robustesse* indique quant à elle si le modèle est capable de tenir compte de vues manquantes dans un contexte réel (qui correspond au scénario 2 présenté en Figure 4.5).

**Extension à plus de deux vues.** La manière la plus naturelle d'étendre la CCA à un nombre quelconque de vues (*Multiview Canonical Correlation Analysis* - MCCA) consiste à maximiser toutes les corrélations croisées entre toutes les paires possibles de vues, cette formulation (appelée SUMCOR) de la MCCA [Horst 1961a,b] s'écrit :

$$\min_{(\mathbf{U}_m)_{m=1}^M, \mathbf{S}} \sum_{\substack{m=1 \\ m' > m}}^M \|\mathbf{X}_m \mathbf{U}_m - \mathbf{X}_{m'} \mathbf{U}_{m'}\|_{\mathcal{F}}^2 \quad \text{t.q.} \quad \mathbf{U}_m^T (\mathbf{X}_m^T \mathbf{X}_m) \mathbf{U}_m = \mathbf{I}_{q_m}. \quad (4.12)$$

Malheureusement, ce problème est difficile, il est réputé NP-Difficile [Rupnik et al. 2013]. Pour surmonter cette difficulté, de nombreuses relaxations ont été proposées [Kettenring 1971 ; Nzobounsana and Gaymard 2010], aucune d'entre elles ne se démarque fortement des autres. Toutefois, on peut s'intéresser à une formulation populaire due à Carroll [1968] qui consiste à introduire un auxiliaire de calcul  $\mathbf{S} \in \mathbb{R}^{n \times q}$  qui fait office de représentation unique pour les éléments. L'objectif de cette formulation est de déterminer les projecteurs  $\{\mathbf{U}_m\}_{m=1}^M$  qui maximisent les corrélations croisées entre  $\mathbf{S}$  et les projections  $\{\mathbf{X}_m \mathbf{U}_m\}_{m=1}^M$  :

$$\min_{(\mathbf{U}_m)_{m=1}^M, \mathbf{S}} \sum_{m=1}^M \|\mathbf{X}_m \mathbf{U}_m - \mathbf{S}\|_F^2 \quad \text{t.q.} \quad \mathbf{S}^T \mathbf{S} = \mathbf{I}_q. \quad (4.13)$$

Cette formulation ne donne accès qu'à une mesure indirecte des corrélations canoniques, mais en contrepartie elle peut être résolue facilement via une décomposition en éléments propres de la matrice  $\sum_{m=1}^M \mathbf{X}_m (\mathbf{X}_m^T \mathbf{X}_m) \mathbf{X}_m^T$  [Chen et al. 2018]. Ainsi les  $q$  premiers vecteurs propres de cette matrice constituent les  $q$  colonnes de la solution  $\mathbf{S}^*$ . L'intérêt de cette formulation est qu'elle permet d'intégrer facilement des contraintes structurelles sur  $\mathbf{S}^*$  comme nous allons le voir. Le coût de construction de cette matrice (et de la méthode) est quadratique.

**Extension aux données multivues structurées.** [Chen et al. \[2019\]](#) ont proposé d'étendre la méthode multivues de [Carroll \[1968\]](#) aux données multivues structurées. Ils ont montré que la prise en compte d'une potentielle structure pouvait permettre d'augmenter la qualité des représentations, notamment pour faire du *clustering*. Dans leur modèle (*Graph Aware Multiview Canonical Correlation Analysis* - GMCCA), la structure est prise en compte en s'assurant que le signal que constitue  $\mathbf{S}$  sur ce graphe soit lisse. Pour cela, ils ajoutent un terme favorisant la continuité de  $\mathbf{S}$  sur le graphe à la fonctionnelle (4.13). En considérant le laplacien  $\mathbf{L}$  du graphe, ce terme s'écrit avec  $\gamma > 0$  un hyperparamètre :

$$\gamma \text{tr}(\mathbf{S}^T \mathbf{L} \mathbf{S}) = \gamma \sum_{\substack{i=1 \\ j \in \mathcal{V}(i)}}^n \|\mathbf{S}(i, :) - \mathbf{S}(j, :)\|_{\mathcal{F}}^2 \quad (4.14)$$

Ce terme permet de rendre  $\mathbf{S}$  plus lisse sur le graphe. D'où la nouvelle fonctionnelle :

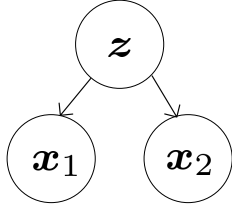
$$\min_{(\mathbf{U}_m)_{m=1}^M, \mathbf{S}} \sum_{m=1}^M \|\mathbf{X}_m \mathbf{U}_m - \mathbf{S}\|_{\mathcal{F}}^2 + \gamma \text{tr}(\mathbf{S}^T \mathbf{L} \mathbf{S}) \quad \text{t.q.} \quad \mathbf{S}^T \mathbf{S} = \mathbf{I}_q. \quad (4.15)$$

La solution  $\mathbf{S}^*$  de ce problème a ses colonnes égales aux  $q$  premiers vecteurs propres de la matrice  $\sum_{m=1}^M \mathbf{X}_m (\mathbf{X}_m^T \mathbf{X}_m) \mathbf{X}_m^T - \gamma \mathbf{L}$  [[Chen et al. 2018](#)]. Tout comme pour la méthode de [Carroll \[1968\]](#), le coût de résolution est quadratique. Cette méthode est l'unique méthode de la littérature permettant de traiter les données multivues structurées.

**Remarque 17** : *L'objectif de notre travail est ainsi de proposer une alternative scalable à cette méthode. L'idée la plus simple consiste tout simplement à combiner la fonction objectif de DeepCCA ou de AutoencoderCCA avec celle de GMCCA. C'est la première approche que nous avons initialement adopté mais elle a été infructueuse, les résultats de ces expériences peuvent être trouvés en Annexe A.3.1. Ces résultats négatifs nous ont poussé à chercher un autre angle d'approche pour traiter ce problème, c'est ainsi que nous nous sommes intéressés à une interprétation probabiliste de la CCA.*

### 4.3.3 Méthodes Probabilistes

**Interprétation probabiliste de la CCA - PCCA.** [Bach and Jordan \[2005\]](#) ont montré que les projections linéaires optimales (Eq. (4.11)) pouvaient être retrouvées à partir d'un modèle graphique dans lequel chaque couple de vues relatives à un *objet* est expliqué par une variable *dite* latente notée  $\mathbf{z} \in \mathbb{R}^q$  (qui est donc une représentation de l'*objet*). Pour décrire le modèle, on introduit d'une part,  $p$  la distribution suivie par la variable latente ( $\mathbf{z} \sim p$ ) et d'autre part pour chaque vue  $m$ ,  $p_{\theta_m}(\mathbf{x}_m | \mathbf{z})$  la probabilité conditionnelle (paramétrée par  $\theta_m$ ) d'observer une certaine vue  $\mathbf{x}_m \in \mathbb{R}^{d_m}$  sachant que la source est  $\mathbf{z}$  (on appelle cette distribution un décodeur ; elle fait la transition entre l'espace latent et les vues). Le modèle graphique s'écrit  $\forall m \in \{1, 2\}$ ,



Modèle graphique de la PCA  
probabiliste.

$$\begin{aligned} z &\sim p = \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d) \\ \mathbf{x}_m &\sim p_{\theta_m}(\cdot|z) = \mathcal{N}(\mathbf{W}_m z + \boldsymbol{\mu}_m, \boldsymbol{\Psi}_m) \end{aligned} \quad (4.16)$$

avec  $\boldsymbol{\mu}_m \in \mathbb{R}^{q_m}$ ,  $\mathbf{W}_m \in \mathbb{R}^{q_m \times q}$  and  $\boldsymbol{\Psi}_m \in \mathbb{R}^{q_m \times q_m} \succcurlyeq 0$  (matrice semi-définie positive). Pour chaque vue, l'ensemble de ces paramètres est noté  $\theta_m$ , i.e  $\theta_m = (\mathbf{W}_m, \boldsymbol{\mu}_m, \boldsymbol{\Psi}_m)$ . Tandis que l'ensemble des paramètres du modèle est noté  $\theta = (\theta_1, \theta_2)$ . Les paramètres optimaux  $\theta^*$  qui expliquent le mieux les données sous les hypothèses de modèles, sont obtenus en maximisant le log de la fonction de vraisemblance de l'ensemble de données, c'est-à-dire :

$$\log p_{\theta}(\mathbf{X}) = \log p_{\theta}(\mathbf{X}_1, \mathbf{X}_2) = \sum_{i=1}^n \sum_{m=1}^2 \log \int_{\mathbb{R}^q} p_{\theta_m}(\mathbf{x}_m^i|z) p(z) dz. \quad (4.17)$$

Pour pouvoir calculer les représentations des éléments multivues, il nous faut calculer la distribution postérieure  $p_{\theta}(\mathbf{z}|\mathbf{x}_m^i)$  (qui elle sera appelée l'encodeur car elle fait la transition entre les vues et l'espace latent). Le théorème de Bayes permet de la déterminer via la relation :

$$p_{\theta_m^*}(\mathbf{z}|\mathbf{x}_m^i) = \frac{p_{\theta_m^*}(\mathbf{x}_m^i|\mathbf{z})p(\mathbf{z})}{p_m(\mathbf{x}_m^i)}. \quad (4.18)$$

Cette expression fait intervenir la distribution inconnue  $p_m$  suivie par les vues  $m$  des éléments du jeu de données. [Bach and Jordan \[2005\]](#) nous apprennent que l'espérance de la distribution postérieure est exactement la projection optimale (Eq. (4.11)) de la CCA linéaire (Eq. (4.10)).

$$\mathbb{E}_{\mathbf{z} \sim p_{\theta_m^*}}(\mathbf{z}|\mathbf{x}_m^i) = \mathbf{M}_m^T \mathbf{U}_m^* \mathbf{x}_m^i. \quad (4.19)$$

Les solutions de ces deux problèmes sont donc identiques à la multiplication près par des matrices  $\mathbf{M}_m \in \mathbb{M}^{q \times q}$  ( $m = \{1, 2\}$ ) telles que  $\mathbf{M}_1^T \mathbf{M}_2 = \mathbf{P}_q$  où  $\mathbf{P}_q$  est la matrice des  $q$  premières corrélations canoniques évoquées en Section 4.3.1. Comme on peut le voir, ce modèle graphique est équivalent à la CCA linéaire. Ce modèle peut être facilement étendu à des ensembles de données avec  $M > 2$ , il suffit d'introduire autant de décodeurs qu'il y a de vues. Néanmoins, résoudre un problème de maximisation tel que celui de l'Equation (4.17) (i.e maximisation de la vraisemblance d'une distribution multi-dimensionnelle) est souvent impossible. En effet, le calcul de la log-vraisemblance requiert une intégration rédhibitoire dans tout l'espace latent. De plus, même une approximation de cette fonctionnelle (par un échantillonnage de Monte-Carlo par exemple) n'est pas suffisante pour résoudre entièrement le problème. En effet, quand bien

même pourrions nous déterminer  $\theta^*$ , comme les véritables distributions  $p_m$  sont inconnues, nous ne pourrions pas déterminer les décodeurs optimaux  $p_{\theta^*}(\mathbf{z}|\mathbf{x}_m^i)$  (Eq. (4.18)). L'approche variationnelle que nous allons voir permet de surmonter simultanément ces deux problématiques et notamment de résoudre ce problème avec une complexité linéaire en le nombre d'éléments  $O(n)$  (la même complexité que la CCA linéaire classique d'ailleurs). Un des intérêts des méthodes variationnelles est que cette complexité linéaire reste valable pour les extensions multivues  $M > 2$ , contrairement aux extensions de la CCA linéaire multivues  $M > 2$  qui ont souvent un coût quadratique. Autre point d'intérêt de ces modèles variationnelles, c'est qu'à la différence de la CCA linéaire (et de la plupart des méthodes algébriques) où l'introduction de nouvelles données nécessite que la méthode soit réexécutée sur l'ensemble des données (anciennes et nouvelles), ces méthodes peuvent calculer les représentations des nouvelles données sans entraînement supplémentaire.

#### 4.3.4 Auto-encodeur variationnel - VAE

Kingma and Welling [2014] ont montré qu'en introduisant des distributions  $q_\eta(\mathbf{z}|\mathbf{X}^i) = q_\eta(\mathbf{z}|\mathbf{x}_1^i, \mathbf{x}_2^i)$  paramétrées par  $\eta$  en lieu et place de l'inaccessible distribution  $p_\theta(\mathbf{z}|\mathbf{X}^i) = p_\theta(\mathbf{z}|\mathbf{x}_1^i, \mathbf{x}_2^i)$ , on pouvait minorer la log-vraisemblance (Eq. (4.17)). Cette borne inférieure est appelé borne inférieure variationnelle (*Evidence Lower BOund* - ELBO)<sup>7</sup>,

$$\log p_\theta(\mathbf{X}_1, \mathbf{X}_2) \geq \sum_{i=1}^n \mathbb{E}_{\mathbf{z} \sim q_\eta(\mathbf{z}|\mathbf{x}_1^i, \mathbf{x}_2^i)} [\log(p_\theta(\mathbf{x}_1^i, \mathbf{x}_2^i|\mathbf{z}))] - D_{\text{KL}}(q_\eta(\mathbf{z}|\mathbf{x}_1^i, \mathbf{x}_2^i) \| p(\mathbf{z})). \quad (4.20)$$

Maximiser cette borne permet de maximiser indirectement la log-vraisemblance. Le premier terme assure que les décodeurs soient bien les fonctions « réciproques » des encodeurs. Le second terme agit comme un terme de régularisation qui limite l'expressivité des décodeurs  $q_\eta$ , cela permet notamment d'établir une certaine continuité dans l'espace latent, les éléments avec des vues proches auront des représentations latentes proches. Pour maximiser l'ELBO il faut à la fois gérer les paramètres des encodeurs  $\theta$  mais aussi les paramètres  $\eta$  des décodeurs. Malgré l'ajout de ces nouveaux paramètres, cette formulation est très efficace, dès lors qu'on obtient les paramètres optimaux  $(\theta^*, \eta^*)$ , on obtient alors directement les décodeurs optimaux  $q_{\eta^*}$  sans connaissance préalable de  $p_m$ . De plus, cette fonction objective autorise l'entraînement par *batch* ce qui permet d'avoir une complexité linéaire (mais quadratique en la taille du *batch*). Ces avancées apportées par les VAE sur les modèles d'inférences bayésiens ont été exploitées avec succès pour construire des modèles de CCA probabilistes :

- Wang et al. [2016] ont introduit le premier modèle CCA s'appuyant sur un modèle variationnel (*Deep Variational Canonical Correlation Analysis* - *VCCA*), ce modèle est une adaptation directe de la PCCA avec le formalisme variationnel ; il se limite donc aux jeux de données présentant 2 vues. Toutefois dans ce modèle, les décodeurs ne sont plus paramétrés par des fonctions linéaires  $(\mathbf{W}_m, \boldsymbol{\mu}_m)$  (Eq. (4.16)) mais par des réseaux de neurones (tout comme les décodeurs). Ce changement permet à la méthode de pouvoir

7. On notera un abus de notation,  $q_\eta(\mathbf{z}|\mathbf{x}_1^i, \mathbf{x}_2^i)$  et  $p(\mathbf{z})$  sont identifiés à  $q_\eta(\cdot|\mathbf{x}_1^i, \mathbf{x}_2^i)$  et  $p$ . Ce type d'identification implicite se retrouvera dans d'autres équations de ce chapitre lorsqu'il n'y pas d'ambiguïté.

capturer des relations non linéaires. Parmi les méthodes partageants ses propriétés (limité à 2 vues, non linéaire et scalable - ex : DeepCCA -), c'est la méthode qui obtient les meilleurs résultats expérimentaux. Ces résultats valident la pertinence de l'approche variationnelle dans le cadre de la CCA. Par ailleurs, Wang et al. [2016] ont fait le choix de conditionner leurs encodeurs  $q_\eta(\mathbf{z}|\mathbf{x}_1^i)$  par une seule des 2 vues tout en attendant des décodeurs  $p_\theta(\mathbf{x}_1^i, \mathbf{x}_2^i|\mathbf{z})$  qu'ils puissent toujours décoder les deux vues. L'avantage de ce choix est qu'après avoir entraîné le modèle, on peut calculer les représentations de nouveaux éléments avec une seule des 2 vues. Toutefois, cela ne rend pas le modèle robuste, puisqu'il est inopérant si la vue en question est manquante même lorsque l'autre vue est disponible.

- Karami and Schuurmans [2020] ont proposé récemment un modèle multivues ( $M > 2$ ), généralisant la PCCA. Dans ce modèle, les encodeurs sont de nouveau paramétrés par des fonctions linéaires. De plus, contrairement au modèle précédent, ce modèle conditionne les encodeurs  $q_\eta(\mathbf{z}|\mathbf{x}_1^i, \dots, \mathbf{x}_M^i)$  avec chaque vue. C'est un modèle très efficace avec des résultats de référence dans l'état de l'art. Ce modèle propose également une forme de robustesse faible, le modèle permet en effet de calculer les représentations d'éléments nouveaux après l'entraînement lorsque seul un nombre restreint  $M' < M$  de vues *quelconques* est disponible. Toutefois, c'est à condition que les  $M' < M$  vues disponibles soient les mêmes pour tous les éléments. C'est un premier pas vers la robustesse mais on peut aller plus loin, dans la réalité les différents éléments peuvent avoir des vues manquantes distinctes. Un modèle robuste devrait pouvoir s'accommoder d'éléments présentant des vues manquantes différentes.

Notre objectif est de construire un modèle variationnel permettant la forme de robustesse évoquée plus haut, tout en étant capable d'intégrer les informations de structure, pour ce faire nous nous appuyons sur les auto-encodeurs variationnels pour graphes qui vont être introduits ci-dessous.

**Auto-encodeur variationnel pour graphe (GVAE).** Il est possible de tenir compte d'une structure locale en utilisant une extension des VAE proposée par Kipf and Welling [2016a] pour la prédiction d'arête dans un graphe. Le contexte est le suivant : on a un graphe dont seule une partie des sommets et des arêtes les liant est connue<sup>8</sup> (i.e la matrice d'adjacence  $\mathbf{A}$  est connue partiellement) ; les nœuds de ce graphe portent une caractéristique unique (ici  $M = 1$ ). L'objectif est de créer un modèle qui puisse, étant donné deux sommets et leurs caractéristiques, déterminer s'ils sont liés ou non. Avant d'introduire le modèle en question, remarquons que pour pouvoir décrire les relations entre des éléments, on ne pourra plus écrire notre ELBO comme une somme de termes qui ne dépendent que d'un élément (comme dans l'Equation (4.20)) et donc d'une seule variable latente. Pour pouvoir décrire les relations existantes entre les éléments, on introduit la matrice des variables latentes (à l'origine des vues des éléments)  $\mathbf{Z} \in \mathbb{R}^{n \times q}$  qui nous permet de pouvoir écrire la probabilité  $p_\theta(\mathbf{A}|\mathbf{Z})$  d'avoir une certaine matrice d'adjacence

8. C'est le cas dans un graphe dynamique comme celui d'un réseau social, après l'entraînement du modèle certains sommets ou liens sont susceptibles d'apparaître ou de disparaître. L'objectif est de construire un modèle capable de prédire ces évolutions, comme par exemple de prédire à qui un nouveau sommet (utilisateur) va se lier.

$\mathbf{A}$  sachant qu'on a les  $n$  variables latentes<sup>9</sup> que constituent les lignes de  $\mathbf{Z}$ . On construit alors un modèle variationnel dans l'optique de maximiser la log-vraisemblance d'avoir  $\mathbf{A}$  comme matrice d'adjacence structurant nos données. Dans ce cadre là, l'ELBO qui minore cette quantité s'écrit :

$$\log p_\theta(\mathbf{A}) \geq \mathcal{L}_{ELBO} = \mathbb{E}_{\mathbf{Z} \sim q_\eta(\mathbf{Z}|\mathbf{X}, \mathbf{A})}[\log(p_\theta(\mathbf{A}|\mathbf{Z}))] - D_{\text{KL}}(q_\eta(\mathbf{Z}|\mathbf{X}, \mathbf{A})\|p(\mathbf{Z})), \quad (4.21)$$

où  $q_\eta(\mathbf{Z}|\mathbf{X}, \mathbf{A})$  est la distribution paramétrique de l'encodeur qui est choisit comme étant paramétrés par un GCN,  $p(\mathbf{A}|\mathbf{Z})$ <sup>10</sup> est le décodeur du graphe (qui permet de faire la prédiction d'arête) et  $p(\mathbf{Z}) = \prod_{i=1}^n p(\mathbf{Z}(i, :))$  est la distribution de l'espace latent choisie comme une loi normale multivariée. De manière similaire à l'Equation (4.20), le premier terme de l'ELBO permet d'assurer que l'on puisse reconstruire le graphe à partir des variables latentes mais il ne permet pas d'assurer la reconstruction des données sur les nœuds (pour cela il faudrait remplacer  $p_\theta(\mathbf{A}|\mathbf{Z})$  par  $p_\theta(\mathbf{X}, \mathbf{A}|\mathbf{Z})$ ) puisque la tâche de reconstruction d'arêtes ne le nécessite pas. Nous nous appuyerons sur cette formulation pour proposer une approche scalable et géométrique (i.e structurelle) de la CCA.

## 4.4 *Multiview Graph Canonical Correlation Analysis*

Dans cette section, nous présentons notre modèle (*Multiview Variational Graph Aware Canonical Correlation Analysis*), qui est un modèle probabiliste, multivues pour la CCA qui peut prendre en compte des informations structurelles et notamment composer avec des données incomplètes. Nous considérons un jeu de données multivues  $X$  avec  $n$  éléments, ces éléments sont issus de  $n$  sources structurées selon un graphe que l'on connaît et de matrice d'adjacence  $\mathbf{A}$ .

### 4.4.1 Modèle : propriétés et mise en équation

En prenant comme point de départ le modèle de la PCCA 4.16, notre contribution est la suivante :

- On étend ce modèle pour  $M > 2$ . On introduit autant de décodeurs qu'il y a de nombre de vues  $M$ . Ces décodeurs sont paramétrés par des perceptrons multicouches différents pour chaque vue  $m$  (noté  $\text{MLP}_m$ ).
- On intègre des contraintes structurelles dans la construction de l'espace latent  $\mathbf{z}$ ; de manière similaire à [Kipf and Welling 2016a] (Eq. (4.21)), l'encodeur sera également conditionné par le graphe et le décodeur devra être en capacité de retrouver la structure mais aussi les vues à partir des vecteurs de l'espace latent.
- On choisira une forme particulière pour l'encodeur qui permettra de traiter des données contenant des vues manquantes et de les régénérer à partir des autres vues disponibles si nécessaire.

---

9. On peut en réalité décrire le modèle avec deux variables latentes seulement mais pour plus de clarté on en utilisera autant qu'il y a d'éléments.

10. Cette probabilité est choisie tel que  $p(\mathbf{A}|\mathbf{Z}) = \prod_{i,j} p_E(\mathbf{A}_{i,j}|\mathbf{Z}^T(:, i), \mathbf{Z}^T(:, j))$  ce qui permet lors de la phase de prédiction de l'utiliser pour n'importe quel nombre d'éléments.

Ces différents points se traduisent par le modèle graphique qu'on peut retrouver en Figure 4.3 et les équations qui vont suivre. Dans un premier temps, nous présentons d'abord les équations directement inspirées de la PCCA (Eq. (4.16)) qui permettent de décrire les décodeurs des vues à partir des vecteurs de l'espace latent,  $\forall m \in \{1, \dots, M\}$  :

$$\begin{aligned} \mathbf{z} &\sim \mathcal{N}(\mathbf{0}_q, \mathbf{I}_q); \\ \mathbf{x}_m &\sim p_{\theta_m}(\cdot|\mathbf{z}) = \mathcal{N}(\mathbf{W}_m^{\mu\text{dec}} \text{MLP}_m(\mathbf{z}), \mathbf{\Psi}_m). \end{aligned} \quad (4.22)$$

Comme ses prédécesseurs [Bach and Jordan 2005; Kingma and Welling 2014] ce modèle se base sur deux hypothèses. D'une part, la variable latente  $\mathbf{z}$  qui explique les vues suit une distribution normale multivariée; cette hypothèse peut paraître irréaliste mais c'est une hypothèse courante dans la littérature des auto-encodeurs [Kingma and Welling 2014; Kipf and Welling 2016a]. Elle se justifie car cette distribution est supposément la plus neutre permettant d'avoir une distribution suffisamment riche pour décrire un ensemble de données complexe mais suffisamment neutre pour ne pas biaiser le modèle. On peut néanmoins se passer de cette hypothèse en formulant des modèles variationnels légèrement différents, mais ces modèles sont généralement plus difficile à entraîner [Tomczak and Welling 2018; Takahashi et al. 2019]. Nous avons donc adopté l'hypothèse de la distribution normale multivariée communément admise pour l'espace latent qui en pratique, permet d'avoir de bonnes performances et permet d'avoir un cadre mathématique plus pratique : il est facile d'échantillonner des éléments selon cette distribution et cette hypothèse permet d'avoir une formulation explicite de l'ELBO. La deuxième hypothèse qui stipule que les distributions conditionnelles des vues par rapport aux variables latentes sont des distributions normales multivariées<sup>11</sup>, est choisie car elle permet d'une part de rester dans un cadre proche de celui de la PCCA, et d'autre part cette hypothèse combinée à la précédente permet d'avoir une forme explicite de la divergence de Kullback-Leibler dans l'ELBO (voir Fig. (4.35) et Annexe A.3.2)).

Les deux hypothèses ci-dessus permettent de totalement décrire les décodeurs des vues, pour intégrer les contraintes structurelles, il faut maintenant introduire le décodeur du graphe. Par hypothèse, notre ensemble multivues  $X$  à  $n$  éléments est associé à une matrice d'adjacence pondérée  $\mathbf{A} \in [0, 1]^{n \times n}$ <sup>12</sup>. Pour décrire ce modèle, tout comme dans la Section 4.3.4, il faut introduire la matrice de vecteur latente  $\mathbf{Z} \in \mathbb{R}^{n \times q}$ ; le modèle s'écrit ainsi<sup>13</sup> :

$$\begin{aligned} \mathbf{Z}(:, i) &\sim \mathcal{N}(\mathbf{0}_q, \mathbf{I}_q); \\ \mathbf{A} &\sim p_{\mathcal{G}}(\cdot|\mathbf{Z}). \end{aligned} \quad (4.23)$$

Pour décrire le décodeur de graphe  $p_{\mathcal{G}}$ , on va supposer que les poids du graphe sont des variables indépendantes. On introduit alors le décodeur de poids entre deux variables latentes  $\mathbf{z}$  et  $\mathbf{z}'$  dans  $\mathbb{R}^q$  qui est tel que,

11. Paramétrées par un perceptron multicouche pour la moyenne et une matrice de poids pour la matrice de covariance

12. C'est toujours possible d'être dans ces conditions quitte à normaliser la matrice d'adjacence.

13. La première ligne de cette équation est en réalité redondante avec celle de l'Equation (4.22).



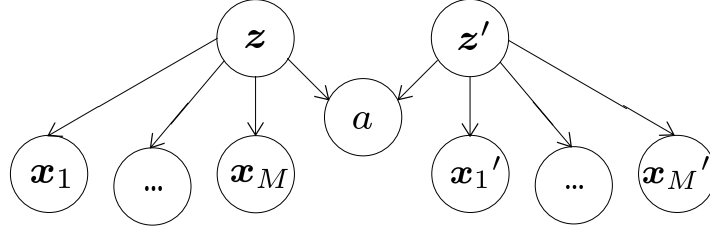


FIGURE 4.3 – *Modèle graphique de MVGCCA*. Les vues des éléments sont conditionnées par une variable latente tandis que les poids des arêtes entre ces éléments sont conditionnés par des paires de variables latentes.

$$\log p_E(\mathbf{A}_{i,j} | \mathbf{z}, \mathbf{z}') = \mathbf{A}_{i,j} \log \tau(\mathbf{z}^T \mathbf{z}') + (1 - \mathbf{A}_{i,j}) \log(1 - \tau(\mathbf{z}^T \mathbf{z}')) \quad (4.24)$$

où  $\tau$  est la fonction sigmoïde.  $p_E$  est une version continue de la loi de Bernoulli, qui est connue pour bien fonctionner avec les modèles variationnels [Loaiza-Ganem and Cunningham 2019]. Ce choix particulier de forme de distribution va pousser le modèle à aligner les représentations des éléments voisins dans le graphe (plus les éléments sont alignés plus leur produit scalaire est grand et donc plus la probabilité qu'il soient liés est grande). Finalement, la probabilité d'avoir une matrice d'adjacence  $\mathbf{A}$  étant donné  $\mathbf{Z}$  est :

$$p_G(\mathbf{A} | \mathbf{Z}) = \prod_{i=1}^n \prod_{j=1}^n p_E(\mathbf{A}_{i,j} | \mathbf{Z}^T(i, :), \mathbf{Z}^T(j, :)). \quad (4.25)$$

On peut remarquer que le décodeur de graphe est sans paramètre ce qui permet de limiter le potentiel de sur-apprentissage du modèle vis-à-vis de la structure. Les paramètres entraînaibles du modèle seront notés tous ensemble en  $\theta$ .

#### 4.4.2 Vraisemblance

L'hypothèse d'indépendance des vues est adjointe à celle d'indépendance des poids des sommets. Elle permet d'écrire la log-vraisemblance du jeu de données  $(\mathbf{X}, \mathbf{A})$  comme suit :

$$\log p_\theta(\mathbf{X}, \mathbf{A}) = \log(p_\theta(\mathbf{X}) p_G(\mathbf{A})) = \int_{\mathbb{R}^q} p_\theta(\mathbf{X} | \mathbf{Z}) p_G(\mathbf{A} | \mathbf{Z}) p(\mathbf{Z}) d\mathbf{Z}. \quad (4.26)$$

Dans cette équation,  $p_\theta(\mathbf{X}) = p_\theta(\{\mathbf{X}_m\}_{m=1}^M)$  est la probabilité jointe sur tous les éléments et toutes les vues de  $\mathbf{X}_1$  à  $\mathbf{X}_M$  ; l'Equation (4.26) est donc le logarithme de la probabilité d'obtenir ce jeu de données multivues avec ce graphe de matrice d'adjacence  $\mathbf{A}$  étant donné les paramètres  $\theta$  du modèle. L'expression peut être développée ainsi :

$$\begin{aligned}
\log p_\theta(\{\mathbf{X}_m\}_{m=1}^M) &= \sum_{i=1}^n \sum_{m=1}^M \log \int_{\mathbb{R}^q} p_{\theta_m}(\mathbf{X}_m^i | \mathbf{z}) p(\mathbf{z}) d\mathbf{z}; \\
\log p_G(\mathbf{A}) &= \sum_{i=1}^n \sum_{j=1}^n \log \int_{\mathbb{R}^q} p_E(\mathbf{A}_{i,j} | \mathbf{z}, \mathbf{z}') p(\mathbf{z}) p(\mathbf{z}') d\mathbf{z} d\mathbf{z}'.
\end{aligned} \tag{4.27}$$

Pour les mêmes raisons que dans l'Equation (4.17), cette expression ne peut être maximisée facilement et de même l'ignorance des vraies distributions des vues  $p_m$  implique l'impossibilité de calculer les encodeurs associés  $p_\theta(\mathbf{z} | \mathbf{x}_M^i)$  aux décodeurs définis en (4.22); pour résoudre ces problèmes, on s'en remet donc à l'utilisation d'un auto-encodeur variationnel.

### 4.4.3 Approche variationnelle

**Encodeurs paramétriques.** Tout comme en Section 4.3.4, on introduit la distribution postérieure  $q_\eta(\mathbf{Z} | \mathbf{X}, \mathbf{A})$  en lieu et place de  $p_\theta(\mathbf{Z} | \mathbf{X}, \mathbf{A})$ . Cette distribution est paramétrée par un GCN, sa dépendance aux vues s'écrit (avec un abus de notation) comme suit :

$$q_\eta(\mathbf{Z} | \mathbf{X}, \mathbf{A}) = \prod_{i=1}^n q_\eta(\mathbf{Z}(i, :) | \mathcal{V}_p(\mathbf{x}_M^i), \mathbf{A}) = \prod_{i=1}^n \prod_{m=1}^M q_{\eta_m}(\mathbf{Z}(i, :) | \mathcal{V}_p(\mathbf{x}_M^i), \mathbf{A}). \tag{4.28}$$

Les paramètres  $\eta_m$  sont la collection des paramètres entraînaibles relatifs à l'encodeur  $q_{\eta_m}$  de chaque vue  $m$  et on a noté  $\eta = (\eta_1, \dots, \eta_M)$ . Le paramètre de profondeur de voisinage  $p$  dépend de la paramétrisation du GCN et sera décidé dans la partie expérimentale. On choisit chaque  $q_{\eta_m}$  comme une distribution gaussienne multivariée (de matrice de covariance diagonale) paramétrisée par le GCN nommé **Krylov**<sup>14</sup> (voir expression Equation 2.50) :

$$\begin{aligned}
q_{\eta_m}(\mathbf{z} | \mathcal{V}_p(\mathbf{x}_M^i), \mathbf{A}) &= \mathcal{N}(\boldsymbol{\mu}_m^{\text{enc}}(i, :), \text{diag}(\boldsymbol{\sigma}_m^{\text{enc}2}(i, :))). \\
\boldsymbol{\mu}_m^{\text{enc}} &= \mathbf{Krylov}_m(\mathbf{X}_m, \mathbf{A})(\mathbf{W}_m^{\mu^{\text{enc}}})^T. \\
\log \boldsymbol{\sigma}_m^{\text{enc}2} &= \mathbf{Krylov}_m(\mathbf{X}_m, \mathbf{A})(\mathbf{W}_m^{\sigma^{\text{enc}}})^T.
\end{aligned} \tag{4.29}$$

où  $\boldsymbol{\mu}_m^{\text{enc}}$  et  $\log \boldsymbol{\sigma}_m^{\text{enc}2}$  sont les matrices renvoyées par le GCN, dont les  $i$ -ième lignes correspondent respectivement à la moyenne et la diagonale de la matrice de covariance de la distribution  $q_{\eta_m}(\mathbf{z} | \mathcal{V}_p(\mathbf{x}_M^i), \mathbf{A})$ . La matrice de covariance est choisie diagonale car cela simplifie le modèle sans en dégrader les performances. Le choix du GCN est justifié car on souhaite que tous les éléments soient encodés en intégrant les informations de sa position au sein de la structure, or les GCN sont des fonctions efficaces pour extraire ce type d'information en tenant compte notamment des voisinages [Defferrard et al. 2016; Kipf and Welling 2016b]. Il se trouve que Wu et al. [2020] ont montré que les nombreuses méthodes de GCN existantes ont des performances similaires, ce choix n'étant pas critique nous avons simplement réutilisé Krylov qui a été introduit

14. Le log dans  $\log \boldsymbol{\sigma}_m^{\text{enc}2}$  est appliqué à chaque élément de la matrice  $\boldsymbol{\sigma}_m^{\text{enc}}$ .

dans le chapitre précédent. Toutefois, on peut rappeler que Luan et al. ont montré que Krylov était un GCN avec de bonnes propriétés de conservation de l'information, notamment lorsque le réseau est profond. Cette discussion peut être retrouvée en Section 2.3.5.

**Remarque 18 (Variance.)** *On calcule le logarithme de la variance plutôt que la variance elle-même pour assurer facilement la positivité de cette dernière.*

**Fonction objectif.** Pour déterminer l'ELBO de ce modèle, on prend comme point de départ la positivité de la divergence de Kullback-Leibler entre l'encodeur paramétrique et l'encodeur réel (qu'on ne sait pas calculer) calculé via la formule de Bayes :

$$D_{\mathbf{KL}}(q_\eta(\mathbf{Z}|\mathbf{X}, \mathbf{A})\|p_\theta(\mathbf{Z}|\mathbf{X}, \mathbf{A})) \geq 0. \quad (4.30)$$

Ce qui équivaut à :

$$\int_{\mathbb{R}^{n \times q}} q_\eta(\mathbf{Z}|\mathbf{X}, \mathbf{A}) \log \frac{q_\eta(\mathbf{Z}|\mathbf{X}, \mathbf{A})}{p_\theta(\mathbf{Z}|\mathbf{X}, \mathbf{A})} d\mathbf{Z} \geq 0. \quad (4.31)$$

On utilise le théorème de Bayes :

$$\int_{\mathbb{R}^{n \times q}} q_\eta(\mathbf{Z}|\mathbf{X}, \mathbf{A}) \log \frac{q_\eta(\mathbf{Z}|\mathbf{X}, \mathbf{A})p_\theta(\mathbf{X}, \mathbf{A})}{p(\mathbf{Z})p_\theta(\mathbf{X}, \mathbf{A}|\mathbf{Z})} d\mathbf{Z} \geq 0. \quad (4.32)$$

Ce qui après développement donne :

$$\begin{aligned} & \int_{\mathbb{R}^{n \times q}} q_\eta(\mathbf{Z}|\mathbf{X}, \mathbf{A}) \log \frac{q_\eta(\mathbf{Z}|\mathbf{X}, \mathbf{A})}{p(\mathbf{Z})} d\mathbf{Z} \\ & - \int_{\mathbb{R}^{n \times q}} q_\eta(\mathbf{Z}|\mathbf{X}, \mathbf{A}) \log p_\theta(\mathbf{X}, \mathbf{A}|\mathbf{Z}) d\mathbf{Z} \geq -\log p_\theta(\mathbf{X}, \mathbf{A}). \end{aligned} \quad (4.33)$$

Soit :

$$\log p(\mathbf{X}, \mathbf{A}) \geq \mathbb{E}_{\mathbf{Z} \sim q_\eta(\mathbf{Z}|\mathbf{X}, \mathbf{A})} \log p_\theta(\mathbf{X}, \mathbf{A}|\mathbf{Z}) - D_{\mathbf{KL}}(q_\eta(\mathbf{Z}|\mathbf{X}, \mathbf{A})\|p(\mathbf{Z})). \quad (4.34)$$

Maximiser la borne inférieure de cette inégalité, c'est en fait minimiser la divergence de Kullback-Leibler entre le véritable encodeur et l'encodeur paramétrique. L'*evidence lower bound* (4.34) de notre problème peut également s'écrire de la manière suivante :

$$\begin{aligned} \mathcal{L}_{ELBO}^{[1, n]} &= \sum_{i \in [1, n]} \sum_{j \in [1, n]} \mathbb{E}_{\substack{\mathbf{z} \sim q_\eta(\mathbf{z}|\mathcal{V}_p(\mathbf{x}^i), \mathbf{A}) \\ \mathbf{z}' \sim q_\eta(\mathbf{z}'|\mathcal{V}_p(\mathbf{x}^j), \mathbf{A})}} \log p_E(\mathbf{A}_{i, j}|\mathbf{z}, \mathbf{z}') \\ &+ \sum_{i \in [1, n]} \sum_{m=1}^M \mathbb{E}_{\mathbf{z} \sim q_\eta(\mathbf{z}|\mathcal{V}_p(\mathbf{x}^i), \mathbf{A})} \log p_{\theta_m}(\mathbf{x}_m^i|\mathbf{z}) \\ &- \sum_{i \in [1, n]} D_{\mathbf{KL}}(q_\eta(\mathbf{z}|\mathcal{V}_p(\mathbf{x}^i), \mathbf{A})\|p(\mathbf{z})). \end{aligned} \quad (4.35)$$

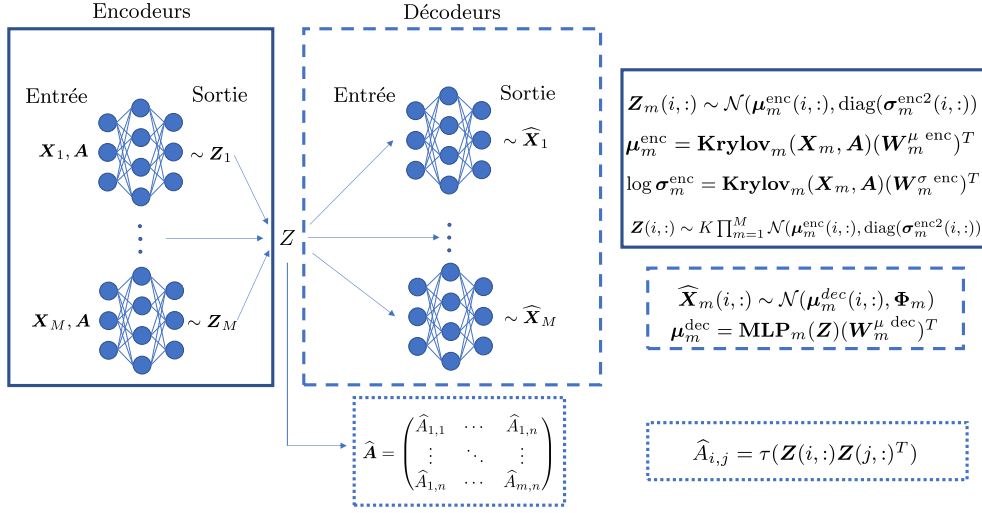


FIGURE 4.4 – **Représentation de MVGCCA.** Toutes les vues sont encodées dans leur propres espaces latents  $Z_m \in \mathbb{R}^{q_m}$  à l'aide du graphe commun. Elles sont ensuite *fusionnées* en une vue commune  $z$  pour former la représentation commune des vues. Finalement  $z$  est décodé pour reconstruire toutes les vues et le graphe commun.

Dans cette formule, on reconnaît dans l'ordre, le terme qui assure la reconstruction du graphe, le terme qui permet la reconstruction des vues et le dernier terme qui est une régularisation qui agit de manière analogue à son équivalent dans l'Equation (4.20).

**Remarque 19 (Encodeurs paramétriques.)** Les formes choisies pour chaque  $q_{\eta m}$  implique que  $q_{\eta}$  (Eq. (4.28)) n'est pas une distribution correctement normalisée, toutefois il existe une constante  $K$  telle que  $Kq_{\eta}$  est une distribution. Si le calcul qui précède est effectué avec  $Kq_{\eta}$  en lieu et place de  $q_{\eta}$ , il aboutirait au même résultat à une constante additive (et une constante multiplicative) près. Par souci de simplicité, nous continuerons à travailler avec  $q_{\eta}$  comme si c'était une distribution.

#### 4.4.4 Entraînements et inférences

**Conditions d'entraînement.** L'ensemble des paramètres du modèle à déterminer sont les poids  $W_m^{\mu \text{ dec}} \in \mathbb{R}^{q_m \times q}$ ,  $W_m^{\sigma \text{ enc}} \in \mathbb{R}^{q \times q_m}$ ,  $W_m^{\mu \text{ enc}} \in \mathbb{R}^{q \times q_m}$  et les paramètres non explicités<sup>15</sup> des perceptrons multicouches des décodeurs et **Krylov** des encodeurs. Pour déterminer ces paramètres, on maximise donc la fonction objectif (4.35) qui permet de maximiser indirectement la log-vraisemblance. Les deux premiers termes de cette fonction objectif sont approximables à l'aide d'un échantillonnage à la Monte-Carlo tandis que le dernier terme a une forme explicite. Les détails techniques sur le calcul en pratique de cette équation peuvent être trouvés en Annexe A.3.2. Comme on peut le remarquer, cette fonction n'est pas séparable en une somme de termes ne dépendant que d'un élément  $i$ , par contre on peut la découper en sous-ensemble d'éléments

15. Dont les caractéristiques sont données en Section (expérimentale) 4.5.

du jeu de données ce qui permet de l'entraîner par *batch*. On procède donc à une partition aléatoire  $\{B_1, \dots, B_k, \dots, B_{n_b}\}$  des indices des éléments  $\llbracket 1, n \rrbracket$ . On effectue ensuite  $n_b$  itérations de l'algorithme de descente (pour maximiser) successivement sur chaque ELBO partielle  $\mathcal{L}_{ELBO}^{B_k}$ . Après  $n_b$  itération, la méthode de descente a vu toutes les données (on a fait une *epoch*). On effectue un certains nombres d'*epochs* (avec des partitions différentes des indices) pour réaliser l'optimisation. Cette procédure d'entraînement permet d'assurer une complexité linéaire de la méthode en la taille des données.

**Inférences.** Lorsque l'entraînement est achevé, les encodeurs permettent de déterminer les représentations de chacun des éléments  $i$ . Par analogie avec l'Equation (4.19) la représentation de  $i$  est choisie comme étant l'espérance de son encodeur  $\mathbb{E}_{\mathbf{z} \sim q_\eta(\mathbf{z}|\mathcal{V}_p(\mathbf{x}^i), \mathbf{A})}$  c'est-à-dire le vecteur de  $\mathbb{R}^q$  :

$$\mathbb{E}_{\mathbf{z} \sim q_\eta(\mathbf{z}|\mathcal{V}_p(\mathbf{x}^i), \mathbf{A})} = \left[ \sum_{m=1}^M \frac{\boldsymbol{\mu}_m^{\text{enc}}(i, k)}{\boldsymbol{\sigma}_m^{\text{enc}2}(i, k)} / \sum_{m=1}^M \frac{1}{\boldsymbol{\sigma}_m^{\text{enc}2}(i, k)} \right]_{k=1}^q. \quad (4.36)$$

La démonstration de ce résultat peut être trouvée en annexe A.3.3. En réalité, le modèle créé pour chaque vue  $m$  d'un élément  $i$ , une représentation latente :

$$\mathbb{E}_{\mathbf{z} \sim q_{\eta_m}(\mathbf{z}|\mathcal{V}_p(\mathbf{x}_m^i), \mathbf{A})} = \boldsymbol{\mu}_m^{\text{enc}}(i, \cdot) \quad (4.37)$$

On peut constater que la représentation d'un élément est en fait le barycentre des représentations de chacune de ses vues, comme on va le voir c'est cette propriété qui est au cœur de la robustesse de notre modèle.

#### 4.4.5 Robustesse et *views dropout*

**Robustesse.** La formule (4.36) est directement issue du choix de  $q_\eta$  comme un produit de loi normale multivariée avec une matrice de covariance diagonale, c'est donc ce choix qui est à l'origine de la propriété de robustesse évoquée ci-dessus. Dans les hypothèses du modèle, on considère que chaque vue d'un élément  $i$  est issue d'une variable  $\mathbf{z}$  (qui est un proxy mathématique pour modéliser la source des vues) que l'on recherche. Pour chacune de ses vues, on a défini une distribution  $q_{\eta_m}(\mathbf{z}|\mathcal{V}_p(\mathbf{x}_m^i), \mathbf{A})$  qui donne la probabilité que la variable latente  $\mathbf{z}$  soit la source de cette vue  $m$ . D'autre part,  $q_\eta(\mathbf{z}|\mathcal{V}_p(\mathbf{x}^i), \mathbf{A}) = \prod_{m=1}^M q_{\eta_m}(\mathbf{z}|\mathcal{V}_p(\mathbf{x}_m^i), \mathbf{A})$  est la probabilité que  $\mathbf{z}$  soit à l'origine de toutes les vues de  $i$  (donc en quelque sorte de  $i$ ). Donc plus une variable latente  $\mathbf{z}$  a une probabilité élevée d'avoir généré les différentes vues de  $i$ , plus  $\mathbf{z}$  est potentiellement à la source de cet élément  $i$ . Mais comme on peut le voir dans l'Equation (4.36) les vues ne contribuent pas nécessairement de manière égale à ces probabilités. La représentation de  $i$  est un barycentre des représentants de chacune de ses vues pondérées par la précision des différentes distributions encodant les vues. Les coefficients de précisions sont les inverses des coefficients diagonaux des matrices de covariance de ces distributions. L'interprétation qu'on

peut faire de cette équation, c'est que chaque vue donne une information sur la variable latente à l'origine de l'élément  $i$ , cette information est prise en compte par ce barycentre. Par contre, toutes les vues ne donnent pas la même quantité d'information sur la source et c'est ce que quantifie la précision. Si une vue est très bruitée ou peu informative sur la nature de la source, on s'attend à une précision faible, limitant ainsi le poids de cette vue dans le calcul du barycentre. A contrario, lorsqu'une vue est très informative alors elle sera très contributive au barycentre. Les différentes contributions sont évaluées composante par composante. Cette propriété est très importante car elle permet au modèle de se passer de certaines vues lorsqu'elles sont manquantes. En effet, on peut calculer le barycentre avec les seules vues disponibles. Si la somme des précisions des vues disponibles est suffisamment grande devant celles des vues manquantes, alors on peut approximer correctement la représentation de cet élément. Cette habileté de MVGCCA à se passer de certaines vues est confirmée expérimentalement dans les expériences en Section 4.5.5, 4.5.5.b et 4.5.5.c.

Cette propriété de robustesse est unique parmi les méthodes liées à la CCA. Les deux autres modèles variationnels CCA présentés en Section 4.3.4 ne satisfont pas à cette propriété. VCCA et VCCA-(p) la méthode de [Wang et al. 2016] est paramétrée de telle sorte à n'utiliser qu'une seule des deux vues<sup>16</sup> pendant l'encodage, cela rend la méthode non robuste à l'absence de cette vue. Le modèle plus récent VPCCA [Karami and Schuurmans 2020] lui peut s'accommoder de la présence d'un nombre limité de vues au moment de l'inférence, mais il nécessite que ces vues existent pour tous les éléments dont ils souhaitent déterminer une représentation ce qui le rend non robuste à des situations réelles de vues manquantes (voir Fig. (4.5)). Notre modèle exploite le maximum d'information disponible pour chaque élément du jeu de données.

**Views dropout** Pour renforcer la capacité de robustesse de notre modèle, on introduit une nouvelle pratique qu'on appellera le *views dropout*. Cette pratique consiste de manière aléatoire durant l'entraînement par *batch* à ignorer certaines vues durant l'encodage, ce qui concrètement signifie qu'au lieu d'échantillonner selon (dans (4.35)) :

$$z \sim q_\eta(z|\mathcal{V}_p(\mathbf{x}^i), \mathbf{A}) = \prod_{m=1}^M q_{\eta_m}(z|\mathcal{V}_p(\mathbf{x}_m^i), \mathbf{A})$$

en notant  $I_k$  l'ensemble des vues choisies comme disponibles, on échantillonnera selon :

$$z \sim q_{\eta_{I_k}}(z|\mathcal{V}_p(\mathbf{x}^i), \mathbf{A}) = \prod_{m \in I_k} q_{\eta_m}(z|\mathcal{V}_p(\mathbf{x}_m^i), \mathbf{A}).$$

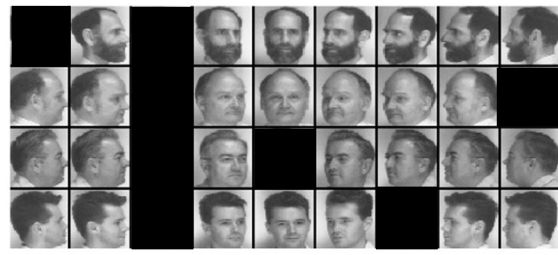
Tout en demandant au modèle d'être en capacité de les régénérer, ce qui donne la fonctionnelle suivante pour un *batch*  $B_k$  (voir Annexe A.3.2 pour la forme explicite de cette équation) :

---

16. Ils proposent une variation de leur modèle dans lequel les deux vues sont utilisées pour conditionner l'encodeur, mais la conclusion quant à la robustesse est la même.

**Premier scénario de vues manquantes**

*Certaines vues sont entièrement disponibles*

**Deuxième scénario de vues manquantes**

*Les éléments peuvent avoir des vues manquantes différentes*

FIGURE 4.5 – *Jeu de données avec des vues manquantes.* **A gauche**, on peut voir un jeu de données dans lequel certaines vues sont manquantes, mais d'autres sont disponibles pour tous les éléments. Les modèles variationnels comme VCCA (limité à des jeux de données comportant 2 vues) et VPCCA peuvent traiter ce type de données. MVGCCA est évalué selon ce scénario en Section 4.5.5.b. **A droite**, on peut voir un jeu de données avec de nombreux éléments corrompus, tous les éléments ont des vues manquantes et elles ne sont pas nécessairement les mêmes. Ce type de jeu de données est une représentation bien plus réaliste du type de corruption qui peut être trouvé dans un jeu de données réel. Notre modèle peut traiter ce type de dégradation à la fois pendant l'entraînement et pendant la phase d'inférence. MVGCCA exploite le maximum d'information disponible pour chacun des éléments du jeu de données. MVGCCA est évalué selon ce scénario en Section 4.5.5.c.

$$\begin{aligned}
\mathcal{L}_{ELBO}^{B_k, \text{vdrop}} &= \sum_{i \in B_k} \sum_{j \in B_k} \mathbb{E}_{\substack{\mathbf{z} \sim q_{\eta_{I_k}}(\mathbf{z} | \mathcal{V}_p(\mathbf{x}^i), \mathbf{A}) \\ \mathbf{z}' \sim q_{\eta_{I_k}}(\mathbf{z}' | \mathcal{V}_p(\mathbf{x}^j), \mathbf{A})}} \log p_E(\mathbf{A}_{i,j} | \mathbf{z}, \mathbf{z}') \\
&+ \sum_{i \in B_k} \sum_{m=1}^M \mathbb{E}_{\mathbf{z} \sim q_{\eta_{I_k}}(\mathbf{z} | \mathcal{V}_p(\mathbf{x}^i), \mathbf{A})} \log p_{\theta_m}(\mathbf{x}_m^i | \mathbf{z}) \\
&- \sum_{i \in B_k} D_{\text{KL}}(q_{\eta}(\mathbf{z} | \mathcal{V}_p(\mathbf{x}^i), \mathbf{A}) \| p(\mathbf{z})).
\end{aligned} \tag{4.38}$$

Les expériences ci-dessous présentent en détail la manière dont nous avons évalué la robustesse de notre modèle et notamment comment le *views dropout* permet d'augmenter cette dernière.

**Remarque 20 (Distribution non normalisée.)** *Lorsqu'on échantillonne selon  $q_{\eta_{I_k}}$ , on échantillonne en réalité selon la version normée de cette pseudo-distribution.*

## 4.5 Expériences

Dans cette section, nous présentons les expériences illustrant les propriétés de la méthode et démontrant son efficacité. Nos expériences ont été faites sur 3 jeux de données couramment utilisés dans la littérature de la CCA.

### 4.5.1 Jeux de données

**UCI *handwritten digits***<sup>17</sup> est un jeu de données multivues de  $n = 2000$  éléments dont les sources des vues sont des images<sup>18</sup> représentant un chiffre écrit à la main. Chaque élément a donc une étiquette comprise entre 0 et 9, et chaque étiquette est portée par 200 éléments. Ces éléments possèdent 6 vues de dimension :  $q_1 = 76$ ,  $q_2 = 216$ ,  $q_3 = 64$ ,  $q_4 = 240$ ,  $q_5 = 47$  et  $q_6 = 6$  qui correspondent respectivement aux transformations suivantes de l'image originale : coefficients de Fourier de la forme des caractères  $\mathbf{X}_1 \in \mathbb{R}^{q_1 \times n}$ , profil de corrélations  $\mathbf{X}_2 \in \mathbb{R}^{q_2 \times n}$ , coefficients de Karhun-Loeve  $\mathbf{X}_3 \in \mathbb{R}^{q_3 \times n}$ , échantillonnage de l'image originale de 240 pixels (les pixels de l'échantillonnage sont des moyennes des pixels de l'image originale sur une fenêtre de taille  $2 \times 3$ )  $\mathbf{X}_4 \in \mathbb{R}^{q_4 \times n}$ ; modes de Zernike  $\mathbf{X}_5 \in \mathbb{R}^{q_5 \times n}$  et 6 caractéristiques morphologiques  $\mathbf{X}_6 \in \mathbb{R}^{q_6 \times n}$ . Des tâches de *clustering*, classification et de reconstruction ont été effectuées sur cet ensemble de données (*uci10*) et sur une version réduite (*uci7*) où les éléments appartenant aux classes 0, 5 et 6 ont été retirés<sup>19</sup>. Ce jeu de données ne possède pas de structure pré-établie, un graphe structurant les éléments est construit préalablement suivant la méthode proposée par [Chen et al. 2019] (cette procédure est décrite en annexe A.3.4). Cet ensemble de données est considéré pour pouvoir se comparer avec GMCCA Chen et al. [2019] qui était jusqu'alors la seule méthode de la littérature traitant des données multivues (avec  $M > 2$ ) structurées.

**Twitter friend recommendation**<sup>20</sup> est un jeu de données basé sur des messages postés sur twitter, il a été proposé par Benton et al. [2016]. Les éléments de ce jeu de données ont des vues qui sont des représentations des messages d'utilisateurs de la plateforme. Il a été construit en prenant 1% des tweets publiquement disponibles en avril 2015. Parmi eux, tous les tweets écrits soit dans une langue différente de l'anglais, soit par des utilisateurs qui n'ont pas posté entre janvier et février 2015, sont exclus.<sup>21</sup> Finalement, les 100 derniers tweets des utilisateurs ( $n = 102327$ ) restants sont conservés. Ces messages sont intégrés de différentes façons dans un espace euclidien de grande dimension, sur lesquelles une PCA est appliquée. Les utilisateurs se retrouvent ainsi avec 6 vues de dimension 1000 représentant chacune une caractéristique de leurs tweets ou du réseau dont ils font partie : *EgoTweets*, *MentionTweets*, *FriendTweets*, *FollowerTweets*, *FriendNetwork* et *FollowerNetwork*. Une tâche de recommandation d'ami est effectuée sur ce jeu de données, qui sera explicité dans la partie expérimentale dédiée en Section 4.5.4. Le graphe de twitter n'est pas fourni dans ce jeu de données, mais certaines vues donnent des informations sur la structure du réseau, ces informations sont exploitées pour construire un

17. [archive.ics.uci.edu/ml/datasets/Multiple+Features](http://archive.ics.uci.edu/ml/datasets/Multiple+Features)

18. Ces images ont été perdues aujourd'hui.

19. On considère ce jeu de données réduit, pour pouvoir se comparer à GMCCA.

20. <http://www.cs.jhu.edu/~mdredze/data/>

21. Il y a d'autres critères d'exclusions mineurs qui peuvent être retrouvés dans l'article de Benton et al. [2016].





FIGURE 4.6 – *Two-view MNIST*. Trois éléments du jeu de données two-view MNIST. Pour chaque chiffre (1, 5 et 8) on a : **à gauche** l'image originale, **au milieu** la première vue (pivotée), **à droite** la seconde vue (bruitée).

graphe qu'on détermine en suivant la procédure proposée par [Chen et al. \[2019\]](#) qui est décrite en annexe [A.3.4](#). Encore une fois, ce jeu de données a été choisi dans le but de se comparer avec GMCCA.

*Two-view MNIST* est un jeu de données multivues proposé par [Wang et al. \[2015\]](#) comportant 2 vues construit à partir de MNIST, qui est un jeu de données constitué d'images  $28 \times 28$  de chiffres écrits à la main. Ce jeu de données est composé d'un ensemble d'entraînement de 50000 images, de validation de 10000 images et de test d'autant d'images. On construit le jeu de données de la manière suivante : la première vue est obtenue après avoir procédé à une rotation des 70 000 images de MNIST. Chaque angle de rotation est choisi aléatoirement selon la distribution uniforme  $\mathcal{U}(-\frac{\pi}{4}, \frac{\pi}{4})$ . Quant à la seconde vue, pour chaque première vue on choisit aléatoirement une image ayant la même étiquette (dans l'ensemble des 70 000 images) à laquelle on ajoute un bruit uniforme  $\mathcal{U}(0, 1)$ . Les pixels de cette seconde vue sont ensuite tronqués pour conserver une valeur comprise entre 0 et 1. On associe à ce jeu de données un graphe des étiquettes : chaque élément a une chance sur 10 (resp. une chance sur 1000) d'être connecté aux éléments qui ont une étiquette identique (resp. une étiquette différente). Ainsi, chaque élément est connecté à environ 10% d'éléments qui ont une étiquette différente. Le graphe apporte donc une information supplémentaire à propos des étiquettes mais il est bruité pour ne pas faciliter trop la tâche à notre modèle. Cet ensemble de données a été choisi car il permet de se comparer à de nombreuses méthodes de la littérature des CCA, notamment les méthodes variationnelles [Wang et al. \[2016\]](#) ; [Karami and Schuurmans \[2020\]](#). Une illustration de ce modèle peut-être trouvée Figure en [4.6](#).

#### 4.5.2 Paramètres

Toutes les architectures et hyperparamètres spécifiés ici sont les paramètres par défaut des expériences<sup>22</sup>, si rien n'est indiqué dans le texte. Chaque vue est centrée et normalisée. Et pour chaque jeu de données, la matrice d'adjacence est normée par sa valeur maximale.

**Décodeurs :** Chacun des perceptrons multicouches  $\text{MLP}_m$  qui paramètrent les décodeurs ont 4 couches cachées et sont suivis par la non linéarité : ReLU. Les matrices de covariance des décodeurs  $\Psi_m$  sont choisies comme des matrices scalaires i.e.,  $\Psi_m = ((\sigma_m + 10^{-6})^2)I_{q_m}$ . Ce choix permet de réduire la complexité du modèle, et d'améliorer la stabilité des performances

22. Code disponible : <https://github.com/Yacnnn/MVGCCA>.

de l'algorithme sans les dégrader.

**Encodeurs :** Les GCN **Krylov** qui calculent la moyenne et la variance des encodeurs  $q_{\eta m}$  ont également 4 couches cachées et une profondeur d'exploration des nœuds de 4, i.e.,  $p = 4$ .

**Généralités :** La taille des *batch* est fixée à 512. Un taux de dropout de 0.5 est systématiquement appliqué à toutes les couches intermédiaires des réseaux de neurones. *Adam* est l'algorithme utilisé pour la descente de gradient. Cet algorithme met à jour automatiquement le *learning rate* pour accélérer la convergence. Toutefois, on a constaté expérimentalement que mettre à jour le *learning rate* au début de chaque *epoch* selon la loi  $l_r \times 1.1^{-50 \frac{e}{E}}$  permet d'aider la convergence du modèle. Ici  $l_r$  est le *learning rate* initial  $1 \cdot 10^{-3}$ ,  $e$  est l'indice de *epoch* actuelle + 1, et  $E$  le nombre d'*epochs* maximal qui est fixé à 600. Des informations supplémentaires sur des paramètres du modèle sont données en Annexe A.3.2.

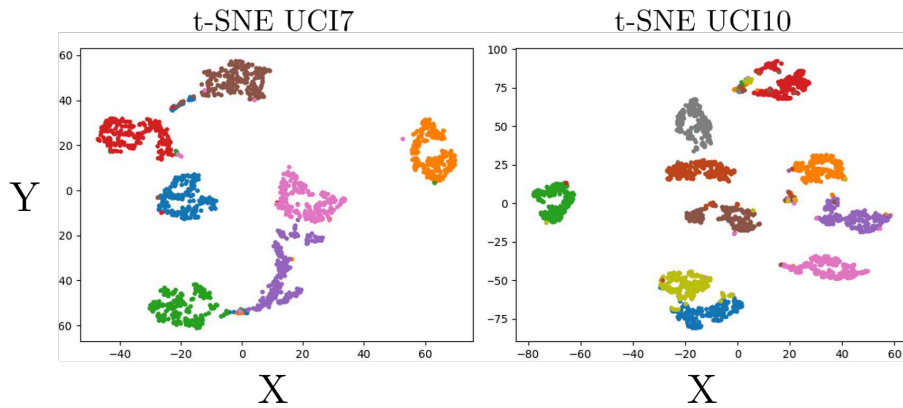


FIGURE 4.7 – Visualisation en 2D de l'espace latent ( $q = 3$ ) via l'application de la t-SNE [Van der Maaten and Hinton 2008]. pour les jeux de données uci7 et uci10. Chaque couleur représente une classe différente.

### 4.5.3 Classification et *clustering*

Dans cette partie, on évalue les tâches de classification et de *clustering*. La première expérience est une comparaison entre notre méthode MVGCCA et GMCCA qui sont les deux seules méthodes qui peuvent tenir compte d'une structure de graphe. La seconde expérience est une comparaison avec d'autres méthodes qui ne tiennent pas compte d'une tel structure : DeepCCA, KernelCA, VCCA etc.

#### 4.5.3.a UCI

**Paramètres.** Notre modèle est entraîné par *batch* sur 100% des jeux de données *uci7* et *uci10*, selon les modalités décrites en Section 4.4.4. La dimension de l'espace latent est fixée à  $q = 3$ . Dans cette expérience, une partie des paramètres est déterminée à l'aide d'une grille de recherche : le *learning rate*  $\{1 \cdot 10^{-3}, 1 \cdot 10^{-4}\}$ , le nombre de couches cachées  $\{3, 4\}$  et la dimension de ces couches  $\{512, 1024\}$ . Nous avons aussi soumis à cette grille s'il fallait où non appliquer

un *learning rate* décroissant et s'il fallait ou non utiliser une matrice de covariance (pour les décodeurs) scalaire. Pour chaque combinaison de ces paramètres, la méthode a été entraînée 3 fois, et toutes les 100 *epochs* les représentations latentes ont été sauvegardées, ce qui permet de faire du nombre d'*epochs* dans le processus d'optimisation un autre hyperparamètre à déterminer. Une visualisation des espaces latents produits par notre algorithme peut être trouvée en Figure 4.7.

**Évaluation.** Après l'entraînement, nos représentations latentes sont découpées en deux : 9/10-ième sert d'ensemble d'entraînement et 1/10-ième de test. Le classifieur utilisé est une machine à vecteur de support avec un noyau gaussien (SVM-RBF), et la métrique des performances est l'*accuracy*. Ce classifieur possède ses propres paramètres ( $C$  et  $\gamma$ ) qui sont aussi soumis à sélection au même titre que les paramètres de MVGCCA. L'ensemble des paramètres (paramètres du modèle + paramètre du classifieur + nombre d'*epochs*) sont ajustés à l'aide d'une validation croisée à 5 blocs sur l'ensemble d'entraînement. Pour la meilleure combinaison de paramètres on entraîne finalement un modèle sur l'ensemble d'entraînement puis on l'évalue sur l'ensemble de test. On évalue également deux méthodes de clustering que sont le *k-means* et le *spectral clustering* sur l'ensemble de toutes les données. Cette procédure est moyennée sur 3 exécutions de l'algorithme. A cause de la petite taille de ce jeu de données, les résultats de ces expériences sont très dépendant du découpage initial entre ensemble d'entraînement et ensemble de test. Pour gommer cet effet, cette expérience est reproduite 100 fois avec des découpages entraînement/test différents. Les résultats de cette expérience peuvent être retrouvés dans la Table 4.2. Dans cette table, on retrouve également les résultats de ces expériences pour la PCA (appliquée à la concaténation de l'ensemble des vues), MCCA 4.13, et GMCCA 4.15. On peut voir que MVGCCA est une méthode compétitive à la fois dans la tâche de classification et de clustering. Elle obtient notamment de meilleurs résultats sur la version la plus complexe du jeu de données (*uci10*) que GMCCA qui est la seule autre méthode prenant en compte des données multivues structurées.

#### 4.5.3.b Two-view MNIST

**Paramètres.** Comme indiqué plus haut, ce jeu de données possède un ensemble d'entraînement, de validation et de test. Contrairement à l'expérience précédente où la méthode a été entraînée sur l'ensemble des données, ici on va entraîner le modèle uniquement sur l'ensemble d'entraînement. Les représentations latentes de l'ensemble de validation et de test seront inférées après l'entraînement. Par ailleurs dans cette expérience on soumet à la validation, via une grille de recherche, un nombre plus restreint de paramètres qui sont : la dimension de l'espace latent  $q = \{30, 60\}$  et l'utilisation de matrices de covariances scalaires pour les décodeurs et le nombre d'*epochs*. Le nombre d'*epochs* maximal est fixé à 200 et l'espace latent est sauvegardé toutes les 10 *epochs* à partir de la 50-ième.

**Évaluation.** Dans le but d'avoir une comparaison juste avec les deux autres méthodes d'inférence variationnelle, on se place dans une situation où seule la première vue est disponible pour le calcul de la représentation des éléments. Dans notre modèle, cela équivaut à utiliser

Jeu de données	uci7			uci10			Recommandation		
Métrique	Acc.	ARI	ARI2	Acc.	ARI	ARI2	Prec.	Recall	Mrr
PCA	0.84	0.55	-	0.69	0.42	-	0.1511	0.0795	0.3450
GPCA	0.93	0.71	0.77	0.87	0.63	0.62	0.1578	0.0831	0.3649
MCCA	0.86	0.66	-	0.76	0.59	-	0.0815	0.0429	0.2225
GMCCA	<b>0.95</b>	<b>0.83</b>	0.84	0.90	0.69	0.71	<b>0.2290</b>	<b>0.1206</b>	<b>0.4471</b>
MVGCCA	<b>0.95</b>	0.82	<b>0.85</b>	<b>0.94</b>	<b>0.74</b>	<b>0.77</b>	0.1753	0.0583	0.4432
Jeu de données							Recommandation (Large)		
Métrique							Prec.	Recall	Mrr
MVGCCA							0.1745	0.0960	0.4301

TABLE 4.2 – *Résultats des expériences des tâches d’apprentissage effectuées sur UCI et Tfr.* Acc. est une abréviation pour désigner l’*accuracy*; ARI désigne l’*adjusted rand index* c’est une mesure de la qualité du *clustering*; ARI1 (resp. ARI2) désigne cette mesure après avoir effectué un *clustering* avec la méthode des *K*-moyenne (resp. du *spectral clustering*). Pour la tâche de recommandation, les abréviations Prec. et Mrr désignent respectivement la *precision* et le *mean reciprocal rank*. Pour uci7 et uci10 toutes les méthodes ont été recodées et les expériences refaites, notamment car ces méthodes n’avaient pas été évaluées sur une SVM. Pour la recommandation, nous avons repris les résultats de [Chen et al. 2019].

$\mu_1^i$  (voir Eq. (4.37)) comme représentant de l’élément. Cette représentation est ensuite utilisée pour réaliser la tâche de classification. Le classifieur utilisé est une machine à vecteur de support à noyau linéaire (SVM-linéaire) et la métrique d’évaluation est de nouveau l’*accuracy*. On n’utilise pas un noyau gaussien, car ce jeu de données est relativement simple et donne des résultats proches de 100% pour la plupart des méthodes. L’intérêt de la SVM-linéaire est que c’est une méthode relativement simple, un bon taux de classification avec cette méthode est le signe que les représentations sont de bonnes qualités i.e., que de simples équations linéaires permettent de les séparer. Quoiqu’il en soit, la SVM-linéaire à un paramètre ( $C$ ) qui est lui aussi soumis à validation. Les combinaisons de paramètres possibles sont évaluées sur l’ensemble de validation après entraînement sur l’ensemble d’entraînement. La meilleure combinaison de paramètres est ensuite entraînée sur l’ensemble d’entraînement et de validation avant d’être finalement évaluée sur l’ensemble de test. On moyenne nos expériences sur 3 exécutions de ces dernières.

Comme on peut le voir sur la Table 4.3, notre méthode est encore compétitive par rapport aux meilleures méthodes de la littérature, notamment grâce à l’information supplémentaire apportée par la structure de graphe lors de l’entraînement directement dans l’espace latent. Cette expérience illustre comment la CCA peut bénéficier du fait de tenir compte des relations structurelles dans l’espace latent.

#### 4.5.4 Twitter friend recommandation

Dans cette Section, on décrit l’expérience de recommandation qui ne concerne que le jeu de données de twitter.

Jeu de données	Two-view MNIST
Métrique	Acc.
CCA linéaire	0.804
SpliAE	0.881
Kernel CCA	0.949
Deep CCA	0.971
DCCAE	0.978
VCCA	0.970
VCCA-(p)	0.976
VPCCA	0.981
MVGCA	<b>0.985</b>

TABLE 4.3 – *Résultats des expériences de classification avec une SVM linéaire.* Les résultats des méthodes de la littérature sont issues des articles de Wang et al. [2015] et Karami and Schuurmans [2020]. Ces résultats comprennent ceux de la CCA linéaire Hotelling [1936]; de certaines méthodes non linéaires : Kernel CCA Lopez-Paz et al. [2014], Deep CCA Andrew et al. [2013], Deep CCA autoencoders Wang et al. [2015]; et de modèle variationnel : Variational CCA (-private) Wang et al. [2016] et Variational Probabilistic CCA Karami and Schuurmans [2020]. Pour VCCA, VPCCA et MVGCCA, toutes les vues sont disponibles pendant l’entraînement par contre lors de la phase d’inférence des représentations latentes, seule la première vue est disponible. MVGCCA and VPCCA sont les seuls modèles de cette liste qui peuvent opérer sur des jeux de données avec plus de 2 vues.

**Paramètres.** On entraîne sur 100% de ce jeu de données. Les hyperparamètres de ce jeu de données sont fixés : le *learning rate* à  $1.0 \cdot 10^{-4}$ , le nombre de couches cachées à 4 et leur taille à 1024. Cet ensemble de données est très large, il comporte plus de 100 000 éléments, ce qui rend donc son traitement difficile pour beaucoup de méthodes CCA, c’est pourquoi dans la littérature seul un échantillon aléatoire d’utilisateurs est traité [Benton et al. 2016; Chen et al. 2019]. Dans un souci de comparabilité, nous ferons de même; nous travaillerons aussi sur des échantillons de 2506 utilisateurs. Toutefois, comme l’une des revendications de ce chapitre est la scalabilité de notre méthode, nous traiterons aussi un échantillonnage plus grand de ce jeu de données pour l’illustrer.

**Évaluation.** La procédure d’évaluation des tâches de recommandations et ses paramètres sont issus de Benton et al. [2016]; Chen et al. [2019]. On choisit les 20 utilisateurs les plus suivis (de tout le jeu de données). Pour chacun de ces utilisateurs, on sélectionne 10 utilisateurs (dans l’échantillon) qui les suivent et on fait la moyenne de leurs représentations calculées par MVGCCA. Cette moyenne sera en quelque sorte la représentation type de l’utilisateur qui suit cette célébrité et servira donc de référence. Ensuite, on calcule la similarité cosinus<sup>23</sup> entre ces profils typiques et les représentations des utilisateurs. Les  $L = 100$  utilisateurs les plus proches

23. La similarité cosinus entre deux vecteurs  $\mathbf{a}$  et  $\mathbf{b}$  dans  $\mathbb{R}^d$  est la quantité  $\frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|_2 \|\mathbf{b}\|_2} \in [-1, 1]$ . Cette grandeur vaut  $-1$  lorsque les vecteurs sont opposés et  $1$  lorsqu’ils sont égaux. Lorsque  $d = 2$ , la similarité cosinus est égale au cosinus de l’angle entre les deux vecteurs, d’où le nom de cette dernière.

de chacune de ces représentations types seront considérés comme devant recevoir une recommandation pour suivre l'artiste en question. S'il s'avère que ces utilisateurs suivent réellement la célébrité, on considère que la recommandation est bonne. On mesure la qualité de la recommandation, à travers 3 métriques : la *precision*<sup>24</sup>, le *recall*<sup>25</sup> et la *mrr*<sup>26</sup> (*mean reciprocal ranking*). Cette expérience est moyennée sur 100 échantillonnages de 2506 utilisateurs.

Les résultats de cette expérience sont consultables Table 4.2. Comme on peut le voir, MVGCCA est comparable aux autres méthodes sauf à GMCCA pour le *recall*. Cependant, cette tâche est tout à fait artificielle puisque seule une portion du jeu de données est traitée. La nature variationnelle de notre méthode fait que plus on lui fournira de données, plus on peut espérer que ses performances soient meilleures. Pour le démontrer en même temps que la scalabilité de notre méthode, nous avons refait les mêmes expériences sur un jeu de données 5 fois plus large, soit avec  $n = 12530$  utilisateurs en considérant cette fois-ci 50 célébrités. La dimension de l'espace latent est fixée à  $q = 10$ . Lors de l'évaluation, le paramètre  $L$  est cette fois-ci fixé à 500 pour avoir une difficulté relativement identique à l'expérience précédente et rendre la comparaison valable. Cette expérience a été répétée 20 fois avec un échantillonnage d'utilisateurs différents. Comme on peut le voir sur la table 4.2, l'augmentation de la quantité de données a permis d'augmenter la qualité du *recall*. Cette capacité à traiter de grand volumes de données, résulte en une meilleure capacité à faire de la recommandation ; en plus de pouvoir envisager la résolution de problème réel. Une comparaison de la croissance du temps de calcul entre MVGCCA et GMCCA sur ce jeu de données peut être trouvée Figure 4.8.

**Remarque 21 (Traitement complet de twitter.)** *Ce qui empêche de traiter la totalité du jeu de données, c'est la manière dont le graphe est créé. Cette procédure est coûteuse (voir Annexe A.3.4) mais elle est spécifique à ces exemples artificiels où le véritable graphe n'est pas disponible.*

#### 4.5.5 Inférences

Dans les trois expériences qui suivent, nous évaluons les capacités d'inférence et de robustesse de notre modèle, nous évaluerons d'une part sa capacité à produire des représentations de qualité malgré l'existence d'éléments avec des vues manquantes et d'autre part sa capacité à régénérer ses vues manquantes. Pour régénérer la vue manquante d'un élément, on utilise simplement l'encodage approximatif  $\mathbf{z}^{\text{approx}}$  de cet élément calculé avec l'Equation (4.36) restreint à l'ensemble des vues disponibles. Ensuite, cette approximation est utilisée par le décodeur de la vue manquante qu'on souhaite régénérer :

24. La *precision* correspond au nombre de recommandations correctes sur le nombre de recommandations faites (i.e  $L$ ).

25. Le *recall* correspond au nombre de recommandations correctes sur le nombre de recommandations correctes maximal possible.

26. Le *mean reciprocal ranking* est un indicateur qui permet de mesurer le rang des recommandations correctes. Etant donné nos  $L$  recommandations (classées par ordre de proximité aux vecteurs de références), en notant  $L_v$  le nombre de recommandations valides et  $r_1, \dots, r_{L_v}$  les rangs de ces recommandations valides dans les  $L$  recommandations. Alors,  $mrr = \frac{1}{L_v} \sum_{i=1}^{L_v} \frac{1}{r_i}$ .

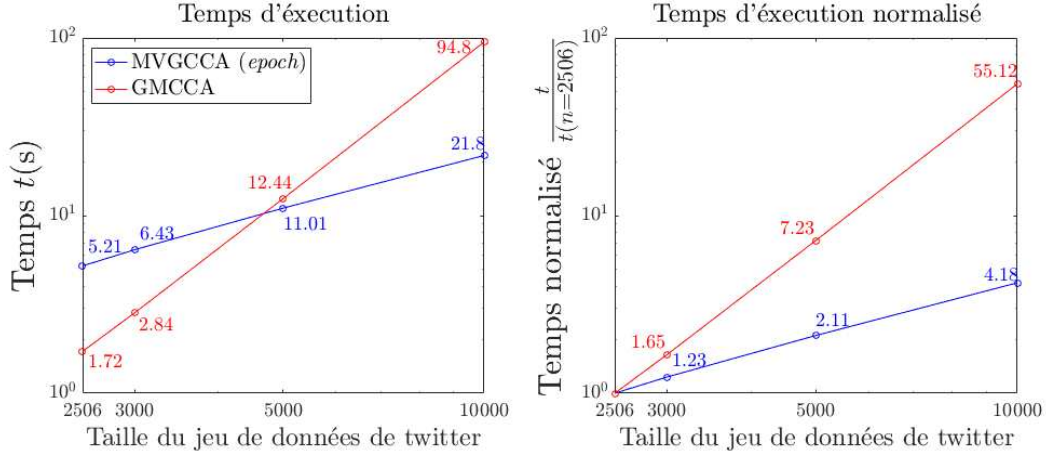


FIGURE 4.8 – *Comparaison expérimentale de la complexité temporelle*. Ces deux figures sont des diagrammes log-log des temps d’exécution des deux méthodes de CCA tenant compte de contraintes structurelles : MVGCCA et GMCCA. On a mesuré le temps pour une exécution de GMCCA et une *epoch* de MVGCCA pour 4 tailles d’échantillon du jeu de données de twitter ( $n = 2506, 3000, 5000, 10000$ ). Sur la figure de **droite**, les temps sont normalisés par rapport aux temps d’exécution respectifs des méthodes pour  $n = 2506$ . On peut voir que le temps d’exécution de GMCCA, croît avec la taille des données, bien plus rapidement que celui de MVGCCA.

$$\mathbf{x}_m^{\text{regenerate}} = \mathbb{E}_{\mathbf{x}_m \sim p_{\theta_m}}(\mathbf{x} | \mathbf{z}^{\text{approx}}) = \text{MLP}_m(\mathbf{z}^{\text{approx}})(\mathbf{W}_m^{\mu \text{dec}})^T. \quad (4.39)$$

#### 4.5.5.a Inférences des vues manquantes

**Paramètres.** Dans cette expérience, on va tester qualitativement la capacité de notre modèle à régénérer des vues manquantes sur *uci7* et *uci10*. Pour ce faire, on se place dans le scénario suivant : on entraîne MVGCCA sur 90% des données où toutes les vues pour tous les éléments sont disponibles. On choisit parmi les 10% restants 10 éléments, auxquels on retire la 4<sup>ème</sup> vue qui correspond à un échantillonnage de l’image originale. De plus, pour tout  $k \in \llbracket 0, 4 \rrbracket$  on retire<sup>27</sup>  $k$  vues (les mêmes pour tous). On utilise ensuite les vues restantes pour régénérer la 4<sup>ème</sup> vue selon le principe expliqué ci-dessus.

**Évaluation.** Les résultats de cette expérience sont visibles Figure 4.9. On peut voir à gauche les résultats de reconstruction pour *uci7* qui sont plutôt de bonne qualité même lorsque très peu de vues sont disponibles, même si on peut toutefois constater des cas problématiques. Ces cas problématiques ont de multiples causes, notamment des erreurs dans le jeu de données : on

27. Les  $k$  vues retirées à chaque fois sont les vues de plus petites dimensions.

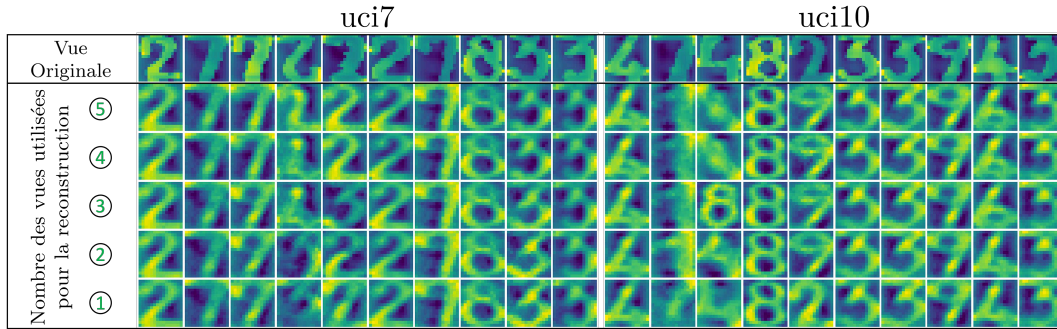


FIGURE 4.9 – *Reconstruction de vues manquantes*. On peut voir ici, les résultats de la reconstruction de la 4<sup>ème</sup> vue d’éléments de uci7 et uci10. La première ligne correspond à la vue originale qui a été retirée. Les vues des lignes  $l > 2$  sont les vues reconstruites à partir de  $7 - l$  vues disponibles. Dans cette expérience, nous n’avons pas usé du *views dropout*, mais il pourrait potentiellement améliorer ces résultats.

peut voir sur la colonne 4 que la vue originale est à l’envers. Ou encore le manque de données : ce jeu de données n’est pas assez grand, un entraînement sur une base plus grande pourrait améliorer les résultats et notamment aider la méthode à faire la distinction entre le 7 et le 9 qui sont confondus comme on peut le voir sur la figure. Avec *uci10*, on peut voir que la méthode a plus de difficultés, c’est attendu puisque c’est un jeu de données plus complexe et n’ayant pas beaucoup plus d’exemples. On peut notamment relever la confusion faite entre le 4 et 5 ou encore entre le 7 et le 1. Quoiqu’il en soit la plupart des reconstructions sont bonnes (au sens où on reconnaît le chiffre associé) même lorsque peu de vues sont disponibles, ce qui valide la capacité de la méthode à régénérer des vues manquantes.

#### 4.5.5.b Robustesse - scénario 1

**Paramètres.** Dans ce scénario, nous allons évaluer quantitativement la capacité de la méthode à s’accommoder de l’existence de vues manquantes selon le scénario 1, c’est-à-dire que seul un nombre limité de vues est disponible (mais elles sont disponibles pour tous). On travaille de nouveau avec *uci7* et *uci10* mais également avec une version étendue dans laquelle on a ajouté une 7<sup>ème</sup> vue qui correspond à un *one-hot encoding* de l’étiquette de l’élément. On entraîne MVGCCA sur 90% des éléments pour chacun de ces jeux de données.

**Évaluation.** Ces jeux de données sont évalués de la manière suivante :

- Pour le jeu de données ”classique” on entraîne une SVM-RBF, qu’on évalue sur l’ensemble de test après qu’un certain nombre de vues aient été retirées.
- Pour la version étendue, on régénère la 7<sup>ème</sup> vue avec un nombre limité de vues, ce qui nous donne directement une estimation de l’étiquette.

Pour un certain nombre de vues disponibles dans l’ensemble de test  $\nu \in \llbracket 1, 5 \rrbracket$ , on considère toutes les combinaisons possibles de vues disponibles pour l’ensemble de test (exemple : Si  $\nu = 2$  l’ensemble des vues disponibles est  $\{(1,2), (1,3), \dots, (5,6)\}$ ), on évalue ensuite chacune de ces



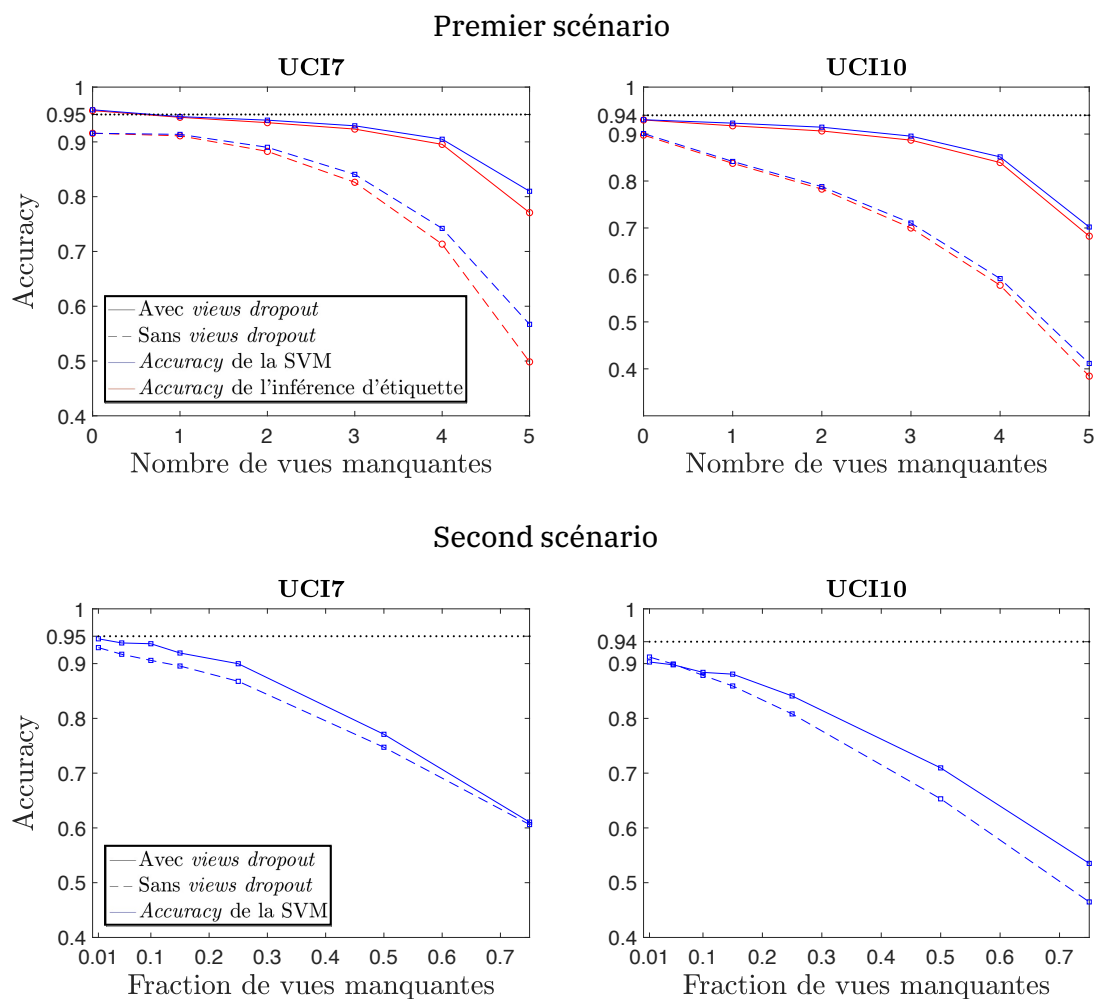


FIGURE 4.10 – *Illustration de la robustesse de MVGCCA*. Voir Section 4.5.5.b, 4.5.5.b et la Figure 4.5 pour des détails sur les deux scénarios. **En haut, le premier scénario :** L' *accuracy* du clustering est comparé pour les deux expériences décrites en Section 4.5.5.b (SVM-RBF et inférence), sans et avec *views dropout*. La ligne noire en pointillée correspond à l'*accuracy* déterminée en Section 4.5.3.a avec toutes les vues. Dans cette expérience, nous n'avons pas fait de recherche de paramètres optimaux, il est donc attendu d'avoir une *accuracy* inférieure lorsque aucune vue n'est manquante. Toutefois, avec le *views dropout* pour *uci7* on obtient un meilleur résultat. **En haut, le second scénario :** Dans ce scénario, la robustesse est évaluée uniquement avec la SVM-RBF *accuracy*. On peut voir un comportement similaire à l'expérience du haut. Plus il y a de vues manquantes, plus les performances décroissent. La procédure du *views dropout* aide quelque peu à améliorer la robustesse mais son effet est bien moindre que dans l'expérience du haut. Globalement, ce scénario semble plus difficile, toutefois MVGCCA est la seule méthode qui puisse le traiter.

possibilités et on en fait la moyenne (à  $\nu$  constant). Ce procédé permet d'évaluer la performance de notre modèle à s'accommoder d'un certain nombre de vues indépendamment de la nature de la vue manquante. Cette expérience est répétée et moyennée 10 fois. Les résultats peuvent être observés Figure 4.10 en ligne pointillée. Comme on peut le voir, le modèle fonctionne même lorsqu'un nombre réduit de vues est disponible. Comme attendu les performances du modèle décroissent avec le nombre de vues manquantes.

Dans la même Figure 4.10, on peut voir les résultats de deux autres expériences en lignes continues. Ce sont les résultats de la même expérience mais avec le *views dropout*. A chaque itération, on a retiré entre 1 et 5 vues aux éléments du *batch* tout en demandant tout de même à la méthode de régénérer ses vues (voir fonctionnelle en Figure 4.38). Comme on peut clairement le constater, ce procédé augmente de beaucoup la robustesse du modèle. L'*accuracy* avec le *view dropout* est bien plus importante que sans en plus de décroître beaucoup moins vite lorsque le nombre de vues manquantes décroît. Ceci valide la pertinence de ce nouveau procédé.

#### 4.5.5.c Robustesse - scénario 2

**Paramètres.** Ce scénario correspond au deuxième de la Figure 4.5, c'est le scénario le plus proche de la réalité. Dans l'expérience précédente, des sous-ensembles limités de vues disponibles étaient disponibles pour générer l'espace latent, mais toutes les instances avaient le même ensemble de vues disponibles. Dans un scénario réel, différentes instances peuvent avoir des vues manquantes différentes. MVGCCA est la seule méthode capable de traiter ce scénario que nous évaluons dans cette partie. De nouveau, on entraîne MVGCCA sur 90% du jeu de données et nous inférons ensuite l'ensemble de test dans lequel nous avons retiré  $r = \{1\%, 5\%, 10\%, 15\%, 25\%, 50\%, 75\%\}$  de vues tout en s'assurant que chaque éléments ait au moins une vue. On a répété cette expérience 10 fois, et on moyenne les résultats. L'expérience est faite sans et avec *views dropout* (on retire cette fois-ci entre 1 et 3 vues à chaque *batch*).

**Évaluation.** Comme on peut le voir Figure 4.10, on a des résultats similaires à ce qu'on a pu voir dans l'expérience qui précède. La méthode est capable de traiter un jeu de données incomplet. Encore une fois, la procédure de *views dropout* est utile mais n'améliore les résultats que très légèrement. Ce comportement est attendu puisque c'est un scénario plus difficile. De plus, le *views dropout* qu'on a introduit retire les mêmes vues à tous les éléments, une piste d'amélioration serait de faire un *views dropout* qui retire des vues différentes aux éléments (comme dans le scénario 2).

**Remarque 22 (Inférences selon le scénario 2.)** *Comme discuté plus tôt, pour calculer une représentation lorsqu'une vue est manquante, il suffit de restreindre la formule (4.36) aux vues disponibles. Ceci est possible sans problème dans le scénario 1 mais pas dans dans le scénario 2. En effet, les quantités  $\mu_m^{enc}(i, :)$  et  $\sigma_m^{enc}(i, :)$  de cette équation dépendent de l'existence de la vue  $m$  dans le voisinage de  $i$ . Or dans le scénario 2, les vues  $m$  des voisins peuvent être manquantes. Pour surmonter cette difficulté dans l'encodeur  $q_{\eta_m}(z|\mathcal{V}_p(\mathbf{x}_m^i), \mathbf{A})$  on remplace simplement  $\mathbf{A}$  par  $\mathbf{A}_m$  qui est la matrice d'adjacence où on a supprimé les liens des éléments qui n'ont pas de vue  $m$ . Ainsi  $\mathcal{V}_p(\mathbf{x}_m^i)$  devient l'ensemble des vues  $m$  dans le voisinage (de profondeur  $p$ ) de l'élément  $i$  selon  $\mathbf{A}_m$  (c'est-à-dire celles qui sont disponibles).*

## 4.6 Conclusion

Dans ce chapitre, on a proposé une nouvelle méthode de CCA, MVGCCA. Cette méthode est non linéaire et peut tenir compte de contraintes structurelles. La méthode se base sur une approche bayésienne de la CCA et une approche variationnelle qui lui permet de passer à l'échelle sur de larges volumes de données. Notre méthode est aussi robuste aux jeux de données corrompus, notamment grâce à sa structure mais aussi grâce à la méthode de robustification nommée *views dropout* que nous avons introduite. La nature probabiliste du modèle nous permet également de pouvoir effectuer des tâches d'inférence et notamment de restaurer les vues manquantes. Plusieurs pistes intéressantes pourraient être poursuivies pour aller plus loin. Il serait notamment possible de tester le modèle avec des décodeurs différents, et non pas seulement des gaussiennes multivariées. Autre exemple, comme le modèle s'y prête, on pourrait aussi examiner la question de la prédiction de d'arêtes dans un cadre où les nœuds sont multi-attribués. Le prochain chapitre sera consacré à l'établissement de métriques entre graphes, et plus particulièrement de métriques spécifiquement construites pour réaliser certaines tâches notamment de classification de graphes.





## Chapitre 5

# Apprentissage de métrique entre graphes

*Dans ce chapitre, nous présentons Simple Graph Metric Learning – SGML – un modèle d’apprentissage de métrique – AM – entre graphes. Ces dernières années, l’automatisation des tâches d’apprentissage sur graphes s’est faite essentiellement par le développement de modèles basés sur des GNN. Étonnamment, il existe à ce jour peu de méthodes d’apprentissage de métriques sur graphes, alors qu’il en existe un grand nombre sur les données euclidiennes. Pourtant, l’AM a un intérêt certain pour les graphes : certains noyaux sur graphes atteignent déjà des performances supérieures aux modèles basés sur des GNN et pourraient être améliorés par l’AM ; l’apprentissage de métrique peut également améliorer les performances d’algorithmes beaucoup plus simples et souvent moins coûteux à faire fonctionner. C’est pourquoi, nous proposons SGML, un modèle d’AM se basant sur Simple Graph Convolutional Networks – SGCN – et des éléments de la théorie de transport optimal ; il permet à partir de la connaissance de graphes étiquetés dans un jeu de données, de construire une métrique adaptée, notamment pour classifier ces graphes. Cette métrique possède peu de paramètres et s’entraîne relativement rapidement avec notamment des temps d’entraînement de l’ordre de la minute pour des jeux de données usuels. Le travail associé à ce chapitre est disponible sous la forme d’un rapport technique [Kaloga et al. 2021c].*

### 5.1 Introduction

Les distances sont parmi les outils les plus anciens et utiles de l’apprentissage automatique car la façon la plus simple de caractériser un objet est de lui attribuer les caractéristiques d’objets dont il est le plus proche. Ce principe est notamment exploité par certaines des méthodes les plus célèbres de l’apprentissage : la méthode des  $k$ -moyennes ( $k$ -means) [Lloyd 1957 ; MacQueen et al. 1967] permet de faire du *clustering* en cherchant pour un jeu de données une partition en  $k$  *clusters* de ses éléments qui minimise la somme des distances entre les éléments d’un *cluster* et leur barycentre. Autre exemple, la méthode des  $k$  plus proches voisins ( $k$ -nn) [Cover and Hart 1967] permet de faire de la classification en attribuant à chaque objet (d’étiquette inconnue) l’étiquette majoritairement présente chez ses  $k$  plus proches voisins. On peut aisément comprendre que la qualité des résultats de ces deux algorithmes sera hautement dépendante de la

distance choisie, d'où l'importance du choix de cette dernière. [Xing et al. \[2002\]](#) sont les premiers auteurs à avoir proposé une méthode d'AM en vue d'améliorer une tâche d'apprentissage, en l'occurrence il s'agissait de la méthode des  $k$  moyennes. Ces premiers travaux ont suscité un vif intérêt pour l'AM qui a abouti au développement d'un large panel de méthodes basées par exemple sur des modèles probabilistes [[Goldberger et al. ; Mensink et al. 2013](#)], des méthodes à principe de marge maximum [[Weinberger and Saul 2009](#)] ou de réduction dimensionnelle [[Li and Jain 2009 ; Wang and Zhang 2007](#)], etc. Pour plus de détails sur ces méthodes, on pourra se référer aux *reviews* de [Bellet et al. \[2013\]](#) et de [Suárez-Díaz et al. \[2018\]](#). Cette diversité de méthodes d'AM sur les données euclidiennes tranche avec celles sur graphes. Il existe en effet assez peu de méthodes d'AM *entre* graphes ; à cet égard on peut relever une certaine confusion dans l'utilisation du terme *Graph Metric Learning* qui désigne des tâches bien différentes dans la littérature comme l'apprentissage de métrique *entre* les nœuds du graphe ou encore l'approximation de distance *existante* entre les graphes difficile à calculer (voir la discussion à ce sujet en Annexe [A.4.1](#)). Quant aux méthodes qui cherchent à apprendre des métriques permettant de comparer spécifiquement des graphes, il existe quelques travaux notables :

- La *Graph Edit Distance* – GED –, est une distance au sens mathématique sur graphes et est égale au minimum d'opérations à effectuer (addition, substitution, suppression de nœuds ou d'arêtes) pour transformer un graphe en un autre. Dans sa formulation classique, la GED attribue un coût unitaire à toutes ces opérations. Il existe toute une série d'articles [[Bellet et al. 2012 ; Neuhaus and Bunke 2007 ; Jia et al. 2021](#)] (liste non exhaustive) qui traite de l'AM sur graphes en estimant les coûts optimaux à attribuer à ces opérations en vue de résoudre certaines tâches. L'inconvénient majeur de ces méthodes est la nécessité de calculer la GED<sup>1</sup> qui est très coûteuse et ne peut donc être effectué que pour des graphes de tailles modestes ; ces méthodes se limitent d'ailleurs souvent à certains types de graphes comme les arbres où la GED est plus facile à calculer.
- Certains auteurs ont également proposé de construire des distances à partir de réseaux de neurones. [Ktena et al. \[2018\]](#) proposent un tel modèle dans le cadre de l'étude des signaux de connectivités cérébrales qui peuvent être représentés comme des signaux sur graphe. Le graphe en question est le même pour tous les signaux. Dans ce modèle, un réseau de convolution sur graphe est utilisé pour extraire les caractéristiques de ces signaux de connectivités sur graphes. Ensuite, pour deux de ces signaux, un produit scalaire (nœud par nœud) est appliqué. Les résultats de ces produits scalaires sont utilisés en entrée d'un réseau de neurones final pour produire la mesure de similarité entre les signaux sur graphes. Le réseau de convolution sur graphes et le réseau de neurones final sont entraînés pour que la mesure de similarité soit « élevée » lorsque les signaux sur graphes ont la même étiquette et « faible » dans le cas contraire. Comme leur modèle fait intervenir un produit scalaire entre graphes, il nécessite donc que ces derniers aient la même structure, ce qui constitue une forte limite de la méthode.
- D'autres auteurs construisent également des modèles sans réseaux de neurones. [Yoshida et al. \[2021\]](#) proposent *Interpretable Graph Metric Learning*, une méthode d'AM, qui évalue de la similarité entre graphes en comptabilisant les sous-graphes les plus pertinents pour effectuer une tâche de classification. Leur méthode a l'avantage de proposer un

---

1. C'est un problème NP-Complet.

certain degré d'interprétabilité puisque les sous-graphes extraits renseignent sur les parties des graphes qui aident à la prédiction. Toutefois, leur méthode ne peut pas traiter de graphe de très grande taille. Zhao and Wang [2019] proposent d'apprendre un noyau en s'appuyant sur des notions d'homologie. Le modèle résultant est également performant, mais il présente l'inconvénient de ne pas pouvoir exploiter les informations portées par les nœuds lorsqu'elles sont de natures discrètes.

La moindre diversité des méthodes d'AM sur graphes peut s'expliquer par plusieurs facteurs. D'une part, comme longuement relaté dans le Chapitre 2, il est assez difficile de définir des mesures de similarités pertinentes entre graphes, il est encore plus difficile d'en construire qui soient différentiables ; c'est pourtant une condition quasi nécessaire pour faire de l'AM. De plus, l'attention qu'ont reçu les GNN ces dernières années a amoindri l'intérêt pour ces méthodes. Pourtant, les mesures de similarités et de distances qui permettent de construire des noyaux sur graphes sont très efficaces [Kriege et al. 2016 ; Shervashidze et al. 2011] et sont encore sur un bon nombre de tâches (notamment en classification) aussi efficaces voire supérieures aux modèles se basant sur des réseaux de neurones (voir Table (3.3)). Ces méthodes pourraient potentiellement être améliorées par l'AM. Par ailleurs, l'apprentissage de métrique sur graphes peut également améliorer les performances d'algorithmes beaucoup plus simples comme la méthode des  $k$  plus proches voisins ou l'algorithme des  $k$  moyennes. Dans un contexte où l'on souhaite réduire l'impact énergétique des méthodes d'apprentissage<sup>2</sup> (notamment des réseaux de neurones pouvant contenir des centaines de milliers voire des millions de paramètres), l'utilisation de l'AM pour l'utilisation *in fine* de modèles plus simples est souhaitable ; à condition que la procédure d'AM ne soit pas elle-même trop coûteuse.

Pour répondre à ces problématiques, nous proposons dans ce chapitre une méthode simple d'apprentissage de métrique sur graphes que nous appelons SGML pour *Simple Graph Metrics Learning*. La méthode est notamment inductive et possède peu de paramètres. L'entraînement des poids entraînaibles se fait notamment en temps linéaire et peut être effectué seulement sur une fraction des données. Nous montrerons que cette méthode a des performances compétitives avec l'état de l'art en classification de graphes que ce soit en utilisation directe via un noyau ou par l'intermédiaire de méthodes de classifications plus simples comme la méthode des  $k$  plus proches voisins.

En Section 5.2, nous présenterons les principales définitions nécessaires à la compréhension de ce chapitre ainsi que le contexte dans lequel s'inscrivent nos travaux. Nous y présenterons également des éléments de compréhension du transport optimal sur lesquels se basent nos travaux ainsi que des distances et mesures de similarités sur graphes se basant sur le transport optimal. En Section 5.3, nous présenterons notre modèle et enfin en Section 5.4, nous présenterons quelques expériences mettant en jeu la méthode.

## 5.2 Définitions et contexte

Dans ce travail, nous considérons un jeu de données noté  $\mathbb{G}_x$  et constitué de graphes attribués (mais non pondérés)  $\mathcal{G} \in \mathbb{G}_x$ . Pour alléger les notations on n'utilisera que les notations,

---

2. D'où l'engouement ces dernières années pour le *transfer learning* et les travaux visant à la réduction de la taille des architectures de réseaux de neurones.



vectorielles introduites dans le Chapitre 2, c'est-à-dire que pour tout  $u_i \in V$ ,  $\mathbf{x}_i = \mathbf{x}(u_i) \in \mathbb{R}^q$  et  $\mathbf{X} \in \mathbb{R}^{n \times q}$  tels que  $\mathbf{X}(:, i) = \mathbf{x}(u_i)$ . Les graphes de  $\mathbb{G}_x$  sont étiquetés, on note  $\mathbb{E} = \{e_1, \dots, e_{|\mathbb{E}|}\}$ <sup>3</sup> l'ensemble de ces étiquettes et  $\mathcal{E} : \mathbb{G}_x \rightarrow \mathbb{E}$  la fonction qui étiquette les graphes. Les différentes notations introduites dans le Chapitre 2 seront également utilisées dans ce chapitre.

**Definition 3 (Distance.)** Une distance  $d$  sur un espace  $\mathbb{X}$  est une fonction  $d : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}_+$  qui vérifie les propriétés suivantes :

- (**symétrie**)  $\forall (x, y) \in \mathbb{X}^2, d(x, y) = d(y, x)$
- (**séparation**)  $\forall (x, y) \in \mathbb{X}^2, d(x, y) = 0 \Leftrightarrow x = y$
- (**inégalité triangulaire**)  $\forall (x, y, z) \in \mathbb{X}^3, d(x, z) \leq d(x, y) + d(y, z)$

En général, la propriété la plus difficile à assurer sur les graphes est une des implications de la **séparation** i.e  $\forall x, y \in \mathbb{X}^2, d(x, y) = 0 \Rightarrow x = y$ . Cette condition est souvent relaxée, on parle alors de **pseudo-distance**.

### 5.2.1 Apprentissage de métrique

L'apprentissage de métrique suppose que l'on ait une distance paramétrable. Lorsque l'espace  $\mathbb{X} = \mathbb{R}^q$  est un espace euclidien, la distance canonique pour ce type de tâche est la distance de Mahalanobis.

#### 5.2.1.a Distance de Mahalanobis

**Definition 4 (Distance de Mahalanobis.)** Soit  $\mathbf{M} \in \mathbb{R}^{q \times q} \succcurlyeq 0$ , une matrice semi-définie positive, la distance de Mahalanobis associée à cette matrice s'écrit :

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^q, d_{\mathbf{M}}(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T \mathbf{M} (\mathbf{x} - \mathbf{y})} \quad (5.1)$$

La distance de Mahalanobis a été introduite dans les années 30 par le mathématicien indien Prasanta Chandra Mahalanobis pour détecter des données aberrantes dans des échantillons statistiques. Dans ce contexte,  $\mathbf{M} = \mathbf{\Sigma}^{-1}$  est la matrice de covariance empirique des échantillons. Aujourd'hui, cette distance est très présente dans l'apprentissage de métrique où les coefficients de  $\mathbf{M}$  correspondent aux poids entraînaibles. Pour simplifier la tâche d'optimisation, notamment vis-à-vis de la contrainte de semi-définie positivité de  $\mathbf{M}$ , on s'appuie sur le théorème de décomposition des matrices semi-définies positives qui indique que :

$$\forall \mathbf{M} \succcurlyeq 0, \text{rang}(\mathbf{M}) = k \Leftrightarrow \exists \mathbf{N} \in \mathbb{R}^{k \times n}, \mathbf{M} = \mathbf{N}^T \mathbf{N} \text{ avec } \text{rang}(\mathbf{N}) = k \quad (5.2)$$

Cette matrice est unique à une isométrie (i.e matrice orthonormée) près et pour une telle matrice  $\mathbf{N}$ , on peut alors écrire :

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^q, d_{\mathbf{N}}(\mathbf{x}, \mathbf{y}) = \sqrt{\|\mathbf{N}(\mathbf{x} - \mathbf{y})\|_2^2} = \sqrt{\|\mathbf{N}\mathbf{x} - \mathbf{N}\mathbf{y}\|_2^2} \quad (5.3)$$

et  $d_{\mathbf{M}}(\mathbf{x}, \mathbf{y}) = d_{\mathbf{N}}(\mathbf{x}, \mathbf{y})$ . Cette formulation permet de s'affranchir de la condition de semi-définie positivité de  $\mathbf{M}$  en effectuant l'optimisation directement sur les coefficients de  $\mathbf{N}$ ; de

3. On notera que  $|\mathbb{E}|$  désigne le nombre d'étiquettes possibles et non le nombre d'arêtes qui lui est noté  $|\mathbb{E}|$ .

plus, on obtient un paramètre de contrôle supplémentaire  $k \in \{1, \dots, n\}$  qui s'interprète comme la dimension dans laquelle la distance va être évaluée.<sup>4</sup> Par contre, l'optimisation de  $N$  à un désavantage non négligeable, nombre de fonctionnelles d'AM font intervenir le carré des distances ou leurs opposés, or alors que la fonction  $(\mathbf{x} - \mathbf{y}) \mapsto -d_M(\mathbf{x}, \mathbf{y})^2$  est convexe, la fonction  $(\mathbf{x} - \mathbf{y}) \mapsto -d_N(\mathbf{x}, \mathbf{y})^2$  est concave.

Pour conclure, un grand nombre de méthodes paramètrent leur métrique selon (5.3), leur objectif est donc d'établir la projection  $N$  optimale selon des critères relativement différents mais qui obéissent tous à quelques principes directeurs caractéristiques des méthodes d'apprentissage de métriques.

### 5.2.1.b Principes directeurs

Dans le cadre de l'AM, on suppose qu'un jeu de données fini  $\mathbb{X} = \{\mathbf{x}_i\}_{i=1}^{|\mathbb{X}|} \subset \mathbb{R}^q$  est adjoint de la connaissance de deux ensembles  $\mathcal{S}$  (semblable) et  $\mathcal{D}$  (dissemblable) contenant des paires des certains éléments de  $\mathbb{X}$ . Les paires d'éléments dans  $\mathcal{S}$  doivent être proches tandis que les paires dans  $\mathcal{D}$  doivent au contraire être lointaines pour la distance qu'on souhaite construire<sup>5</sup>. Ces ensembles sont souvent construits à partir des étiquettes des éléments de  $\mathbb{X}$  :

$$\mathcal{S} = \left\{ \{\mathbf{x}_i, \mathbf{x}_j\} \in \mathbb{X} \times \mathbb{X} \mid i \neq j, \mathcal{E}(\mathbf{x}_i) = \mathcal{E}(\mathbf{x}_j) \right\} \quad (5.4)$$

$$\mathcal{D} = \left\{ \{\mathbf{x}_i, \mathbf{x}_j\} \in \mathbb{X} \times \mathbb{X} \mid i \neq j, \mathcal{E}(\mathbf{x}_i) \neq \mathcal{E}(\mathbf{x}_j) \right\} \quad (5.5)$$

En se basant sur les informations apportées par ces ensembles, on souhaite construire une distance  $d$  sur  $\mathbb{X}$  qui va aider aux tâches d'apprentissage sur cet ensemble. Un modèle à minimiser (ou à maximiser) dépendant de  $d$ ,  $\mathcal{S}$ ,  $\mathcal{D}$  est alors défini dans cet objectif.

$$\min_d \mathcal{F}(d, \mathcal{S}, \mathcal{D}) \quad (5.6)$$

La minimisation par rapport à  $d$  se fait généralement par l'intermédiaire de ses paramètres (par exemple  $M$  pour la distance Mahalonobis  $d_M$ ). La forme que peut prendre  $\mathcal{F}$  sera discutée dans la partie suivante.

**Remarque 23 (Éléments non étiquetés.)** *L'intérêt de ces méthodes tient dans le fait que  $\mathbb{X}$  est inclus dans un ensemble plus grand comportant des éléments qui eux ne sont pas étiquetés. La distance alors construite sera alors à même d'aider à retrouver ces étiquettes.*

### 5.2.1.c Méthodes d'apprentissage de métrique

Parmi la très grande diversité de méthodes d'AM, on peut citer quelques-unes d'entre elles qui se distinguent à la fois par leur simplicité et leur efficacité. On pourra se référer à la *review*

4. Cela établit le lien entre l'AM et les modèles de réduction dimensionnelle [Suárez-Díaz et al. 2018]. À cet égard, la PCA (ou même la CCA) peuvent être vues comme des méthodes d'apprentissage de métriques. Par exemple, il est bien connu que pour les jeux de données de grande dimension, il est plus efficace d'appliquer l'algorithme des  $k$ -moyennes après application de la PCA que sur les données d'origine directement.

5. Certains algorithmes utilisent un troisième type d'informations, qui consiste en des triplets indiquant qu'un élément donné doit être plus proche de tel élément que de tel autre élément [Bellet et al. 2013].

faite par Suárez-Díaz et al. [2018] qui ont procédé à une comparaison expérimentale exhaustive de ces méthodes.

- **Neighbourhood Components Analysis - NCA.** [Goldberger et al.] Dans ce modèle (probabiliste), la distance est une distance de Mahalanobis (voir Eq. (5.3)) paramétrée par la projection linéaire  $\mathbf{N}$ . Muni de cette distance, on définit simplement la probabilité qu'un élément  $\mathbf{x}_i \in \mathbb{X}$  ait pour plus proche voisin un autre voisin  $\mathbf{x}_j \in \mathbb{X}$  ( $i \neq j$ ) :

$$p^{\mathbf{N}}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\exp(-\|\mathbf{N}\mathbf{x}_i - \mathbf{N}\mathbf{x}_j\|_2^2)}{\sum_{\substack{k=1 \\ k \neq i}}^{|\mathbb{X}|} \exp(-\|\mathbf{N}\mathbf{x}_i - \mathbf{N}\mathbf{x}_k\|_2^2)} \quad (5.7)$$

Ce modèle est construit pour améliorer les résultats de la méthode des  $k$  plus proches voisins. Pour réaliser cet objectif, on cherche la distance qui maximise les probabilités que les éléments qui ont les mêmes étiquettes soient les proches voisins les uns des autres, c'est-à-dire les probabilités  $p(\mathbf{x}_i, \mathbf{x}_j)$  lorsque  $\{\mathbf{x}_i, \mathbf{x}_j\} \in S$ , ce qui mène à la résolution du problème suivant (définissant donc  $\mathcal{F}$ ) :

$$\max_{\mathbf{N}} \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in S} p^{\mathbf{N}}(\mathbf{x}_i, \mathbf{x}_j). \quad (5.8)$$

La distance peut aussi être apprise par descente de gradient stochastique en évaluant cette fonctionnelle sur des *batch* de données, ce qui permet d'avoir un modèle linéaire en la taille des données  $|\mathbb{X}|$ .

- **Nearest Class Mean Metric Learning - NCMML.** [Mensink et al. 2013] Ce modèle est également probabiliste et est basé sur la même distance. Par contre, il est cette fois-ci construit pour améliorer l'algorithme du barycentre de classe le plus proche (*Nearest Class Mean Classifier*). Il s'agit d'un modèle de classification qui opère en calculant d'abord les barycentres<sup>6</sup> des éléments appartenant à une même classe (i.e ayant la même étiquette), puis la phase de classification se fait en attribuant aux éléments non étiquetés la classe du barycentre dont ils sont le plus proche. On note pour  $l = \{1, \dots, |\mathbb{E}|\}$ ,

$$\boldsymbol{\mu}_{e_l} = \frac{1}{|\mathcal{E}^{-1}(e_l)|} \sum_{\substack{i=1 \\ \mathcal{E}(\mathbf{x}_i)=e_l}}^{|\mathbb{X}|} \mathbf{x}_i \quad (5.9)$$

le barycentre des éléments de  $\mathbb{X}$  dont l'étiquette est  $e_l \in \mathbb{E}$ . La probabilité  $p(e_l|\mathbf{x}_i)$  pour un élément  $\mathbf{x}_i$  d'avoir l'étiquette  $e_l$  est choisie telle que :

$$p^{\mathbf{N}}(e_l|\mathbf{x}_i) = \frac{\exp(-\frac{1}{2}\|\mathbf{N}\mathbf{x}_i - \mathbf{N}\boldsymbol{\mu}_{e_l}\|_2^2)}{\sum_{\substack{l=1 \\ l' \neq l}}^{|\mathbb{E}|} \exp(-\frac{1}{2}\|\mathbf{N}\mathbf{x}_i - \mathbf{N}\boldsymbol{\mu}_{e_{l'}}\|_2^2)} \quad (5.10)$$

On souhaite alors maximiser la probabilité de chaque  $\mathbf{x}_i \in \mathbb{X}$  d'avoir sa véritable étiquette  $\mathcal{E}(\mathbf{x}_i)$ , d'où le problème suivant :

$$\max_{\mathbf{N}} \frac{1}{|\mathbb{E}|} \sum_{\mathbf{x}_i \in \mathbb{X}} \log p^{\mathbf{N}}(\mathcal{E}(\mathbf{x}_i)|\mathbf{x}_i) \quad (5.11)$$

---

6. On pourrait aussi considérer ces barycentres comme des prototypes de classe.

La distance peut également être apprise par descente de gradient stochastique.

- **Large Margin Nearest Neighbors - LMNN.** [Weinberger and Saul 2009] Cette méthode n'est cette fois-ci pas probabiliste mais comme NCA elle est construite pour améliorer spécifiquement la méthode des  $k$  plus proches voisins, en particulier pour une valeur de  $k$  spécifique choisie à l'avance. La distance à apprendre est de nouveau une distance de Mahalanobis paramétrée par  $\mathbf{N}$ . Cette méthode procède de la façon suivante :
  - Pour chaque élément  $\mathbf{x}_i \in \mathbb{X}$ , on détermine ce qu'on appelle des cibles, c'est-à-dire des éléments  $\mathbf{x}_j$ , tels que  $\mathcal{E}(\mathbf{x}_i) = \mathcal{E}(\mathbf{x}_j)$  dont on désire que  $\mathbf{x}_i$  soit proche. Lorsque  $\mathbf{x}_j$  est la cible de  $\mathbf{x}_i$ , on écrit que  $\mathbf{x}_i \rightarrow \mathbf{x}_j$ . On ne choisit que *certaines* des éléments avec la même étiquette que  $\mathbf{x}_i$  comme cible de ce dernier. Ces cibles peuvent être choisies par exemple en sélectionnant les  $p$  plus proches éléments (de même étiquette) de  $\mathbf{x}_i$  pour la distance euclidienne classique. On remarquera que pour cette définition la relation de ciblage n'est pas symétrique. Par ailleurs, ce choix de ciblage aura tendance à améliorer en particulier la méthode des  $k$  plus proches voisins lorsque  $k = p$ . Les cibles sont fixées en amont une fois pour toutes.
  - L'objectif de cette méthode est alors d'apprendre une distance qui rapproche  $\mathbf{x}_i$  de ses cibles tout en repoussant les *intrus*. Les *intrus* pour  $\mathbf{x}_i$  sont des éléments qui ont un étiquetage différent de ce dernier et qui se trouvent à une distance de lui inférieure à un certain périmètre (voir Fig. (5.12)). Ce périmètre (au carré) se définit tout simplement comme la distance (au carré) de  $\mathbf{x}_i$  à sa cible  $\mathbf{x}_j$  la plus éloignée plus une marge qui est choisie généralement égale à 1. Plus formellement,  $\mathbf{x}_k$  est un intru s'il existe  $\mathbf{x}_j$ , tel que  $\mathbf{x}_i \rightarrow \mathbf{x}_j$  et  $\|\mathbf{N}\mathbf{x}_i - \mathbf{N}\mathbf{x}_k\|_2^2 \leq \|\mathbf{N}\mathbf{x}_i - \mathbf{N}\mathbf{x}_j\|_2^2 + 1$ . Voir illustration de gauche, Figure 5.1. La marge existe dans le cas où les cibles de  $\mathbf{x}_i$  se trouvent toutes dans un périmètre extrêmement proche de ce dernier, sans cette marge la distance construite pourrait autoriser des intrus très proches de  $\mathbf{x}_i$ . (Voir illustration de droite, Figure 5.1.)
  - Pour réaliser cet objectif, cette méthode construit d'abord une fonctionnelle qui se compose d'une partie à minimiser dite « attractive » qui a pour objectif de rapprocher les  $\mathbf{x}_i \in \mathbb{X}$  de leurs cibles :

$$\mathcal{F}_{pull} = \sum_{i=1}^{|\mathbb{X}|} \sum_{\substack{j=1 \\ \mathbf{x}_i \rightarrow \mathbf{x}_j}}^{|\mathbb{X}|} \|\mathbf{N}\mathbf{x}_i - \mathbf{N}\mathbf{x}_j\|_2^2 \quad (5.12)$$

Et d'une partie répulsive à minimiser également pour rejeter les *intrus* :

$$\mathcal{F}_{push} = \sum_{i=1}^{|\mathbb{X}|} \sum_{\substack{j=1 \\ \mathbf{x}_i \rightarrow \mathbf{x}_j}}^{|\mathbb{X}|} \sum_{l=1}^{|\mathbb{X}|} (1 - \delta_{\mathcal{E}(\mathbf{x}_i), \mathcal{E}(\mathbf{x}_l)}) \max \left\{ 1 + \|\mathbf{N}\mathbf{x}_i - \mathbf{N}\mathbf{x}_j\|_2^2 - \|\mathbf{N}\mathbf{x}_i - \mathbf{N}\mathbf{x}_l\|_2^2, 0 \right\} \quad (5.13)$$

En introduisant la constante  $\gamma \in [0, 1]$  d'équilibrage entre ces deux parties, le problème final s'écrit :

$$\min_{\mathbf{N}} (1 - \gamma)\mathcal{F}_{push} + \gamma\mathcal{F}_{pull} \quad (5.14)$$

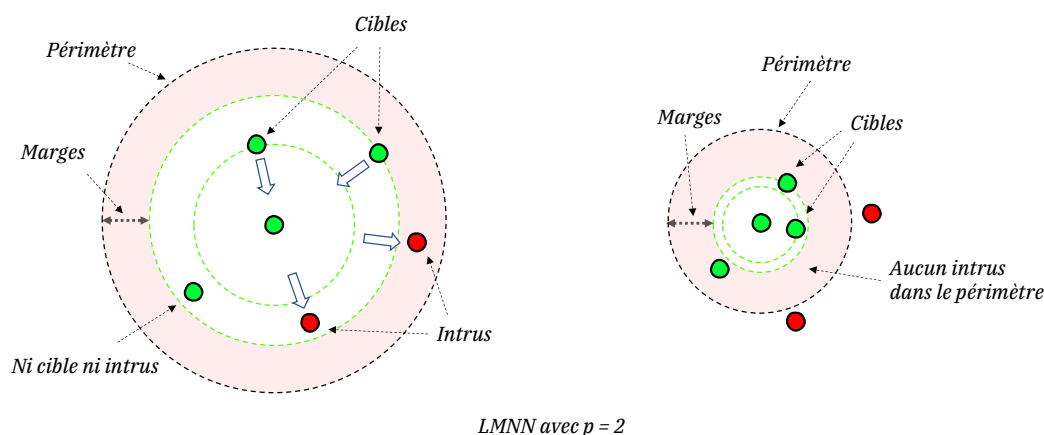


FIGURE 5.1 – *Illustration LMNN.* **A gauche**, on peut voir quelques éléments représentés dans l'espace de la distance paramétrée par  $N$ . Les couleurs représentent les étiquettes des éléments. Notre élément central a deux cibles que l'algorithme essaye de rapprocher mais également deux intrus dans son périmètre que l'algorithme essaye de repousser. **A droite**, la situation lorsque les cibles sont très proches de l'élément central, la marge permet d'empêcher l'algorithme de s'autoriser des situations où les intrus sont très proches du nœud central. Cette illustration ne présente la situation que pour un élément, on peut faire ce type de diagramme pour tous les éléments du jeu de données.

Cette fonctionnelle permet l'apprentissage de la distance par descente de gradient stochastique, mais il faut prêter attention à la taille des *batch* pour qu'ils contiennent des cibles et des intrus.

Nous avons formulé une version de notre procédure d'AM pour graphe pour chacune de ces fonctionnelles, en proposant notamment une adaptation de NCMML pour améliorer l'algorithme des  $k$  plus proches voisins (en Section 5.3.3).

### 5.2.2 Transport optimal

Pour pouvoir utiliser l'une des fonctionnelles de transport optimal introduites plus haut pour faire de l'AM sur graphes, il faut une distance paramétrée et entraînable sur graphe. Pour construire une telle distance, le transport optimal est un outil d'intérêt. En effet, depuis l'essor récent du transport optimal en apprentissage, quelques distances et noyaux sur graphes relativement efficaces ont été construits sur la base de cette théorie. Nous introduirons brièvement dans cette section la théorie du transport optimal, puis dans la section suivante les mesures de similarités entre graphes basées sur cette dernière, auxquelles nous pourrons nous comparer.

La théorie du transport optimal prend ses origines dans le manuscrit *Mémoire sur la théorie des déblais et des remblais* [Monge 1781] dans lequel le mathématicien français Gaspard Monge explore la question de l'optimalité du moyen de transport d'objets localisés à certains endroits (des déblais) vers d'autres localités (des remblais). La formulation initiale de Monge en termes de fonction impliquait que les remblais pouvaient être alimentés par plusieurs déblais mais qu'à

l'inverse les déblais ne pouvaient être répartis vers plusieurs remblais. De ce fait, il y avait de nombreuses situations où le transport ne pouvait même pas être réalisé. Voir illustration du milieu, Figure 5.2. C'est 150 ans plus tard que Kantorovich [1942] résout ce problème en lui donnant une formulation probabiliste qui admet toujours une solution. Cette formulation a longuement été développée par des mathématiciens [Brenier 1991 ; Villani 2003] et puis plus récemment elle a attiré l'intérêt de la communauté de l'apprentissage automatique car cette théorie permet d'une part de calculer des distances entre des distributions, outils au cœur des problématiques modernes de ce domaine, et d'autre part de calculer ces distances relativement rapidement grâce à des travaux récents [Cuturi 2013].

### 5.2.2.a Distance 2-Wasserstein

Nous n'aborderons que le cas des distributions discrètes. Soit  $\mathbb{X} = \{\mathbf{x}_i \in \mathbb{R}^q | i \in \{1, \dots, n\}\}$ ,  $\mathbb{Y} = \{\mathbf{y}_i \in \mathbb{R}^q | i \in \{1, \dots, m\}\}$  deux ensembles discrets ; soit  $\mu \in \mathcal{P}(\mathbb{X})$  et  $\nu \in \mathcal{P}(\mathbb{Y})$  deux distributions respectivement sur  $\mathbb{X}$  et  $\mathbb{Y}$ . Ces distributions s'écrivent :

$$\mu = \sum_{\mathbf{x}_i \in \mathbb{X}} a_i \delta_{\mathbf{x}_i} \quad \text{et} \quad \nu = \sum_{\mathbf{y}_i \in \mathbb{Y}} b_i \delta_{\mathbf{y}_i} \quad (5.15)$$

où les  $(a_i)_{i=1}^n$ ,  $(b_i)_{i=1}^m$  sont des coefficients positifs sommant à un, i.e  $\sum_{i=1}^n a_i = 1$  et  $\sum_{i=1}^m b_i = 1$ . L'objet de la théorie du transport optimal est la détermination du transport de coût minimal qui permet de transporter la distribution  $\mu$  vers  $\nu$  (et inversement). Une illustration peut être trouvée en Figure (5.2). Le problème formel s'écrit :

$$\pi^* = \operatorname{argmin}_{\pi \in \Pi_{a,b}} \sum_{i,j=1}^{n,m} \pi_{i,j} c(\mathbf{x}_i, \mathbf{y}_j) \quad (5.16)$$

où  $c : \mathbb{X} \times \mathbb{Y} \rightarrow \mathbb{R}_+$  est une fonction de coût qui correspondra au carré de la distance euclidienne dans notre travail. Par ailleurs,  $\Pi_{a,b}$  est l'ensemble des distributions jointes sur  $\mathbb{X} \times \mathbb{Y}$ ,  $\pi = \sum_{i,j=1}^{n,m} \pi_{i,j} \delta_{(\mathbf{x}_i, \mathbf{y}_j)}$  dont les marginales sont les distributions  $\mu$  et  $\nu$ , i.e :

$$\Pi_{a,b} = \left\{ \pi \in P(\mathbb{X} \times \mathbb{Y}) \mid b_j = \sum_{i=1}^n \pi_{i,j} \quad \text{et} \quad a_i = \sum_{j=1}^m \pi_{i,j} \right\} \quad (5.17)$$

Les plans de transports optimaux peuvent être utilisés pour construire une distance entre les distributions. Lorsque la fonction de coût est le carré de la norme 2, on peut calculer la distance de 2-Wasserstein  $\mathcal{W}_2$  entre les deux distributions marginales  $\mu$  et  $\nu$  avec ce plan de transport, définie comme :

$$\mathcal{W}_2(\mu, \nu)^2 = \sum_{i,j=1}^{n,m} \pi_{i,j}^* \|\mathbf{x}_i - \mathbf{y}_j\|_2^2 \quad (5.18)$$

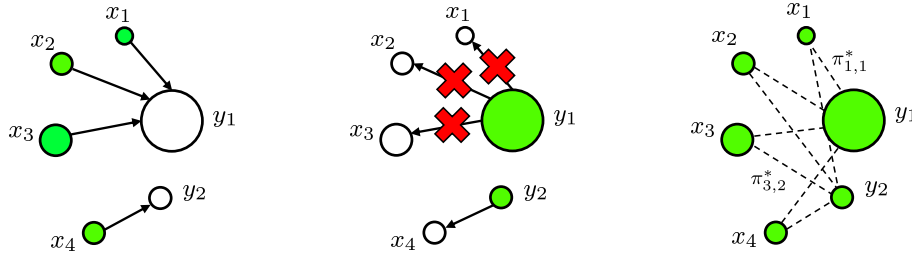


FIGURE 5.2 – *Illustration du transport optimal.* En vert les déblais et en blanc les remblais. La taille des cercles indique le niveau de probabilité des éléments considérés. **A gauche**, Le transport optimal existe ici car les poids sont compatibles avec l'existence d'un transport. **Au milieu**, le transport inverse n'est pas possible car les poids ne sont pas compatibles, notamment car on est limité à une seule assignation par déblai. **A droite**, la formulation probabiliste, on peut maintenant diviser les probabilités dans un sens comme dans l'autre autant qu'on le souhaite, le problème de transport devient symétrique.

Il s'agit d'une manière efficace d'évaluer les distances entre les distributions avec des propriétés de régularité (notamment par rapport à la divergence de Kullback-Leibler) qui la rendent très intéressante, notamment lorsqu'on souhaite différencier la distance par rapport aux paramètres d'une distribution <sup>7</sup> [Arjovsky et al. 2017]. De par cette propriété, cette mesure est sous-jacente à de nombreux modèles d'apprentissage automatique dont les célèbres réseaux de neurones antagonistes génératifs (*Generative Adversarial Networks* - GAN) mais aussi des méthodes de *transfer learning* très impressionnantes [Murez et al. 2018]. Des variations autour de cette distance peuvent être avantageusement utilisées pour construire des distances entre graphes (Section 5.2.2.c).

**Remarque 24 (Complexité du calcul du plan de transport.)** *Pour simplifier considérons que  $n = m$ , le problème (5.16) peut être résolu par l'algorithme du simplexe et sa résolution se fait en  $O(n^3 \log n)$  [Cuturi 2013], c'est également la complexité de la distance de 2-Wasserstein.*

### 5.2.2.b Distribution unidimensionnelle

Un cas particulier intéressant pour la distance de 2-Wasserstein est celui où les deux distributions sont unidimensionnelles et uniformes.

Lorsque de plus,  $n = m$ , le calcul de la distance de 2-Wasserstein est très simple, il suffit de trier par ordre croissant les supports des distributions respectivement  $\{x_i\}_{i=1}^n$  et  $\{y_i\}_{i=1}^n$ . En notant  $\sigma$  (resp.  $\rho$ ) la permutation qui à  $i$  associe l'indice du  $i$ -ème plus petit élément de  $\{x_i\}_{i=1}^n$  (resp.)  $\{y_i\}_{i=1}^n$ . Alors  $\pi_{i,j}^* = \frac{1}{n}$  si  $\sigma^{-1}(i) = \rho^{-1}(j)$  et 0 sinon. Donc :

$$\mathcal{W}_2(\mu, \nu)^2 = \sum_{i=1}^n (x_{\sigma(i)} - y_{\rho(i)})^2 \quad (5.19)$$

7. Par exemple, lorsque  $\mu_\theta$  est une distribution paramétrée par  $\theta$  et  $\nu$  une autre distribution, on peut aisément différencier  $\theta \mapsto \mathcal{W}_2(\mu_\theta, \nu)$ .

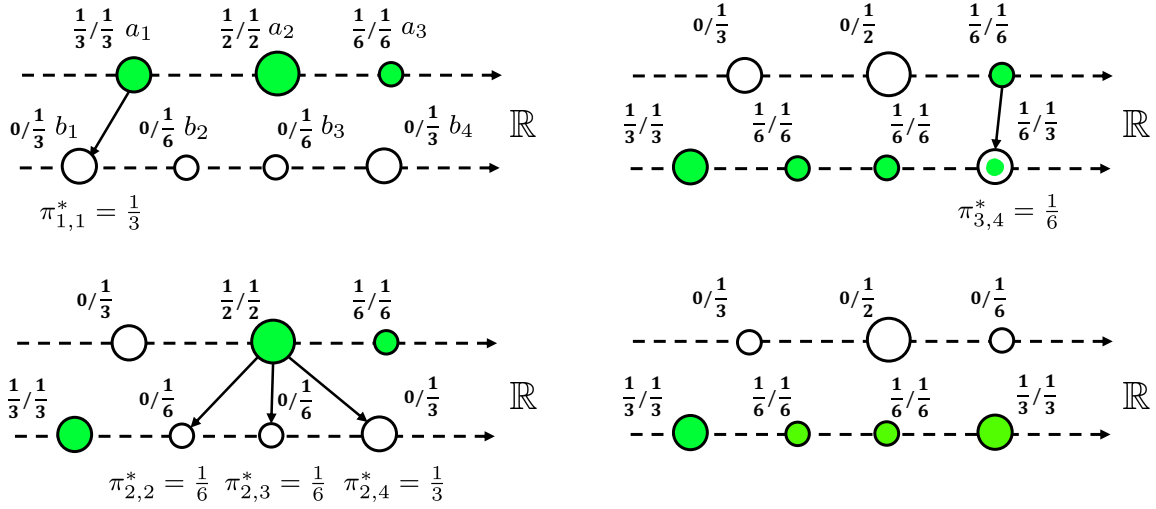


FIGURE 5.3 – *Illustration du transport optimal unidimensionnel.* De haut en bas, puis de gauche à droite, on montre la construction pas à pas de  $\pi^*$  quand  $n \neq m$ .

La complexité de ce calcul est dominée par le tri qui la rend en  $O(n \log n)$ . Lorsque  $n > m$ , il faut de nouveau trier les éléments de  $\{x_i\}_{i=1}^n$  et  $\{y_i\}_{i=1}^m$ . Le calcul de la distance est légèrement différent mais à peine plus complexe. Le transport optimal qui permet de calculer la distance est celui qui envoie toujours la masse (de probabilité) du plus petit des  $x_i$  vers le plus petit des  $y_j$  avec de la masse manquante. La Figure 5.3 donne un exemple d'un tel transport. Le plan de transport se calcule facilement avec au plus  $n + m$  coefficients non nuls. On a donc besoin d'évaluer la fonction de coût au plus  $n + m$  fois, la complexité est donc  $O(n + m + n \log n + m \log m)$ . On remarquera que (modulo le tri) le plan de transport optimal est strictement déterminé par les valeurs de  $n$  et de  $m$ .

### 5.2.2.c Variation autour de la distance de 2-Wasserstein

Notre modèle va impliquer le calcul d'une distance de 2-Wasserstein, toutefois telle que définie en (5.18), son calcul est trop coûteux. Dans la littérature, on peut distinguer deux approches permettant d'éviter ce coûteux calcul : la régularisation entropique et *sliced*-Wasserstein.

**Régularisation entropique.** [Cuturi 2013] Une première approche consiste à utiliser le plan de transport  $\pi_e^*$  (Eq. 5.20) pour calculer la distance de 2-Wasserstein (définie en Eq. 5.18) :

$$\pi_e^* = \operatorname{argmin}_{\pi \in \Pi_{a,b}} \sum_{i,j=1}^{n,m} \pi_{i,j} \|\mathbf{x}_i - \mathbf{y}_j\|_2^2 + \gamma \pi_{i,j} \log \pi_{i,j} \quad (5.20)$$

Le terme supplémentaire introduit une régularisation entropique, qui permet de *convexifier* le problème qui peut alors être résolu par l'algorithme de Sinkhorn. Le plan de transport ainsi



obtenu permet d'approximer la distance de 2-Wasserstein relativement rapidement, avec une complexité quadratique  $O(n^2)$  contre quasi-cubique  $O(n^3 \log n)$  auparavant. Plus la constante de régularisation  $\gamma > 0$  est élevée, plus le calcul peut-être effectué rapidement mais ce gain se fait au prix d'un transport de moins en moins *optimal* vis-à-vis du problème initial (5.16). Il faut donc judicieusement choisir sa valeur. On notera que la distance de 2-Wasserstein entropique ainsi calculée n'est plus réellement une distance. En effet, du fait de la sous-optimalité de  $\pi_e^*$ , on a plus la garantie que  $d(x, x) = 0$ , l'axiome de **séparation** n'est donc plus vérifié.

**Sliced-Wasserstein.** [Bonneel et al. 2015] Une autre solution consiste à comparer les mesures  $\mu$  et  $\nu$  via des projections de ces dernières en dimension 1. Soit  $\theta \in \mathbb{S}^{q-1}$  un vecteur de la sphère unitaire de  $\mathbb{R}^q$ . Les distributions  $\mu$  et  $\nu$  projetées selon  $\theta$  sont notées  $\mu_\theta$  et  $\nu_\theta$  où :

$$\mu_\theta = \sum_{x_i \in \mathbb{X}} a_i \delta_{x_i \cdot \theta} \quad \text{et} \quad \nu_\theta = \sum_{y_i \in \mathbb{Y}} b_i \delta_{y_i \cdot \theta} \quad (5.21)$$

La comparaison entre  $\mu$  et  $\nu$  est effectuée en comparant leurs projections unidimensionnelles sur la sphère unité via la distance de 2-Wasserstein (qui comme on l'a vu est très simple à calculer en dimension 1) et en moyennant ces distances. La distance ainsi calculée est nommée *sliced-Wasserstein* (on omettra le 2) et son expression est la suivante :

$$\mathcal{SW}_2(\mu, \nu)^2 = \int_{\mathbb{S}^{q-1}} \mathcal{W}_2(\mu_\theta, \nu_\theta)^2 d\theta \quad (5.22)$$

Cette distance est liée à la transformée de Radon [Bonneel et al. 2015], c'est une transformée notamment utilisée en imagerie médicale où elle permet de recouvrer une image tridimensionnelle à partir d'images en coupe de cette dernière<sup>8</sup>. Plus généralement, cette transformée (bijective) permet d'associer à toute mesure de  $\mathbb{R}^q$  une mesure de  $\mathbb{R} \times \mathbb{S}^{q-1}$  ;  $\mathcal{SW}$ <sup>9</sup> est en fait la distance de 2-Wasserstein entre les transformées de Radon de  $\mu$  et de  $\nu$ . On pourra se référer à Bonneel et al. [2015] pour plus de détails sur cette transformée. La bijectivité de la transformée de Radon fait de cette distance est une *véritable* distance entre les distributions de  $\mathbb{R}^q$ . La principale difficulté de l'évaluation de cette distance consiste en le calcul de l'intégrale, cette dernière peut-être évaluée via un échantillonnage de Monte-Carlo, sa complexité est alors (lorsque  $n = m$ )  $O(M(n \log n))$  et au plus (quand  $n \neq m$ )  $O(M(m + n + n \log n + m \log m))$  avec  $M$  le nombre d'échantillons tirés pour  $\theta$ .

### 5.2.3 Distance et noyau sur graphe avec le transport optimal

Dans cette section, nous présenterons différentes mesures de similarités entre graphes basées sur du transport optimal et qui nous servirons notamment de référence pour comparer les performances de notre modèle.

8. L'opération de reconstruction est associée à l'inverse de cette transformée

9. On omet ici et pour la suite l'indice 2.

**Fused Gromov-Wasserstein.** [Titouan et al. 2019] La distance de *Fused* Gromov-Wasserstein (A.1) -  $\mathcal{FGW}$  - définie en Annexe A.2.2 est une distance sur l'ensemble des graphes attribués. Elle se base notamment sur la distance de Gromov-Wasserstein qui permet de calculer une distance entre des objets d'espaces structurellement différents<sup>10</sup>, comme des espaces euclidiens de dimensions différentes. En particulier, elle est tout à fait adaptée pour la comparaison d'objets comme les graphes. Expérimentalement, elle permet d'obtenir de très bons résultats en classification. Mais elle se distingue particulièrement par ses propriétés, qui permettent de donner un sens à la notion de barycentre de graphe et de le calculer ou encore de faire de l'interpolation de graphe. L'inconvénient majeur de cette distance est que sa complexité temporelle est (bi)-quadratique en la taille des deux graphes à évaluer  $O(n^2m^2)$ , même si cette complexité peut-être ramenée à  $O(n^2m + m^2n)$  avec une régularisation entropique.

On notera que les auteurs de *Fused* Gromov-Wasserstein ont également proposé une méthode plus rapide d'évaluation de Gromov-Wasserstein, *Sliced* Gromov-Wasserstein possédant une complexité identique à *Sliced*-Wasserstein. Cependant, cette distance ne peut s'appliquer qu'aux graphes non attribués et n'est définie que pour les graphes de même taille.

**Graph Optimal Transport for graph comparison.** [Maretic et al. 2019] Dans ce modèle, les auteurs proposent de comparer des graphes attribués par l'intermédiaire des signaux (i.e les attributs) qu'ils portent. Ils supposent notamment que ces signaux suivent une distribution gaussienne dont le laplacien du graphe est la matrice de covariance. Ces hypothèses leur permettent alors de définir explicitement la distance de 2-Wasserstein entre les signaux de graphes<sup>11</sup>. Cette distance n'est toutefois valable qu'à la condition que les deux graphes sur lesquels portent les signaux soient de même taille et que la correspondance entre leurs nœuds soit établie. Un schéma d'optimisation pour effectuer la difficile tâche d'alignement (voir Section 2.1.2) est proposé mais il donne à la méthode une complexité quasi-cubique en la taille des graphes comparés. Si la méthode présente des résultats expérimentaux satisfaisants, elle est intrinsèquement limitée par la nécessité que les graphes comparés soient de même taille et par l'hypothèse faite sur la nature des signaux portés par ces graphes qui n'est pas forcément vérifiée.

**Wasserstein Weisfeiler-Lehman Graph Kernels - WWL.** [Togninalli et al. 2019] Ce noyau est construit à partir d'une pseudo-distance entre graphes attribués similaire à la méthode précédente. Les distances entre graphes sont évaluées via la distance de 2-Wasserstein entre les différentes caractéristiques portées par leurs nœuds. Chaque graphe est associé à une distribution uniforme et discrète dont le support est ses caractéristiques. Ces dernières sont calculées avec *Graph Isomorphism Network* – GIN – un réseau de neurones *sans paramètre* entraînable [Xu et al. 2019]; il s'agit d'une version continue de Weisfeiler-Lehman (voir Section 3.2.1); il a été démontré que cette procédure est au moins aussi discriminante que WL [Xu et al. 2019]. La structure des graphes est donc prise en compte à travers les caractéristiques construites par ce

10. L'Equation (A.1) représente le cas particulier de cette distance, où il n'y a pas de pondération entre le poids de la contribution de la structure et des attributs à la distance.

11. Étant sous-entendu que les signaux sur le graphe sont considérés comme des distributions uniformes dont les atomes sont les valeurs prises sur les différents nœuds.

GCN. L'avantage de cette formulation est qu'elle ne nécessite pas d'avoir des graphes de même taille et ne nécessite pas de procéder à l'alignement des nœuds. En revanche, il faut calculer le coûteux transport optimal. On notera que la distance ainsi calculée est une pseudo-distance de graphe. En effet, bien que la distance de 2-Wasserstein soit bien une distance pour les distributions; GIN a les mêmes faiblesses que WL et peut produire éventuellement les mêmes distributions de caractéristiques partant de deux graphes structurellement différents (voir Annexe A.2.1). Il existe donc des graphes  $\mathcal{G}$  et  $\mathcal{G}'$  différents tels que leur distance est nulle selon cette (pseudo)-distance. Malgré tout, on construit avec cette pseudo-distance  $D_{WL}$  un noyau de la forme  $e^{-\lambda D_{WL}}$ , avec  $\lambda > 0$ . Ce noyau s'avère très efficace notamment en classification de graphes sur de nombreux jeux de données (voir Table 3.3 dernière colonne).

Chacune des méthodes évoquées ici, est performante mais nécessite de résoudre des problèmes d'optimisation avec des complexités comprises entre  $O(n^2)$  et  $O(n^3 \log n)$ . Ces complexités importantes font de ces méthodes de mauvaises candidates pour paramétriser une distance en vue de faire de l'apprentissage de métrique sur graphes. L'une de nos contributions principales consiste à proposer une distance paramétrique sur les graphes attribués *différentiable* avec une complexité raisonnable (quasi linéaire en la taille des graphes) (voir Section 5.2.1.c). Notre méthode s'inspirera notamment de Wasserstein Weisfeiler-Lehman.

### 5.3 *Simple Graph Metrics Learning*

Dans cette partie, nous allons présenter nos contributions, en particulier notre modèle SGML et ses propriétés :

- Notre modèle d'apprentissage de métrique repose sur une pseudo-distance paramétrique entre graphes attribués basée sur *Sliced-Wasserstein*. Cette distance (comme la distance de Mahalanobis) est paramétrée par un nombre de paramètres de l'ordre de  $q^2$ , où  $q$  est la dimension des caractéristiques portées par les nœuds des graphes. Ces paramètres constituent l'ensemble des poids entraînaables du modèle, lorsque  $q$  est relativement petit, le nombre de paramètres à apprendre reste donc raisonnable.
- Le calcul de  $SW$  implique l'estimation de l'intégrale (Eq. 5.22) qui intervient dans son expression. Nous proposons d'étudier 3 façons d'estimer cette intégrale, chacune d'entre elles ayant ses avantages suivant le contexte applicatif considéré
- Nous proposons également une nouvelle fonctionnelle d'apprentissage de métrique dérivée de NCMML (*Nearest Class Mean Metric Learning*), qu'on nommera *Nearest Class Cloud Metric Learning* - NCCML ; construite en vue d'améliorer l'algorithme des  $k$  plus proches voisins.
- Par ailleurs, nous proposons également d'étendre la définition de *Projected Sliced Wasserstein* -  $PSW$  - une variation de *Sliced Wasserstein* proposée par Rowland et al. [2019] lorsque les deux distributions comparées ont la même taille  $n = m$ . Nous étendons sa définition au cas  $n \neq m$ .
- Nous proposons également une nouvelle pseudo-distance *Restricted Sliced Wasserstein* -  $RSW$  - sur les graphes, souvent plus rapide à estimer que  $SW$ . Nous définirons également un analogue de cette pseudo-distance pour  $PSW$ , nommé *Restricted Projected Wasserstein* -  $RPSW$ .

En Section 5.3.1, nous présenterons comment nous générons des distributions à partir de graphes attribués, puis en Section 5.3.2, nous détaillerons comment ces distributions nous servent pour établir des pseudo-distances et des mesures de similarités entre graphes, ensuite en Section 5.3.3, nous présenterons une nouvelle fonctionnelle d'apprentissage et pour conclure en Section 5.3.4, nous discuterons de la complexité et des questions d'optimisation de notre modèle.

### 5.3.1 Génération des caractéristiques des graphes

La première étape de notre méthode d'apprentissage consiste en la génération de caractéristiques représentatives de la structure de chaque graphe et des attributs de leurs nœuds. Nous utilisons pour cela SGCN, un GCN qui a pour principale trait de supprimer les non linéarités intermédiaires dans le réseau. Cela lui permet de fortement réduire le nombre de paramètres à entraîner sans pour autant dégrader ses performances par rapport aux autres GCN (voir Section 2.3.5.b). Nous rappelons la formule du calcul des caractéristiques effectué (décrit à l'Equation (2.52) ici :

$$\mathbf{Y} = \text{ReLU}(\tilde{\mathbf{A}}^p \mathbf{X} \Theta) \quad (5.23)$$

Où  $\mathbf{X} \in \mathbb{R}^{n \times q}$  sont les attributs initiaux des nœuds et  $\mathbf{Y} \in \mathbb{R}^{n \times q'}$  les caractéristiques calculées par SGCN. La profondeur d'exploration de voisinage  $p$  de ce GCN est un des hyperparamètres de la méthode avec la dimension  $q'$  des caractéristiques extraites. Les coefficients de la matrice  $\Theta \in \mathbb{R}^{q \times q'}$  de ce GCN sont les seules poids entraînaibles de la méthode. On choisira toujours  $q' \leq q$ , la méthode a donc au plus  $q^2$  paramètres. Ce GCN a été choisi pour sa simplicité couplée à ses performances comparables à l'état de l'art des GCN. Ainsi, on définit alors la distribution uniforme dont ces caractéristiques forment le support :

$$\mathcal{D}_\Theta(\mathcal{G}, \mathbf{X}) = \sum_{i=1}^n \frac{1}{n} \delta_{\mathbf{y}_i} \quad (5.24)$$

Cette première étape est très similaire à celle de WWL (Section 5.2.3 - 3e paragraphe) à la différence que nous considérons un GCN et donc des descripteurs entraînaibles.

### 5.3.2 Distance de *Sliced-Wasserstein* appliquée aux graphes

Nous avons choisi *Sliced-Wasserstein* pour évaluer la distance entre les distributions (5.24) générées par chacun des graphes attribués, car il s'agit de l'alternative à la distance de 2-Wasserstein, avec la complexité temporelle la plus faible. Étant donné deux graphes  $\mathcal{G}$ ,  $\mathcal{G}'$ , respectivement attribués par  $\mathbf{X}$  et  $\mathbf{X}'$ , alors la distance entraînable de notre modèle est définie par :

$$d_\Theta^{\text{SW}}(\mathcal{G}, \mathcal{G}') = \text{SW}_2(\mathcal{D}_\Theta(\mathcal{G}, \mathbf{X}), \mathcal{D}_\Theta(\mathcal{G}', \mathbf{X}')) \quad (5.25)$$

Soit :

$$d_\Theta^{\text{SW}}(\mathcal{G}, \mathcal{G}')^2 = \int_{\mathbb{S}^{q'-1}} \sum_{i,j=1}^{n,n'} \pi_{i,j}^{\theta,*} \|\mathbf{y}_i \cdot \boldsymbol{\theta} - \mathbf{y}'_j \cdot \boldsymbol{\theta}\|_2^2 d\boldsymbol{\theta} \quad (5.26)$$

C'est une pseudo-distance car comme WL et GIN, SGCN peut produire des caractéristiques similaires pour des graphes différents. Par ailleurs, on note  $\pi^{\theta,*}$  le plan de transport optimal entre les distributions  $\mathcal{D}_\Theta(\mathcal{G}, \mathbf{X})$  et  $\mathcal{D}_\Theta(\mathcal{G}', \mathbf{X}')$  projetées sur le vecteur  $\theta$ . La principale difficulté pour estimer cette pseudo-distance est le calcul de l'intégrale.

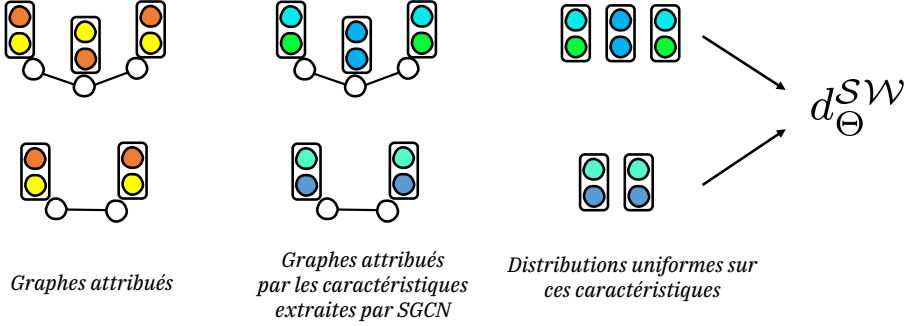


FIGURE 5.4 – *Sliced-Wasserstein appliqué aux graphes attribués*. Dans cette illustration  $q = q' = 2$ , les couleurs représentent des scalaires et les paires de couleurs des vecteurs de dimension 2. La distance de  $SW$  est appliquée entre les distributions discrètes et uniformes dont le support sont les caractéristiques extraites par SGCN. Cette distance sert de mesure de similarité entre les deux graphes.

### 5.3.2.a Estimation de l'intégrale

L'estimation d'intégrale telle que définie ci-dessus (5.26) se fait généralement via la méthode de Monte-Carlo la plus simple, c'est-à-dire en tirant  $M$  angles  $\{\theta_m\}_{m=1}^M \sim \mathcal{U}(\mathbb{S}^{q'-1})$  uniformément sur l'hypersphère  $\mathbb{S}^{q'-1}$  :

$$d_{\Theta}^{SW}(\mathcal{G}, \mathcal{G}')^2 \approx \frac{1}{M} \sum_{\theta_m \in \{\theta_m\}_{m=1}^M} \sum_{i,j=1}^{n,n'} \pi_{i,j}^{\theta_m,*} \|\mathbf{y}_i \cdot \theta_m - \mathbf{y}'_j \cdot \theta_m\|_2^2 \quad (5.27)$$

Il s'agit d'un estimateur non biaisé de la valeur de l'intégrale<sup>12</sup>, on peut aisément démontrer avec le théorème centrale limite que le taux de convergence est  $O(\frac{1}{\sqrt{M}})$ . S'il présente l'avantage d'être indépendant de la dimension  $q'$ , il est relativement lent. Pour l'accélérer, on peut envisager plusieurs approches<sup>13</sup> :

- **Les processus stratifiés de Monte-Carlo (Ortho-MC)**. Ces processus subdivisent l'espace d'intégration en sous-ensembles disjoints puis y échantillonnent des points (i.e nos  $\theta$ ); sous certaines conditions [Fakhereddine 2013]. Ces choix de points permettent d'avoir un taux de convergence plus rapide que l'estimateur simple de Monte-Carlo (5.27).

12. Plus précisément, c'est un estimateur de la moyenne de l'intégrande sur  $\mathbb{S}^{q'-1}$ , pour obtenir une estimation de sa valeur il suffit de multiplier cet estimateur par le *volume* de  $\mathbb{S}^{q'-1}$ .

13. Les processus déterminantaux sont également une piste d'accélération de la convergence, mais le gain de taux de convergence par rapport à Monte-Carlo classique devient rapidement négligeable lorsque la valeur de  $q$  croît (voir Annexe A.4.2).

Rowland et al. [2019] proposent d’approximer ce processus de stratification en tirant des directions partiellement orthogonales les unes des autres, tels que :

$$\{\boldsymbol{\theta}_m\}_{m=1}^M = \{\{\boldsymbol{\theta}_{m1}\}_{m_1=1}^{q'} \cup \dots \cup \{\boldsymbol{\theta}_{ms+1}\}_{m_{s+1}=1}^{r \leq q'}\}$$

avec  $M = q's + r$ <sup>14</sup>, où pour  $k \in \llbracket 1, s + 1 \rrbracket$ ,  $\{\boldsymbol{\theta}_{mk}\}_{m_k=1}^{q'} \sim \mathcal{U}(\mathcal{O}_{q'}(\mathbb{R}))$ . C’est-à-dire que la matrice  $[\boldsymbol{\theta}_{mk}]_{m_k=1}^{q'}$  est une matrice orthogonale échantillonnée selon la distribution uniforme dans l’ensemble des matrices orthogonales :  $\mathcal{O}_{q'}(\mathbb{R})$ <sup>15</sup>. Dans cet article, les auteurs montrent que dans le cas restreint où  $n = n' = 2$  (les tailles des graphes comparés) et  $q' = 2$  (la dimension des caractéristiques extraites), cette façon de tirer les directions ne conduit pas nécessairement à une amélioration de la convergence de l’estimateur de  $SW$ . Toutefois, les hypothèses pour lesquelles ce résultat est démontré sont relativement éloignées de ce qu’on peut rencontrer dans les jeux de données réels ; d’ailleurs pour des valeurs supérieures de  $q'$ , les auteurs observent une nette amélioration de la convergence.

- **Les processus de Quasi Monte-Carlo (Q-MC).** [Morokoff and Caffisch 1995] Dans ces processus, les points tirés pour l’estimation de l’intégrale sont les points de suites à discrédances<sup>16</sup> faibles dans l’espace d’intégration. Intuitivement, les suites finies à discrédances faibles sur un ensemble  $\mathbb{X}$  sont des suites telles que pour tout sous-ensemble  $\mathbb{Q} \subseteq \mathbb{X}$  quelconques, la proportion des points de la suite dans cet ensemble est proche de la mesure de  $\mathbb{Q}$  (voir Annexe A.4.3 pour plus de détails). Ces processus sont connus pour permettre une convergence allant jusqu’à  $O(\frac{\log(M)^{q'-1}}{M})$ <sup>17</sup>. A priori, pour avoir une convergence plus rapide que celle de la méthode de Monte-Carlo classique, il faut choisir  $M$  au moins de l’ordre d’une exponentiation de la dimension  $q'$ . Toutefois, on peut constater empiriquement des convergences plus rapides avec des valeurs de  $M$  plus faibles car le taux de convergence dépend également de la manière dont varie la fonction, ce qui est en général difficile à quantifier (voir Annexe A.4.3 pour plus de détails), d’où l’intérêt des expériences qui vont suivre.

Nous avons évalué expérimentalement ces 2 façons de procéder comparativement à la méthode de Monte-Carlo classique pour quelques jeux de données usuels rencontrés en apprentissage sur graphes. Pour la méthode de quasi-Monte-Carlo, nous nous sommes appuyés sur la suite de Hammersley comme dans l’article de Rehman and Mandic [2010], les détails de la procédure d’estimation peuvent être trouvés en Annexe A.4.3. Les résultats de ces expériences peuvent être consultés table 5.1. Ces expériences sont faites à  $M = 50$  fixé, une valeur qu’on retrouve couramment dans la littérature. Elles montrent que le processus qui permet d’estimer le plus efficacement et de manière stable est le processus orthogonal, c’est donc ce dernier que nous avons choisi dans toutes nos expériences. Nous tirerons toujours  $M = 50$  échantillons. On notera que la méthode de Monte-Carlo simple est tout de même très efficace pour ce problème.

**Remarque 25 (Quasi Monte-Carlo.)** *Des expériences supplémentaires, non reportées ici nous ont montré que l’estimateur de Quasi Monte-Carlo est en fait le processus le plus ef-*

14. Il s’agit d’une division euclidienne de quotient  $s$  et de reste  $r$ .

15. Au sens de la mesure de Haar.

16. Synonyme de divergence (ou disconvenance).

17. Lorsque la suite en question est la suite de Hammersley [Drmotá and Tichý 2006].

Echantillonnage. Jeux de données.	MC		Q-MC		Ortho-MC	
	Moy.	Var.	Moy.	Var.	Moy.	Var.
MUTAG(d)	0.9962	$2.6 \cdot 10^{-3}$	<b>0.99949</b>	<b><math>0.2 \cdot 10^{-4}</math></b>	0.9985	$0.4 \cdot 10^{-3}$
NCI109(d)	0.9969	$3.0 \cdot 10^{-3}$	0.9919	<b><math>0.1 \cdot 10^{-3}</math></b>	<b>0.9983</b>	$0.4 \cdot 10^{-3}$
ENZYMES(d)	0.9984	$3.6 \cdot 10^{-3}$	1.0249	$0.3 \cdot 10^{-3}$	<b>0.9984</b>	<b><math>0.6 \cdot 10^{-3}</math></b>
ENZYMES(c)	0.9973	$5.0 \cdot 10^{-3}$	1.1672	$2.07 \cdot 10^{-2}$	<b>0.9995</b>	<b><math>0.6 \cdot 10^{-3}</math></b>

TABLE 5.1 – *Expériences d’estimation de SW*. Les expériences ont été conduites pour  $M = 50$ , une valeur classique dans la littérature. Pour chaque jeu de données testés, (d) signifie que les caractéristiques des nœuds sont les degrés et (c) qu’il s’agit de caractéristiques continues. MC désigne la méthode classique d’échantillonnage, Q-MC la méthode de quasi Monte-Carlo et Ortho-MC la méthode de Monte-Carlo stratifiée avec estimateur « orthogonal ». Pour chaque graphe de chaque jeu de données, on produit une distribution selon les modalités décrites en Section 5.3.1 ; avec  $p = 2$  et  $\Theta$  pris aléatoirement (selon la procédure uniforme de Glorot). Muni de ces distributions, on évalue  $SW$  pour 400 paires de ces distributions. Le résultat est normalisé par un résultat de référence qui est la valeur estimée de l’intégrale par un Monte-Carlo classique avec  $M = 10000$ . La colonne *Moy.* donne le résultat moyen de l’estimateur sur ces 400 paires. Plus le nombre est proche de 1, plus l’estimation est bonne. La colonne *Var.* indique la variance de l’estimateur sur les 400 paires. Ce tableau montre que la méthode qui a les meilleurs résultats (et ce de manière stable) est l’estimateur orthogonal.

ficace mais il nécessite de choisir une plus grande valeur de  $M$ . Par ailleurs, cette valeur de  $M$  minimale pour atteindre ces performances est variable selon les jeux de données et n’en fait donc pas une méthode extrêmement fiable pour notre usage.

**Remarque 26 (Complexité temporelles.)** En notant  $n$  et  $m$  la taille des deux graphes dont on veut déterminer la distance, en théorie la complexité est donnée par  $O(M(n + m + n \log(n) + m \log(m)))$ . Le premier terme est dû au calcul du plan de transport (dans le pire cas), aux calculs des coûts associées aux coefficients du plan de transport non nuls et à son application. Les termes suivants correspondent à l’opération de tri des distributions projetées. Ces opérations sont effectuées  $M$  fois.

### 5.3.2.b Restricted Sliced-Wasserstein

Dans l’optique de réduire la complexité de calcul du modèle, nous proposons une version de  $d^{SW}$  plus simple à calculer. Cette version consiste à appliquer la distance de  $d^{SW}$  en considérant seulement une seule composante du signal  $\mathbf{Y}$ , et d’effectuer ce calcul pour chaque composante puis de faire la moyenne de ces distances calculées. Mathématiquement, cela revient exactement dans (5.29) à choisir comme poids d’évaluation  $\{\theta_m\}_{m=1}^M = \{e_m\}_{m=1}^{q'}$  les vecteurs de la base canonique  $\mathbb{R}^{q'}$  :

$$d_{\Theta}^{RSW}(\mathcal{G}, \mathcal{G}')^2 = \frac{1}{q'} \sum_{e_m \in \{e_m\}_{m=1}^{q'}} \sum_{i,j=1}^{n,n'} \pi_{i,j}^{e_m,*} \|\mathbf{y}_i \cdot e_m - \mathbf{y}'_j \cdot e_m\|_2^2 \quad (5.28)$$

La quantité ainsi définie *Restricted Sliced-Wasserstein* est bien une pseudo-distance entre graphes ; elle hérite des propriétés des  $q'$  pseudo-distances qui la composent. Elle distingue les mêmes graphes attribués que  $d^{SW}$  à l'exception des graphes pour lesquels SGCN génère des distributions égales à des permutations près indépendantes de leurs composantes<sup>18</sup>. En contrepartie de cette moindre sensibilité, la complexité tombe à  $O(q'(n + m + n \log(n) + m \log(m)))$  elle est donc meilleure dès lors que  $q' < 50$ , ce qui est en général le cas.

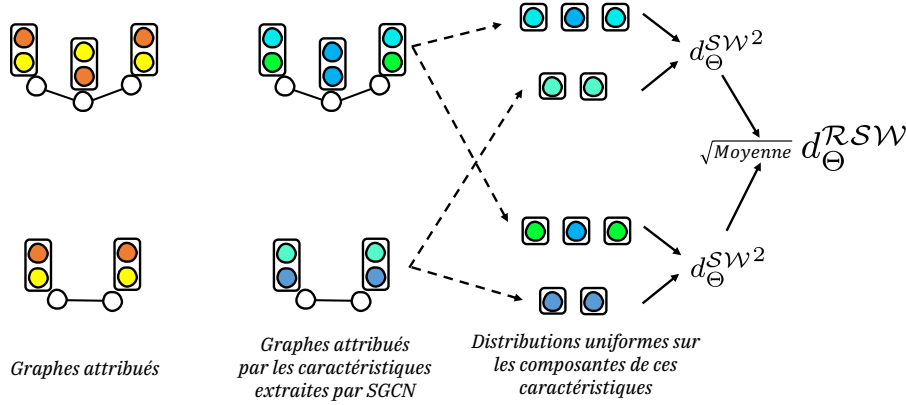


FIGURE 5.5 – *Restricted Sliced-Wasserstein appliqué aux graphes attribués*. On calcule la distance de  $SW$  pour les deux paires de distributions (une par composante des caractéristiques extraites par SGCN) discrètes uniformes dont le support sont les composantes de ces caractéristiques. On moyenne ensuite les carrés de ces distances, la racine de cette moyenne servira de mesure de similarité entre les deux graphes.

### 5.3.2.c Projected Sliced-Wasserstein

*Projected Sliced-Wasserstein* est une méthode proposée par Rowland et al. [2019], en réponse à certaines limitations de *Sliced-Wasserstein*. L'argument principal avancé par les auteurs est que l'estimation (5.27) est biaisée par rapport à Wasserstein, car les  $\theta$  par l'intermédiaire de la projection, déterminent les plans transports optimaux mais également le coût du transport puisqu'ils interviennent dans son expression et y introduisent un biais. Ils ont notamment tendance à sous-estimer la distance entre les distributions. Ils préconisent donc de procéder de manière similaire, mais d'évaluer le coût dans l'espace d'origine des distributions projetées. Toutefois, ils ne considèrent que les cas où les distributions ont les mêmes tailles, nous étendons donc cette définition, dans le cas où les distributions n'ont pas la même taille, ce qui s'écrit :

$$d_{\Theta}^{PSW}(\mathcal{G}, \mathcal{G}')^2 \approx \frac{1}{M} \sum_{\theta_m \in \{\theta_m\}_{m=1}^M} \sum_{i,j=1}^{n,n'} \pi_{i,j}^{\theta_m,*} \|\mathbf{y}_i - \mathbf{y}'_j\|_2^2 \quad (5.29)$$

Lorsque  $n = m$ , il est acquis que  $PSW$  soit une distance sur  $\mathcal{P}(\mathbb{R}^{q'})$ . Dans le cas  $n \neq m$ , la question est plus difficile notamment du fait qu'établir l'inégalité triangulaire n'est pas triviale.

18. C'est donc aussi une pseudo-distance pour les distributions.



Nous considérerons donc cette grandeur comme une mesure de similarité sur graphe empirique. En terme de complexité,  $\mathcal{PSW}$  est un peu plus coûteux que  $\mathcal{SW}$  du fait de l'évaluation des coûts dans l'espace d'origine  $\mathbb{R}^{q'}$ , sa complexité est  $O(M(q' * (n + m) + n \log(n) + m \log(m)))$ . Nous proposons donc également une version *Restricted* plus rapide à évaluer, qu'on nommera *Restricted Sliced-Wasserstein -  $\mathcal{RPSW}$* .

$$d_{\Theta}^{\mathcal{RPSW}}(\mathcal{G}, \mathcal{G}')^2 = \frac{1}{q'} \sum_{\mathbf{e}_m \in \{\mathbf{e}_m\}_{m=1}^{q'}} \sum_{i,j=1}^{n,n'} \pi_{i,j}^{\mathbf{e}_m,*} \|\mathbf{y}_i - \mathbf{y}_j\|_2^2 \quad (5.30)$$

C'est aussi une mesure de similarité. Sa complexité est  $O(q'(q'(n + m) + n \log(n) + m \log(m)))$ .

### 5.3.3 Fonctionnelle d'apprentissage : *Nearest Class Cloud Metric Learning*

Pour se comparer aux distances entre graphes de la littérature, nous souhaitons évaluer notre distance avec la méthode des  $k$  plus proches voisins, qui constitue généralement une bonne façon d'évaluer une distance. Nous avons donc testé plusieurs fonctionnelles présentant de bonnes performances pour cette tâche, notamment NCA et LMNN présentés en Section 5.2.1.c.

Nous proposons également une fonctionnelle originale dérivée de NCMML également présentée en Section 5.2.1.c. Cette méthode construite pour améliorer la méthode des plus proches barycentres nécessite le calcul des barycentres des points de même classe. Cette opération est tout à fait possible avec  $d^{\mathcal{SW}}$  [Bonneel et al. 2015]. Toutefois, il nous a paru plus intéressant de proposer un modèle légèrement différent pour que le problème soit plus axé vers l'amélioration de la méthode des  $k$  plus proches voisins. En remarquant que pour la norme 2 (en conservant les notations de la Section 5.2.1.c), la distance d'un point à un barycentre est majorée (à une constante près) par la distance de ce point aux nuages de points<sup>19</sup> de même classe :

$$\|\mathbf{N}\mathbf{x}_i - \mathbf{N}_{e_l}\|_2^2 = \|\mathbf{N}(\mathbf{x}_i - \frac{1}{|\mathcal{E}^{-1}(e_l)|} \sum_{\substack{x_j \\ \mathcal{E}(x_j)=e_l}} \mathbf{x}_j)\|_2^2 \leq \frac{1}{|\mathcal{E}^{-1}(e_l)|^2} \sum_{\substack{x_j \\ \mathcal{E}(x_j)=e_l}} \|\mathbf{N}(\mathbf{x}_i - \mathbf{x}_j)\|_2^2 \quad (5.31)$$

Où  $|\mathcal{E}^{-1}(e_l)|$  est le nombre d'éléments qui ont l'étiquette  $e_l$ . On peut constater que la distance moyenne aux points du nuage peut être très grande sans pourtant impacter la distance au barycentre, NCMML peut favoriser des situations trompeuses pour la méthode des  $k$  plus proches voisins notamment lorsque les classes ne sont pas regroupées autour de leur moyenne. Nous proposons donc un modèle dans lequel la probabilité d'être étiqueté par  $e$  pour un graphe  $\mathcal{G}$  est fonction de la distance aux nuages de points d'une classe (d'où le nom de la méthode) et non pas de son barycentre :

$$p^{\Theta}(e|\mathcal{G}) = \frac{\exp\left(\sum_{\substack{\mathcal{G}_i \in \mathbb{G}_x \\ \mathcal{E}(\mathcal{G}_i)=e}} -d_{\Theta}^{\mathcal{SW}}(\mathcal{G}, \mathcal{G}_i)^2\right)}{\sum_{e' \in \mathbb{E}} \exp\left(\sum_{\substack{\mathcal{G}_i \in \mathbb{G}_x \\ \mathcal{E}(\mathcal{G}_i)=e'}} -d_{\Theta}^{\mathcal{SW}}(\mathcal{G}, \mathcal{G}_i)^2\right)} \quad (5.32)$$

19. Ou ensembles de points.

On souhaite alors construire la distance  $d_{\Theta}^{SW}$  qui va maximiser la probabilité que les graphes étiquetés à notre disposition aient bien les bonnes étiquettes :

$$\max_{\Theta} \sum_{\mathcal{G}_i \in \mathbb{G}_x} \log p^{\Theta}(\mathcal{E}(\mathcal{G}_i) | \mathcal{G}_i) \quad (5.33)$$

Cette formulation qu'on nommera *Nearest Class Cloud Metric Learning* – NCCML – favorise la méthode des  $k$  plus proches voisins. On a constaté empiriquement que dans notre cadre de travail, cette fonctionnelle fonctionnait mieux que NCA et LMNN.

### 5.3.4 Complexité et optimisation

**Optimisation.** En termes d'optimisation, on peut dériver directement à travers les différentes pseudo-distances/mesures de similarités définies dès lors que les plans de transports sont connus. Les plans de transport sont strictement déterminés par la taille des distributions (uniformes) dès lors que les supports de ces distributions sont triés. On peut donc pré-calculer tous les plans de transport optimaux utiles du jeu de données en amont. Les techniques d'auto différentiation peuvent être utilisées sur ces expressions (voir [Peyré and Cuturi 2020] pour des détails sur ces questions de différentiabilité sur la distance de Wasserstein).

La minimisation des fonctionnelles considérées est effectuée par *batch* et descente de gradient stochastique (en particulier avec l'optimiseur *Adam*). Ce choix de processus d'optimisation présente plusieurs avantages. Ils permettent d'une part à la méthode d'être linéaire en le nombre de graphes. Et d'autre part, comme aucune des fonctionnelles d'AM utilisées ici n'est convexe, l'optimisation par gradient stochastique permet d'éviter certains minimums locaux.

**Complexité temporelle.** Si le calcul de la distance de *Sliced-Wasserstein* est théoriquement donné par  $O(n+m+n \log n+m \log m)$  dans les faits en réorganisant les opérations effectuées, on peut vectoriser une partie du calcul et gagner un temps significatif sur GPU. L'implémentation que nous avons faite a donc en théorie une complexité moins bonne  $O(n*m+n \log n+m \log m)$  mais est expérimentalement plus efficace. En tenant compte de ce fait et que la complexité temporelle de notre modèle est dominé par le calcul des distances, en notant  $E$  le nombre d'*epochs*,  $B$  la taille du *batch* et  $\tilde{n}$  le nombre de nœuds moyens d'un graphe, la complexité *dominante* totale pour  $SW$ ,  $RSW$ ,  $PSW$  et  $RPSW$  est donnée respectivement par  $O(|\mathbb{G}_x|EB^2M\tilde{n}^2)$ ,  $O(|\mathbb{G}_x|EB^2q'\tilde{n}^2)$ ,  $O(|\mathbb{G}_x|EB^2Mq'\tilde{n}^2)$  et  $O(|\mathbb{G}_x|EB^2q'^2\tilde{n}^2)$ . A titre d'exemple, pour  $B = 8$ ,  $E = 10$ ,  $M = 50$ ,  $q' = 5$  sur un petit jeu de données comme MUTAG (constitué de 188 graphes), l'entraînement prend moins d'une dizaine de secondes, et le calcul de toute les paires de distances également. Sur un jeu de données plus grand (constitué de 4127 graphes) comme NCI109 le temps d'entraînement est approximativement de 7 minutes tandis que le calcul de toutes les paires de distances possibles prend à peu près 30 minutes. Ces tests ont été faits en utilisant comme caractéristiques des nœuds un encodage *one-hot* du degrés des nœuds (voir Section 3.4.1 et Table 3.2 pour des informations sur ces jeux de données).

**Complexité Spatiale.** Pour l'implémentation sur GPU, le coût en mémoire est dominé par une matrice contenant tous les coûts de transport entre les caractéristiques de tous les graphes

du *batch*, ce qui correspond pour pour  $SW$ ,  $RSW$ ,  $PSW$  et  $RPSW$  respectivement à une complexité en taille de  $O(MB^2n_b^2)$ ,  $O(q'B^2n_b^2)$ ,  $O(MB^2n_b^2)$  et  $O(q'B^2n_b^2)$ , où  $n_b$  est la taille du plus grand graphe du *batch* courant  $B$ .

## 5.4 Expériences

Les expériences que nous avons mené sur ce modèle n'ont pour le moment été complétées que pour un seul jeu de données : MUTAG (voir Fig. (4.1)), car pour chaque jeu de données et chaque combinaison d'hyperparamètres, les expériences nécessitent d'être reproduites une dizaine de fois pour en moyenner les résultats. Si le temps d'entraînement est relativement rapide, il faut cependant à chaque fois calculer les distances entre toutes les paires de graphes, ce qui prend un certain temps notamment lorsque le nombre de graphes est important (comme pour NCI109). L'implémentation de la méthode peut être trouvée ici : <https://github.com/Yacnmm/SGML>.

**Hyperparamètres.** Dans ces expériences  $q' = q$  est fixé et des expériences préliminaires nous ont amené à fixer la taille du *batch* à 8. Le *learning rate* est fixé à  $l_r = 0.999 \cdot 10^{-2}$  l'application ou non d'un *learning rate* décroissant selon la loi  $l_r = 0.999 \cdot 10^{-2} * 1.1^{-50 \frac{e}{E}}$ , où  $e$  est l'*epoch* courante sera soumis à validation avec les autres paramètres nommés dans la description des expériences.

### 5.4.1 Classification

Dans cette expérience nous avons évalué notre méthode selon deux modalités : en construisant un noyau avec les distances/mesures similarités apprises ou en utilisant la méthode des  $k$  plus proches voisins directement sur les distances calculées et ce pour chacune des distances introduites. Pour ce qui est de la comparaison de nos résultats, pour la première modalité, nous nous comparerons à WWL (Wasserstein Weisfeiler-Lehman) [Togninalli et al. 2019] le noyau dont nous nous sommes inspirés, et à  $\mathcal{FGW}$  (*Fused Gromov-Wasserstein*) Titouan et al. [2019] (voir Section 5.2.3) une distance performante s'appuyant sur le transport optimal. Quant à la seconde modalité, nous nous comparerons notamment à NetLSD [Tsitsulin et al. 2018], une pseudo-distance de faible complexité linéaire en la taille des nœuds et opérant sans attributs (autres que les degrés).

### 5.4.2 Noyau

Pour chacune des distances définies, nous avons procédé à l'apprentissage selon la fonctionnelle de NCCML par un apprentissage par *batch*. Ces expériences ont été faites en s'entraînant sur 20% et sur 90% des graphes étiquetés. Lorsque la pseudo-distance/mesure de similarité  $D$  est calculée, un noyau  $e^{-\lambda D}$  est alors construit à partir de cette dernière. Les paramètres  $\lambda$ <sup>20</sup>,  $p = \{0, 2, 3\}$  et le nombre d'*epochs*  $E = \{10, 20, 30\}$  sont décidés par validation croisée sur 90% de graphes étiquetés, suivant une recherche par grille. Cette valeur de validation est moyennée sur 10 exécutions de l'algorithme. Le paramétrage à l'issue de ce processus présentant les

20. L'ensemble des valeurs de  $\lambda$  possibles sont 6 valeurs espacées de manière régulières entre  $10^{-4}$  et  $10^1$ .

Méthode	$SW$	$\mathcal{R}SW$	$\mathcal{P}SW$	$\mathcal{P}\mathcal{R}SW$	Fused Wass.	WWL
<b>MUTAG</b>	<b>0.900</b> (0.873)	0.884 (0.890)	<b>0.900</b> ( <b>0.900</b> )	<b>0.900</b> (0.884)	0.884	0.873

TABLE 5.2 – *Résultats des expériences de classification avec un noyau.* On peut notamment remarquer sur cet exemple limité, que les méthodes restreintes aux vecteurs de base ont des performances similaires aux autres méthodes. L’adaptation permet d’obtenir des résultats marginalement supérieurs. Entre parenthèses, les résultats pour un entraînement sur 20% des données au lieu de 90%. Les degrés sont utilisés comme caractéristiques initiales des nœuds ( $q = 4$ ).

meilleurs résultats est alors entraîné sur les 90% de graphes qui ont servis à calculer la validation et le modèle est ensuite testé sur les 10% restants. Le résultat est moyenné sur 10 exécutions en plus des 10 exécutions de la partie en validation. Lorsque SGML est entraîné sur 90% des données, ce sont ces mêmes 90% de données qui sont utilisées pour procéder à la validation et à l’entraînement final. Lorsque SGML n’est entraîné que sur 20% des données, on s’assure que ces 20% soient dans les 90% de données qui sont utilisées pour procéder à la validation et à l’entraînement final. De sorte que les étiquettes de l’ensemble de tests n’ont jamais été vues ni pendant l’entraînement ni pendant la validation. Les résultats de cette expérience peuvent être retrouvés en table 5.2 pour l’entraînement partiel sur 90% des données et entre parenthèses sur l’entraînement partiel sur 20% des données.

Compte tenu du fait que les résultats ne portent que sur un seul jeu de données, il est difficile de conclure mais ces résultats sont prometteurs notamment vis-à-vis de la complexité de notre méthode.

**Remarque 27 (NCA et LMNN)** *Nous avons comparé NCA, LMNN et NCCML dans ce cadre limité et la NCCML a présenté les meilleurs résultats que nous avons donc reporté table 5.2 et 5.3. Au terme des expériences, ces fonctionnelles seront comparées sur tous les jeux de données. On notera que dans ce cas, LMNN est entraîné avec 3 cibles mises à jour à chaque batch. Les cibles sont choisies dans ce dernier, en prenant les voisins de même étiquette les plus proches pour la distance courante.*

### 5.4.3 Méthodes des plus proches voisins

L’expérience décrite à la section précédente est strictement reproduite, à la différence qu’on utilise la méthode des  $k$  plus proches voisins directement sur les pseudo-distances/mesures de similarités pour réaliser la classification. La même méthode de validation des paramètres est employée, à la différence près qu’en lieu et place du paramètre  $\lambda$  du noyau, on soumet à la validation la grandeur  $k = \{1, 2, 3, 5, 7\}$  qui indique le nombre de voisins utilisés pour réaliser la prédiction. Les résultats peuvent-être trouvés en table 5.3. On peut également observer en Figure 5.6 une visualisation en 2 dimensions des résultats obtenus après application de la t-SNE aux distances produites d’une part par WWL et par  $SW$ .

De même que précédemment, les résultats sont prometteurs mais trop limités pour conclure.

Méthode	$SW$	$\mathcal{R}SW$	$\mathcal{P}SW$	$\mathcal{P}\mathcal{R}SW$	Net-LSD
MUTAG	0.858 (0.858)	0.858 (0.863)	<b>0.890</b> <b>(0.895)</b>	0.868 (0.863)	0.865

TABLE 5.3 – *Résultats des expériences de classification avec la méthode des  $k$  plus proches voisins*. Les constats sont identiques qu’avec le noyau. L’article [Tsitsulin et al. 2018] dans lequel Net-LSD est présenté, propose 6 variantes de la distance, nous avons pris le résultat de la meilleure variante sur MUTAG. De nouveau, entre parenthèses les résultats pour un entraînement sur 20% des données au lieu de 90%. Les degrés sont utilisés comme caractéristiques initiales des nœuds ( $q = 4$ ).

## 5.5 Conclusion

En réponse à une lacune de la littérature, nous avons proposé une distance paramétrée et entraînable permettant d’être intégrée à un modèle d’apprentissage de distance entre graphes, avec une complexité raisonnable. S’il existe de nombreuses distances et autres mesures de similarités sur graphes, très peu sont en mesure d’être intégrées à un modèle d’apprentissage de métrique et ainsi de s’adapter à une tâche. Nous avons également proposé une nouvelle fonctionnelle pour faire de l’apprentissage de métrique que nous souhaitons évaluer de manière plus intensive dans la suite. Nous comptons procéder à des expériences de classification sur un plus grand nombre de jeux de données, mais également procéder à d’autres tâches comme la régression par exemple, en adaptant la fonctionnelle d’apprentissage de métrique. Le prochain chapitre sera dédié à la conclusion de ce manuscrit et à la présentation des perspectives générales sur l’apprentissage sur graphes.

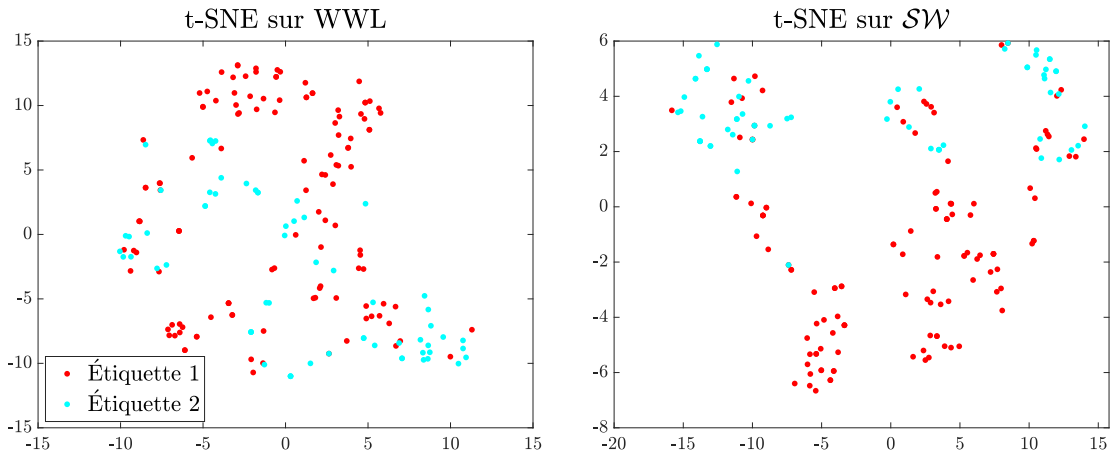


FIGURE 5.6 – *Représentation de métriques sur MUTAG à l'aide de la t-SNE.* **A gauche**, on a généré les différentes paires de distances selon les modalités de l'article de WWL, en utilisant notamment les degrés comme attribut des nœuds. Puis, nous avons appliqué la méthode t-SNE pour visualiser ces distances en dimension 2 [Van der Maaten and Hinton 2008]. Chaque couleur représente une des deux classes du jeu de données. **A droite**, on a pris l'une des exécutions de notre méthode pour  $\mathcal{SW}$ , en choisissant aléatoirement cette exécution parmi les paramètres présentant une valeur de classification de test supérieur à 85% pour la tâche de *clustering* avec la méthode des  $k$ -moyennes. Le rapport des distances moyennes entre les graphes de même étiquettes sur les distances moyennes entre les graphes d'étiquettes différentes est de 0.95 pour WWL contre 0.91 pour  $\mathcal{SW}$ . Ce qui montre que notre méthode a bien rapproché (resp. éloigné) les graphes de même classe (resp. de classes différentes) par rapport à la distance de WWL.



# Chapitre 6

## Conclusion

Comme longuement relaté dans le Chapitre 2, les tâches d'apprentissage sur graphes sont en général difficiles. Dans cette thèse, nous avons voulu proposer plusieurs méthodes d'apprentissage automatique pour faciliter ces tâches. Nous nous sommes notamment appuyés sur les GCN. A ce titre, nos travaux se sont inscrits dans la continuité de la communauté de l'apprentissage automatique qui a porté beaucoup d'attention aux GCN et plus généralement aux GNN ces dernières années. Néanmoins, de nombreuses tâches d'apprentissage sur graphes restent aujourd'hui peu développées. La résolution des défis que posent ces tâches nécessitera peut-être le développement de nouveaux outils en complément voire en remplacement des GCN. Dans cet ultime chapitre, nous résumerons dans un premier temps brièvement les différentes contributions de cette thèse puis dans un second temps, nous discuterons des méthodes et des thématiques qui sont susceptibles de dessiner le futur de l'apprentissage sur graphes.

### 6.1 Résumé des contributions

Un travail sur des questions d'optimisations non exposées dans ce manuscrit a été publié [Kaloga et al. 2019] durant cette thèse, en conclusion de travaux initiés en stage de M2 (voir Section 1.3).

Autrement, dans le Chapitre 3, nous avons proposé *Hierarchical Graph2Vec* – HG2V – [Béthune et al. 2020a,b] un modèle hiérarchique de représentation de graphes attribués se basant sur un modèle de représentation de mots issus du traitement du langage naturel. Sa construction hiérarchique lui permet de construire des représentations de graphes très informatives sur les structures présentes dans le graphe à différentes échelles, et de fait relativement « générales ». Ces représentations peuvent alors être utilisées par des méthodes d'apprentissage classiques opérant sur des données euclidiennes. La complexité du modèle est linéaire en la taille des données, cette propriété couplée à son caractère inductif lui permet d'être utilisable sur de larges jeux de données. Par ailleurs, la différentiabilité du modèle lui permet également de pouvoir s'intégrer au sein de modèles plus grands et de construire des représentations spécifiquement adaptées pour résoudre une tâche.

Dans le Chapitre 4, nous avons proposé *Multiview Graph Canonical Correlation Analysis* -



MVGCCA - [Kaloga et al. 2021a,b] un modèle de représentation de nœuds de graphes multi-attribués. Il se base sur une interprétation probabiliste de la CCA et sur des auto-encodeurs variationnels. A l’instar de HG2V, les représentations ainsi apprises peuvent alors être utilisées par des méthodes d’apprentissage *classiques* notamment pour faire du *clustering*, de la classification ou encore de la visualisation de nœuds. Grâce à sa formulation variationnelle, MVGCCA a la particularité d’être robuste aux données corrompues ; et notamment de pouvoir les régénérer lorsque le nombre de données d’entraînement est suffisant. Cette propriété de robustesse peut également être renforcée via la procédure de *views dropout* que nous avons introduite. Par ailleurs, ce modèle a une complexité linéaire en le nombre de nœuds et peut donc traiter des graphes de grande taille.

Dans le Chapitre 5, nous avons proposé *Simple Graph Metrics Learning* – SGML - [Kaloga et al. 2021c] un modèle d’apprentissage de métrique pour graphes basé sur des éléments de la théorie du transport optimal. Ce modèle est construit notamment dans le but de définir des métriques exploitant des informations sur les graphes (comme des étiquettes) en vue de pouvoir ensuite appliquer des algorithmes plus simples de classification (ou de *clustering*) comme l’algorithme des  $K$  moyennes par exemple. Si les expériences pour ce modèle ne sont pas complètes, elles sont néanmoins prometteuses et vont être poursuivies en vue d’une publication. Ce modèle est aussi entraînable en temps linéaire en la taille du jeu de données et la métrique apprise est relativement rapide à évaluer notamment grâce à une vectorisation partielle du calcul des distances.

En résumé, nos travaux ont consisté en une utilisation variée des GCN pour résoudre différentes problématiques d’apprentissage sur graphes. Nous espérons que le lecteur physicien ou informaticien (ou un peu des deux) peu familier avec les concepts de l’apprentissage automatique ait pu découvrir ce domaine et acquérir une certaine intuition de la difficulté intrinsèque des tâches d’apprentissage sur graphes. Nous espérons également avoir pu convaincre de l’intérêt des GCN (outils prenant ses origines à la fois dans la physique et l’informatique) qui ont permis de faire des progrès importants dans le domaine, en particulier lorsqu’ils sont intégrés aux sein de modèles. Toutefois, les problématiques abordées dans cette thèse sont très reliées aux questions de classification et de *clustering* et ne représentent qu’une infime partie de l’ensemble des tâches d’apprentissage envisageables sur graphes.

## 6.2 Perspectives futures

Nous donnons ici, quelques éléments et thématiques qui pourraient à l’avenir devenir importants pour l’apprentissage sur graphes.

**Les limites des GCN.** Les GCN et les GNN tels que définis dans la littérature sont basés sur des opérations d’échanges plus ou moins sophistiqués d’informations entre nœuds. Ce paradigme de fonctionnement, s’il est efficace, rencontre certaines limites, comme on a pu le voir dans ce manuscrit. Des travaux permettant de surmonter ces limites sont et seront nécessaires à l’avenir et devraient constituer un axe de recherche important du domaine. A cet égard, les informations portées par les arêtes souvent négligées devront être considérées avec plus d’atten-

tion. L'amélioration des GNN pourrait aussi provenir d'un changement de paradigme et de la recherche de nouveaux modèles d'extraction d'information dans un graphe. Ainsi, la recherche de modèles alternatifs sera probablement également un axe important de recherche du domaine. A titre d'exemple d'une telle démarche, certains auteurs [Yun et al. 2020 ; Dwivedi and Bresson 2021] essayent de définir des transformeurs (qui constituent l'état de l'art en traitement du langage naturel) sur les graphes.

**Les aspects théoriques et pratiques.** Que l'avenir de l'apprentissage automatique sur graphes s'écrive avec des évolutions des GCN, des transformeurs ou toute autre méthode, il sera nécessaire d'approfondir les connaissances théoriques du domaine. Des bases plus solides permettront de consolider les modèles actuels, de comprendre leurs limites et d'orienter la recherche plus efficacement, notamment via une meilleure compréhension de l'expressivité des GNN [Loukas 2020] et de leur capacité de généralisation entre autres [Garg et al. 2020]. En parallèle, pour améliorer l'efficacité de la recherche, il serait intéressant d'avoir une plus grande diversité de jeux de données sur graphes qui permettrait de différencier plus finement les méthodes qui actuellement présentent des résultats très proches les unes des autres sans qu'aucune ne se démarque réellement du point de vue de la performance.

**Scalabilité.** La scalabilité, ou passage à l'échelle, est un critère qui devrait prendre de plus en plus d'importance. Il existe aujourd'hui de nombreuses méthodes très efficaces pour accomplir certaines tâches sur graphe, elles ont cependant très souvent un coût prohibitif. L'augmentation croissante du nombre d'objets connectés et de l'accroissement de la vitesse d'échanges d'informations va générer des jeux de données de plus en plus volumineux. Cet état de fait couplé à la multiplication de systèmes temps réels embarquant des algorithmes d'apprentissage automatique (ex : voitures autonomes, chaîne de production dans l'industrie, etc.) renforcera le besoin pour des algorithmes à exécution très rapide. On peut noter que les améliorations dans ce domaine pourraient aussi venir d'avancées sur le matériel informatique.

**Modèles génératifs.** Si les questions de classification et de *clustering* sont prépondérantes dans la littérature de l'apprentissage automatique, de très nombreuses tâches d'apprentissage sur graphes sont d'un grand intérêt et restent à être développées, comme, par exemple les algorithmes de générations de graphes. Les modèles génératifs, qu'ils soient basés sur les *Generative adversarial Networks* – GAN – ou les *Variational Auto-Encodeur* – VAE – ont été intensément développés notamment dans le domaine de l'imagerie. Ces types de réseaux sont encore peu développés sur les graphes. L'une des premières approches dans ce domaine consistait en l'utilisation de réseaux de neurones récurrents pour construire des graphes pas à pas [Bjerrum and Threlfall 2017]; comme on peut s'en douter, cette méthode ne permet pas de construire des graphes de grandes tailles. Si récemment des méthodes s'appuyant sur les VAE [Simonovsky and Komodakis 2018] et les GAN [Cao and Kipf 2018] ont été proposées, notamment pour la génération de molécules chimiques, elles restent cependant limitées : elles ne peuvent par exemple générer que des molécules de petite taille, certaines des molécules générées ne sont pas réalistes et parmi les molécules valides, peu sont réellement originales. Ce domaine reste encore à être développé.

**Manipulation de graphe.** La manipulation de graphe consiste en des opérations visant à modifier le graphe par l'ajout ou le retrait de structures à ce dernier (comme des nœuds, des arrêtes mais aussi potentiellement des structures plus complexes). Le caractère intrinsèquement discret de ces opérations rend la différentiabilité au travers de ces opérations difficile. Pourtant les méthodes de manipulation de graphes sont d'un grand intérêt (par exemple, elles sont sous-jacentes à la GED) et devraient devenir des sujets d'importance dans les années à venir car un grand nombre de problématiques peuvent se décrire aisément avec ces derniers. En chimie, les mécanismes réactionnels peuvent se décrire par des opérations successives de manipulation de graphe. Ces outils pourraient alors permettre de construire des modèles aidant les chimistes à déterminer des mécanismes réactionnels. Ces mécanismes pourraient également permettre une analyse plus fine des réseaux, réseaux sociaux et autres structures dynamiques structurées par des graphes notamment par la compréhension de leur dynamique d'évolution via ces outils.

Ces quelques perspectives laissent présager de nombreuses perspectives pour la communauté de l'apprentissage automatique sur les graphes. Les graphes n'ont pas cessé d'être des sujets de recherche féconds depuis près de 3 siècles. Si Euler savait !



# Annexe A

## Résultats complémentaires

*Cet ultime chapitre sera consacré aux preuves, à l'ajout de quelques résultats additionnels et des précisions portant sur les chapitres qui précèdent.*

### A.1 Chapitre 2 - Généralités

#### A.1.1 Complexité informatique

La théorie de la complexité en informatique permet d'apprécier la difficulté de certains problèmes. Cette classification se base sur le modèle de la machine de Turing, qui permet de définir formellement la notion d'algorithme. Nous allons ici, strictement nous limiter aux classes de complexité évoqué dans ce manuscrit. Etant donné un problème de taille  $n$  ; on dira que :

- Ce problème appartient à la classe **P** (*polynomial*) s'il est soluble par une machine de Turing déterministe en un temps polynomial en  $n$ .
- Ce problème appartient à la classe **NP** (*non deterministic polynomial*) s'il est soluble par une machine de Turing non déterministe en un temps polynomial en  $n$ . Cela équivaut en général à ce que la vérification de la validité d'une proposition de solution puisse s'effectuer en temps polynomial.
- Ce problème appartient à la classe **NP-Difficile** si une solution pour ce problème permet de résoudre tout autre problème NP en un temps au pire polynomial en la complexité de cette solution. Ces problèmes ne sont pas nécessairement NP ; toutefois s'ils le sont, on dit qu'ils sont **NP-Complet**.

Cette hiérarchie qualifie les difficultés des problèmes d'algorithmie, les problèmes P sont plus simples que les problème NP, eux-mêmes plus simples que les problème NP-Complet, quant à eux les problèmes NP-Difficile sont les plus difficiles au sens de la complexité. On notera que les problèmes de la classe NP-complet peuvent changer de complexité en fonction des avancées de la recherche. En général, les problèmes NP sont démontrés soit comme étant dans la classe P soit dans la classe NP-Complet. Une des grandes questions de l'informatique théorique est de savoir si la classe P est égale à la classe NP.

### A.1.2 Graphes co-spectraux

En Figure A.1, on peut observer deux graphes qui sont non-isomorphes mais dont les matrices laplaciennes, présentent le même spectre. Toutefois, ces deux graphes peuvent être distingués avec le spectre de leurs matrices d'adjacences. L'adjonction des spectres de la matrice laplacienne  $L$  et de la matrice d'adjacence  $A$  n'est toutefois toujours pas suffisante pour discriminer les graphes de manière parfaite. Un exemple de deux graphes possédant des spectres identiques pour l'une et l'autre des matrices peut-être retrouvé en Figure A.2.

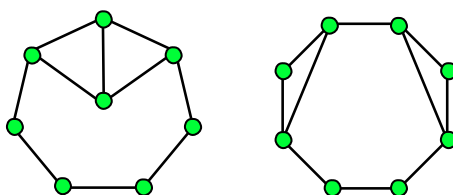


FIGURE A.1 – *Deux graphes différents dont les laplaciens possèdent le même spectre.* Mednykh and Mednykh [2016]; Haemers and Spence [2004].

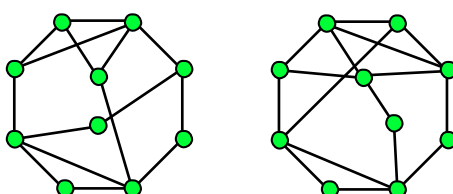


FIGURE A.2 – *Deux graphes différents dont la matrice d'adjacence et le laplacien possèdent le même spectre.* Haemers and Spence [2004].

Évidemment, il existe d'autres opérateurs dont le spectre permet de distinguer les graphes de la figure ci-dessus. Haemers and Spence [2004] ont comparé le pouvoir discriminant de certains de ces opérateurs, en énumérant le nombre de graphes co-spectraux qu'ils induisaient pour les graphes de tailles  $n \leq 11$ . Dans leur analyse, on peut notamment constater que le spectre de la matrice d'adjacence est moins discriminant que le spectre de la matrice laplacienne qui est lui-même moins discriminant que le spectre de la valeur absolue de la matrice laplacienne. Les résultats pour les deux premières matrices sus-citées peuvent-être retrouvés en Table A.1.

Comme on peut le constater sur ce tableau, les graphes co-spectraux sont de plus en plus nombreux pour la matrice d'adjacence, alors qu'ils se raréfient pour la matrice laplacienne lorsque le nombre de sommets croît. Les auteurs présentent dans leurs articles des éléments appuyant le fait que les graphes co-spectraux (pour  $L$ ) tendent à se raréfier lorsque  $n$  tend vers l'infini ; le spectre de la matrice Laplacienne est donc un bon discriminateur bien qu'imparfait et quelque peu coûteux à calculer.

$n$	# graphe	$\mathbf{A}$	$\mathbf{L}$
2	2	0	0
3	4	0	0
4	11	0	0
5	34	0.059	0
6	156	0.064	0.026
7	1 044	0.105	0.125
8	12 346	0.139	0.143
9	274 668	0.186	0.155
10	12 005 168	0.213	0.118
11	1 018 997 864	0.211	0.090

TABLE A.1 – *Tableau de résultats d'énumération des graphes co-spectraux.* La **première colonne** correspond au nombre de sommets des graphes considérés. La **seconde colonne** indique le nombre de graphes uniques à  $n$  sommets (graphes non-connexes compris). La **3 ème (resp. 4 ème) colonne** est la fraction de graphes partageant le spectre de sa matrice d'adjacence (resp. laplacienne) avec au moins un graphe.

## A.2 Chapitre 3 - HG2V

### A.2.1 Weisfeiler-Lehman

Lorsque deux graphes sont différents, la procédure de Weisfeiler-Lehman permet très souvent de les distinguer en produisant des histogrammes d'étiquettes différents. Toutefois, si la procédure renvoie les mêmes histogrammes d'étiquettes, on n'a pas de garantie absolue que les deux graphes considérés soient isomorphes (voir Fig. (A.3)) même si ces configurations sont rares.

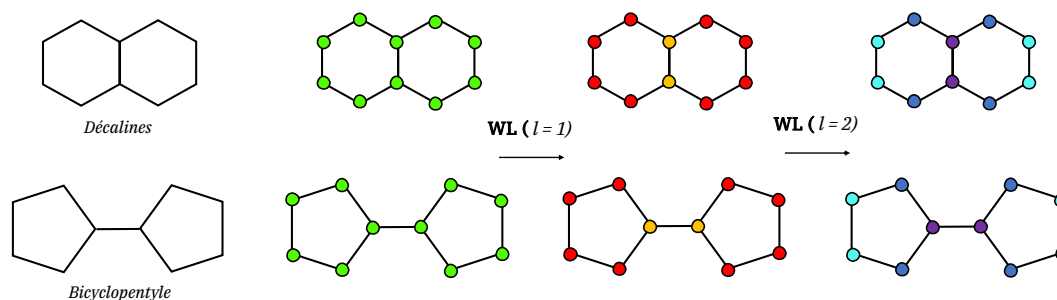


FIGURE A.3 – *isomorphisme de graphes avec Weisfeiler-Lehman.* Un contre exemple tiré de l'article de Sato [2020]. WL génère pour ces deux graphes moléculaires les mêmes histogrammes à chaque étape.

Il existe des variations plus fortes de la procédure de WL appelée Weisfeiler-Lehman  $k$  dimensionnel ( $k$ -dim WL), elles ont été introduites par László Babai et Rudolf Mathon [Babai

1979]. Dans cette procédure, on considère des  $k$ -uplet de nœuds en lieu et place des nœuds. Pour  $k > 1$ , il existe toujours des graphes tel que  $(k + 1)$ -WL puisse les distinguer mais pas  $k$ -WL. On a donc une hiérarchie de procédures exécutables en temps polynomial toujours plus puissante, toutefois la propriété précédente prouve aussi qu'il n'existe aucun entier  $k$  pour lequel  $k$ -WL puisse distinguer tous les graphes non isomorphes. Ces procédures restent néanmoins des discriminateurs de graphes très puissants ; récemment elles ont été utilisées au sein d'un algorithme plus vaste qui résout le problème d'isomorphisme de graphes en temps quasi polynomial (sans hypothèses sur la nature des graphes) Helfgott [2017] ; Babai [2015].

### A.2.2 Continuité des GNN

**Definition 5 (Distance de *Fused* Gromov-Wasserstein.)** Soit  $\mathcal{G}_1 = (V_1, E_1)$  et  $\mathcal{G}_2 = (V_2, E_2)$  deux graphes attribués respectivement associés aux matrices d'adjacence  $\mathbf{A}_1$  et  $\mathbf{A}_2$  dans  $\mathbb{G}_x$ .

$$d_{\mathcal{FGW}}(\mathcal{G}_1, \mathcal{G}_2) = \min_{\pi \in \Pi} \sum_{u, u', v, v'} \pi(u, v) \pi(u', v') (|\mathbf{A}_1(u, u') - \mathbf{A}_2(v, v')| + |\mathbf{x}(u) - \mathbf{x}(v)| + |\mathbf{x}_1(u') - \mathbf{x}_2(v')|) \quad (\text{A.1})$$

où  $\Pi$  est l'ensemble des mesures sur  $V_1 \times V_2$  dont les marginales sont des distributions uniformes sur  $V_1$  and  $V_2$ . La version de *Fused Gromov-Wasserstein* présentée ignore le paramètre de compromis entre le poids de la contribution des caractéristiques des nœuds et la structure du graphe dans le calcul de la distance, car il n'a pas d'utilité ici.

On va établir que les GNN sont des fonctions continues pour (la topologie induite par) la métrique  $d_{\mathcal{FGW}}$  dès lors que les fonctions d'activations sont continues. Pour démontrer cette affirmation, nous allons nous appuyer sur le *Message Passing Neural Networks* – MPNN – qui est un cadre théorique introduit par Gilmer et al. [2017] dans lequel les GNN peuvent être décrits. Soit  $\mathcal{G} = (V, E)$  associé à la matrice d'adjacence  $\mathbf{A}$  dans  $\mathbb{G}_x$ ,  $u \in V$  l'un de ses nœuds et  $l$  l'indice de l'étape, alors un GNN se décrit par les opérations suivantes :

$$\mathbf{x}^{(0)}(u) = \mathbf{x}(u) \quad (\text{A.2})$$

$$\mathbf{m}^{(l+1)}(u) = \sum_{v \in \mathcal{V}(u)} \mathbf{M}^{(l)}(\mathbf{x}^{(l)}(u), \mathbf{x}^{(l)}(v), \{u, v\}) \quad (\text{A.3})$$

$$\mathbf{x}^{(l+1)}(u) = \mathbf{U}^{(l)}(\mathbf{x}^{(l)}(u), \mathbf{m}^{(l+1)}(u)) \quad (\text{A.4})$$

Les fonctions  $\mathbf{M}^{(l)}$  et  $\mathbf{U}^{(l)}$  sont des opérations mettant en jeu des multiplications matricielles et des non linéarités s'appliquant terme à terme, ce sont donc des fonctions continues par composition de fonctions continues. De même, par somme et composition de fonctions continues, il est immédiat que la fonction  $\{\mathbf{x}^{(0)}(v) | \forall v \in V^{(L)}(u)\} \mapsto \mathbf{x}^{(L-1)}(u)$  est continue par rapport aux caractéristiques  $\mathbf{x}^{(0)}(v)$  du graphe.

Considérons la fonction qui à un graphe associe le même graphe après application du GNN :

$$F : \mathbb{G}_x \rightarrow \mathbb{G}_{x^{(L-1)}} \quad (\text{A.5})$$

$$\mathcal{G} \mapsto \mathcal{G}_F \quad (\text{A.6})$$



Cette fonction est telle que  $V = V_F$ ,  $E = E_F$ , seuls les signaux sur graphes sont modifiés par le GNN, on passe de  $x^{(0)}(u)$  dans  $\mathcal{G}$  à  $\mathbf{x}^{(L)}(u)$  dans  $\mathcal{G}_F$ .

Soit une suite de graphes attribués  $(\mathcal{G}_n)_{n \in \mathbb{N}} \in \mathbb{G}_x^{\mathbb{N}}$  tel que  $\lim_{n \rightarrow \infty} \mathcal{G}_n = \mathcal{G}$  dans l'espace topologique induit par la métrique  $d_{\mathcal{FGW}}$ ; considérons les plans de transport  $(\pi_n : V_n \times V)_{n \in \mathbb{N}}$  entre  $\mathcal{G}_n$  et  $\mathcal{G}$  pour cette distance; alors :

$$\begin{aligned} & d_{\mathcal{FGW}}(F(\mathcal{G}_n), F(\mathcal{G})) \\ & \leq \sum_{u, u', v, v'} \pi_n(u, v) \pi_n(u', v') \left( |\mathbf{A}_{F_n}(u, u') - \mathbf{A}_F(v, v')| + |\mathbf{x}_n^{(L)}(u) - \mathbf{x}^{(L)}(v)| + |\mathbf{x}_n^{(L)}(u') - \mathbf{x}^{(L)}(v')| \right) \\ & = \sum_{u, u', v, v'} \pi_n(u, v) \pi_n(u', v') \left( |\mathbf{A}_n(u, u') - \mathbf{A}(v, v')| + |\mathbf{x}_n^{(L)}(u) - \mathbf{x}^{(L)}(v)| + |\mathbf{x}_n^{(L)}(u') - \mathbf{x}^{(L)}(v')| \right) \end{aligned} \quad (\text{A.7})$$

Comme,  $\lim_{n \rightarrow \infty} \mathcal{G}_n = \mathcal{G}$  on a nécessairement :

$$\lim_{n \rightarrow \infty} \pi_n(u, v) \pi_n(u', v') |\mathbf{x}_n^{(L)}(u) - \mathbf{x}^{(L)}(u)| = 0 \quad (\text{A.8})$$

$$\lim_{n \rightarrow \infty} \pi_n(u, v) \pi_n(u', v') |\mathbf{A}_n(u, u') - \mathbf{A}(v, v')| = 0 \quad (\text{A.9})$$

La continuité de  $\mathbf{x}^{(L)}$  par rapport à  $x$  permet de conclure que :

$$\lim_{n \rightarrow \infty} \pi_n(u, v) \pi_n(u', v') |\mathbf{x}_n^{(L)}(u) - \mathbf{x}^{(L)}(u)| = 0 \quad (\text{A.10})$$

Donc la limite du membre de droite (A.7) doit avoir une limite nulle, donc :

$$\lim_{n \rightarrow \infty} F(\mathcal{G}_n) = F(\mathcal{G}) \quad (\text{A.11})$$

On a donc montré que  $\lim_{n \rightarrow \infty} \mathcal{G}_n = \mathcal{G}$  implique  $\lim_{n \rightarrow \infty} F(\mathcal{G}_n) = F(\mathcal{G})$ , ce qui est la définition de la continuité de  $F$  par rapport à la topologie induite par la métrique  $d_{FGW}$ .

### A.2.3 Diffusion Limited Aggregation

Le jeu de données DLA à été généré par la simulation d'un processus de diffusion limité par l'aggrégation (*Diffusion Limited Aggregation* - DL [Witten Jr and Sander 1981]. Dans cette simulation<sup>1</sup>, des particules suivent le mouvement Brownien et lorsqu'elles entrent en contact elles sont susceptibles de s'aggréger avec une probabilité égale à  $p$ . Ce processus se poursuit jusqu'à ce que toutes les particules se soient agrégées. Pour simplifier la simulation, cette dernière est effectuée en 2D. La structure de graphe résultante est un arbre, les particules étant les nœuds et les arêtes des liens physico-chimiques. Ces graphes ont la propriété d'être invariants d'échelle [Witten Jr and Sander 1981]. La distribution de degré des nœuds (ou particules) et leurs positions dans l'espace dépend du paramètre  $p$ . Les positions des particules sont utilisées comme attributs des nœuds.

1. Le code pour générer DLA peut-être trouvé ici : <https://github.com/Algue-Rythme/DiffusionLimitedAggregation>.

Nous avons généré un total de 1000 graphes avec 500 nœuds chacun. Ce jeu de données est découpé en 2 classes, d’une part les graphes générés avec une probabilité d’agrégation  $p = 1$  et d’autre part ceux avec une probabilité d’agrégation  $p = 0.2$ .

Comme on peut le voir dans les résultats expérimentaux, *Graph2vec* n’obtient pas une bonne *accuracy* en usant seulement des degrés comme attribut des nœuds. Alors que HG2V peut se servir des attributs des nœuds pour augmenter la qualité de la prédiction et ceux de façon importante, d’où l’intérêt des méthodes capables d’user des attributs continus et de rendre compte de propriétés globales.

#### A.2.4 Construction de graphes à partir d’images : USPS, MNIST et FASHION-MNIST

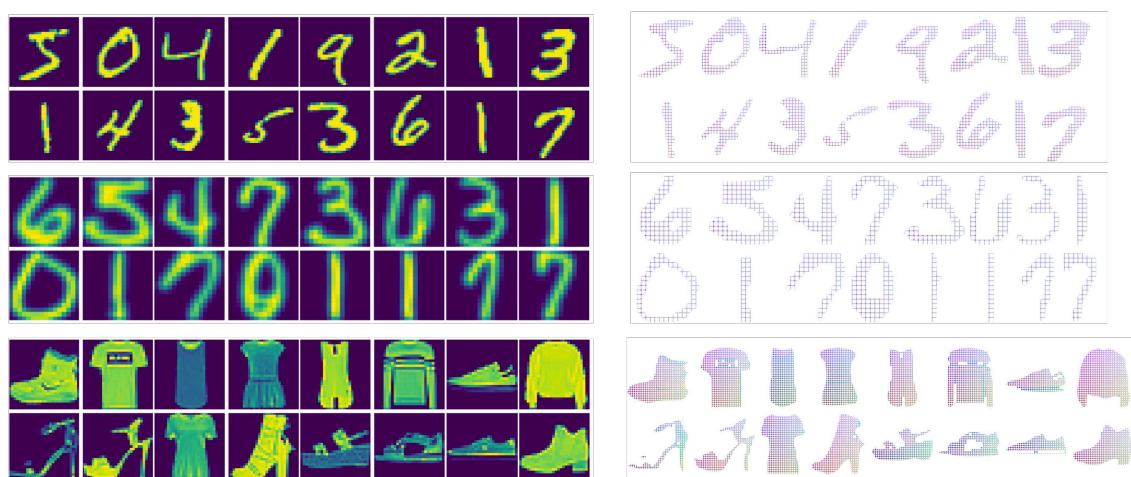


FIGURE A.4 – *De haut en bas : MNIST, USPS et FASHIONMNIST.*

Les graphes produits à partir de MNIST et FASHIONMNIST (resp. USPS) ont été créés en retirant tous les pixels de luminosité nulle (resp. inférieur à 0.3). Chaque pixel résultant est associé à un nœud, dont l’attribut est le vecteur de luminosité du pixel auquel il est associé concaténé avec sa position  $(x,y)$  dans l’image originale. Seules 10 000 images de MNIST et de FASHIONMNIST ont été converties en graphe selon cette procédure. L’ensemble des images d’USPS : 9298 ont subi la même transformation. Ces jeux de données ainsi créés sont largement plus grands que la taille standard des autres jeux de données.

#### A.2.5 HG2V entraînement sur une fraction des données

Dans la Table A.2, on peut voir les résultats des expériences décrites en Section 3.4.2 mais cette fois-ci seulement lorsque seule une partie des jeux de données a été utilisée pour entraîner les poids de HG2V, c’est à dire 20%. Les résultats de cette expérience ne sont pas directement comparables à ceux de l’expérience décrite en Section 3.4.2 car certains paramètres diffèrent notamment le nombre d’*epoch* d’entraînement qui était fixé à 30 soit 3 fois plus qu’en Section 3.4.2.

Jeu de données	HG2V (20%)
IMDB-m	47.9 ± 1.0
PTC_FR	67.5 ± 0.5
FRANK.	65.3 ± 0.7
MUTAG	81.8 ± 1.8
IMDB-b	71.3 ± 0.8
NCI1	76.3 ± 0.8
NCI109	75.6 ± 0.7
ENZYMES	66.0 ± 2.5
PROTEINS	75.7 ± 0.7
MNIST	96.1 ± 0.2
D&D	79.2 ± 0.8
REDDIT-b	91.2 ± 0.6
DLA	99.9 ± 0.1
REDDIT-5K	55.5 ± 0.7

TABLE A.2 – *L’accuracy dans la tâche de classification.* Cette expérience fait écho à l’expérience décrite en Section 3.4.2. HG2V est entraîné sur seulement 20 % de l’ensemble d’entraînement. Après l’entraînement, les poids appris sont utilisés pour générer la représentation latente de tous les graphes (ensemble d’entraînement et de test). La sélection des hyperparamètres de la SVM-RBF est effectuée via une validation croisée à 5 blocs. La table présente les résultats de classification pour l’ensemble de test.

Toutefois, ce résultat démontre la capacité inductive de la méthode qui même en s’entraînant sur une fraction minimale d’un jeu de données permet d’obtenir des résultats de classification proche de l’état de l’art sur nombre d’entre eux.

### A.2.6 Visualisation de l’espace latent

On présente d’autres graphes et leurs 6 plus proches voisins dans l’espace latent, provenant de MNIST, IMDB-b et PTC sur les figures A.5, A.6 et A.7.

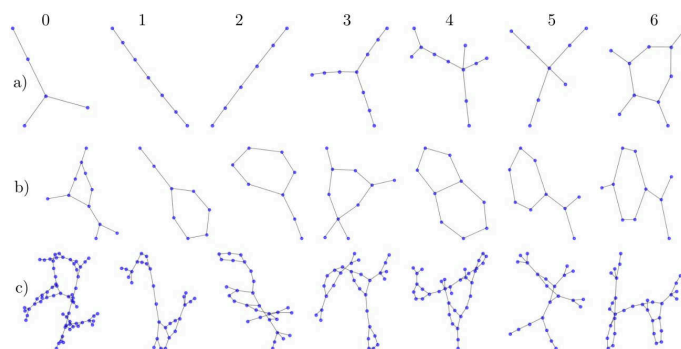


FIGURE A.5 – *Six plus proches voisins dans l’espace latent pour PTC.* La colonne 0 correspond aux graphes choisis aléatoirement et les 6 plus proches voisins (dans l’ordre) correspondent aux colonnes 1 à 6.

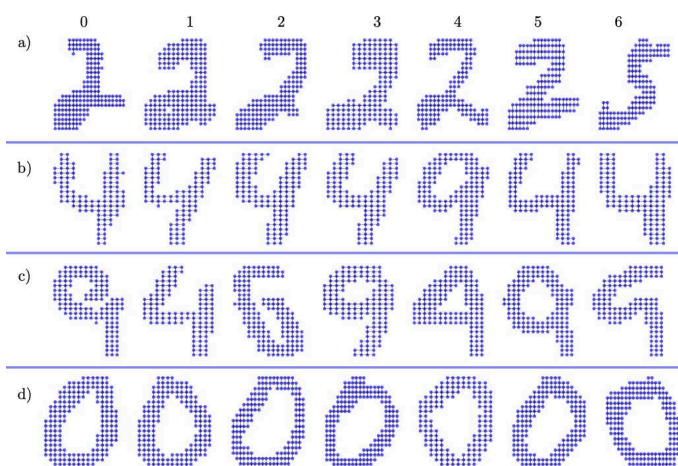


FIGURE A.6 – *Six plus proches voisins dans l'espace latent pour MNIST*. La colonne 0 correspond aux graphes choisis aléatoirement et les 6 plus proches voisins (dans l'ordre) correspondent aux colonnes 1 à 6.

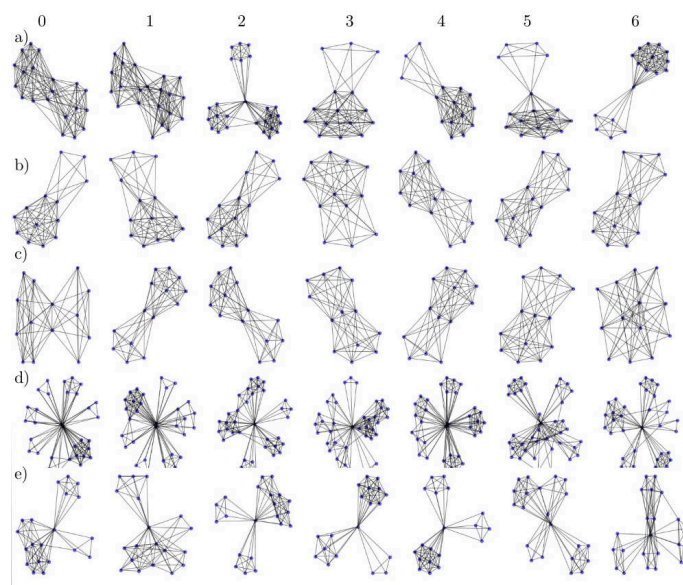


FIGURE A.7 – *Six plus proches voisins dans l'espace latent pour IMDB-b*. La colonne 0 correspond aux graphes choisis aléatoirement et les 6 plus proches voisins (dans l'ordre) correspondent aux colonnes 1 à 6.

## A.3 Chapitre 4 - MVGCCA

### A.3.1 Modèles algébriques alternatifs

Pour développer une méthode CCA à la fois scalable et capable de tenir compte d'un graphe structurant l'espace latent, nous avons initialement envisagé 2 modèles algébriques. Un premier modèle (GCNCCA) se base directement sur DeepCCA [Andrew et al. 2013] et GMCCA [Chen et al. 2019], il consiste à simplement remplacer dans la fonctionnelle (4.13) les vues  $\mathbf{X}_m$  par un  $m$  dépendant de ces vues et du graphe. Ce qui se traduit par la fonctionnelle suivante :

$$\min_{(\mathbf{U}_m)_{m=1}^M, \mathbf{S}} \sum_{m=1}^M \|\text{GCN}_m(\mathbf{X}_m, \mathbf{A})\mathbf{U}_m - \mathbf{S}\|_{\mathcal{F}}^2 + \gamma \text{tr}(\mathbf{S}^T \mathbf{L} \mathbf{S}) \text{ t.q. } \mathbf{S}^T \mathbf{S} = \mathbf{I}_q. \quad (\text{A.12})$$

Le graphe est donc utilisé à la fois pour construire l'espace latent mais aussi pour régulariser ce dernier, en imposant aux représentations d'être lisses sur ce dernier comme dans GMCCA. Le second modèle (DGCCA) est très proche du premier (on peut le rapprocher de Autoencoder CCA [Wang et al. 2015]), les GCN sont remplacés par des perceptrons multicouches, on ajoute une contrainte pour que le modèle soit en mesure de restituer les vues à partir des représentations des vues et on conserve la contrainte des représentations lisses sur le graphe. La fonctionnelle de ce modèle est la suivante :

$$\min_{(\mathbf{U}_m)_{m=1}^M, \mathbf{S}} \|\text{MLP}_m^{\text{enc}}(\mathbf{X}_m)\mathbf{U}_m - \mathbf{S}\|_{\mathcal{F}}^2 + \alpha \|\mathbf{X}_m - \text{MLP}_m^{\text{dec}}(\text{MLP}_m^{\text{enc}}(\mathbf{X}_m))\|_{\mathcal{F}}^2 + \gamma \text{tr}(\mathbf{S}^T \mathbf{L} \mathbf{S}) \text{ t.q. } \mathbf{S}^T \mathbf{S} = \mathbf{I}_q. \quad (\text{A.13})$$

Ce modèle comporte un hyperparamètre supplémentaire  $\alpha$  qui quantifie l'importance donnée à la reconstruction. On peut voir DGCCA comme une prémice à MVGCCA, DGCCA combine un auto-encodeur avec une contrainte de lissage dans l'espace latent algébrique alors que MVGCCA est un auto-encodeur variationnel qui intègre la contrainte de lissage dans l'espace latent directement dans la formulation variationnel. Au delà de la différence manifeste de nature des deux modèles, une des différences majeure entre ces modèles est qu'on décode les vues à partir des représentations des vues  $\text{MLP}_m^{\text{enc}}(\mathbf{X}_m)$  dans DGCCA alors que dans MVGCCA ces vues sont décodées à partir de la représentation commune  $\mathbf{Z}$  (qui correspond à  $\mathbf{S}$ ).

Les projections linéaires  $\mathbf{U}_m$  (comme dans DeepCCA) sont conservées car elles permettent d'assurer plus facilement la condition  $\mathbf{S}^T \mathbf{S} = \mathbf{I}_q$ . Des variations mineures de ces modèles ont également été testées, mais elles présentent des résultats très similaires aux modèles ci-dessus et n'en sont pas significativement différents, ils ne seront donc pas présentés. Les deux modèles présentés ici sont scalables car ils peuvent être appris par *batch*. Nous avons procédé à quelques expériences sur les jeux de données *uci7*, *uci10* et *tfr* introduits en Section 4.5.1. Les résultats de ces expériences peuvent être trouvés Table A.3. Comme on peut le voir, les résultats sont mitigés. On ne perçoit pas de gain notable alors que ces modèles (du fait de la décomposition en éléments propres nécessaires à effectuer à chaque étape de descente de gradient) sont relativement instables numériquement et dont les performances peuvent fortement varier d'une exécution à l'autre.

Jeu de données	uci7			uci10			Recommandation		
	Acc.	ARI	ARI2	Acc.	ARI	ARI2	Prec.	Recall	Mrr
PCA	0.866	0.539	-	0.714	0.417	-	0.15	0.049	0.345
GPCA	0.948	0.712	-	0.869	0.635	-	-	-	-
GMCCA	<b>0.951</b>	<b>0.856</b>	-	<b>0.910</b>	0.735	-	0.249	<b>0.084</b>	0.481
GCNCCA	0.918	0.752	-	0.879	0.672	-	<b>0.252</b>	<b>0.084</b>	0.467
DGCCA	0.938	0.814	-	0.892	<b>0.685</b>	-	0.242	0.08	<b>0.491</b>

TABLE A.3 – *Résultats des expériences des tâches d’apprentissage effectuées sur uci et tfr*. Les abréviations utilisées sont les mêmes que dans la Table 4.2. Les résultats des méthodes de la littérature diffèrent quelque peu de ceux de la Table 4.2 car les expériences ont été refaites avec des conditions légèrement différentes.

### A.3.2 Forme explicite de l’ELBO

Dans cette partie, nous présentons comment l’ELBO d’entraînement de l’Equation (4.38) utilisé dans chaque batch est approximé. Nous considérons dans un premier temps le cas où il n’y a pas de *views dropout*. Dans ce cas, cet ELBO s’écrit pour chaque *batch* d’indice  $B_k$  :

$$\begin{aligned}
\mathcal{L}_{ELBO}^{B_k} &= \sum_{i \in B_k} \sum_{j \in B_k} \mathbb{E}_{\substack{z \sim q_\eta(z | \mathcal{V}_p(\mathbf{x}^i), \mathbf{A}) \\ z' \sim q_\eta(z' | \mathcal{V}_p(\mathbf{x}^j), \mathbf{A})}} \log p_E(\mathbf{A}_{i,j} | z, z') \\
&+ \sum_{i \in B_k} \sum_{m=1}^M \mathbb{E}_{z \sim q_\eta(z | \mathcal{V}_p(\mathbf{x}^i), \mathbf{A})} \log p_{\theta_m}(X_m^i | z) \\
&- \sum_{i \in B_k} D_{\text{KL}}(q_\eta(z | \mathcal{V}_p(\mathbf{x}^i), \mathbf{A}) \| p(z)).
\end{aligned} \tag{A.14}$$

Cette ELBO est approximé de la façon suivante (dans l’ordre respectif des termes) :

$$\begin{aligned}
\mathcal{L}_{ELBO}^{B_k} &\approx \frac{1}{K} \sum_{k=1}^K \sum_{i \in B_k} \sum_{j \in B_k} \log p_G(A_{i,j} | z_k^i, z_k^j) \\
&+ \frac{1}{K} \sum_{k=1}^K \sum_{i \in B_k} \sum_{m=1}^M \log p_{\theta_m}(x_m^i | z_k^i) \\
&- \frac{1}{2} \sum_{i \in B_k} \sum_{k=1}^q 1 + \log \left( \frac{\prod_{m=1}^M \sigma_m^{\text{enc}2}(i, k)}{\sum_{m'=1}^M \prod_{\substack{m=1 \\ m' \neq m}}^M \sigma_m^{\text{enc}2}(i, k)} \right)^2 - \left( \frac{\sum_{m'=1}^M \mu_{m'}^{\text{enc}}(i, k) / \sigma_m^{\text{enc}2}(i, k)}{\sum_{m'=1}^M 1 / \sigma_m^{\text{enc}2}(i, k)} \right)^2 \\
&- \frac{1}{2} \sum_{i \in B_k} \sum_{k=1}^q - \left( \frac{\prod_{m=1}^M \sigma_m^{\text{enc}2}(i, k)}{\sum_{m'=1}^M \prod_{\substack{m=1 \\ m' \neq m}}^M \sigma_m^{\text{enc}2}(i, k)} \right)^2
\end{aligned} \tag{A.15}$$

Les deux premiers premier termes approximent l'espérance via un échantillonnage de Monte-Carlo. Ces termes font intervenir les  $K$  grandeurs  $\mathbf{z}_k^i \in \mathbb{R}^q$  qui sont échantillonnées selon  $q_\eta(\mathbf{z}|\mathcal{V}_p(\mathbf{x}^i), \mathbf{A})$ . Dans nos expériences  $K$  le nombre d'échantillons est fixé à 1 pour chaque élément de chaque *batch*. La dernière ligne est une expression qui découle d'une part de l'expression de la divergence de Kullback-Leibler entre deux distributions gaussiennes multivariées de matrices de covariances diagonales  $p_a(\mathbf{z}) = \mathcal{N}([\mu_{a,1}, \dots, \mu_{a,d}]^T, \text{diag}([\sigma_{a,1}^2, \dots, \sigma_{a,q}^2]))$  et  $p_b(\mathbf{z}) = \mathcal{N}([\mu_{b,1}, \dots, \mu_{b,d}]^T, \text{diag}([\sigma_{b,1}^2, \dots, \sigma_{b,q}^2]))$  :

$$D_{\text{KL}}(p_a(\mathbf{z})\|p_b(\mathbf{z})) = -\frac{1}{2} \sum_{k=1}^q 1 - \log \frac{\sigma_b^2}{\sigma_a^2} - \frac{\sigma_a^2}{\sigma_b^2} - \frac{(\boldsymbol{\mu}_a - \boldsymbol{\mu}_b)^2}{\sigma_b^2} \quad (\text{A.16})$$

Et du fait que :

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{0}_q, \mathbf{I}_q)$$

et (voir Annexe. [A.3.3](#)) :

$$q_\eta(\mathbf{z}|\mathcal{V}_p(\mathbf{x}^i), \mathbf{A}) = \mathcal{N}\left(\left[\frac{\sum_{m'=1}^M \mu_{m'}^{\text{enc}}(i, k)/\sigma_m^{\text{enc}2}(i, k)}{\sum_{m'=1}^M 1/\sigma_m^{\text{enc}2}(i, k)}\right]_{k=1}^q, \text{diag}\left[\frac{\prod_{m=1}^M \sigma_m^{\text{enc}2}(i, k)}{\sum_{m'=1}^M \prod_{\substack{m=1 \\ m' \neq m}}^M \sigma_m^{\text{enc}2}(i, k)}\right]_{k=1}^q\right).$$

Dans le cas où il y a du *views dropout*, alors cette même expression est valable à condition dans tous les produits et toutes les sommes sur le nombre de vues  $M$ , de les restreindre aux vues disponibles.

### A.3.3 Représentation latente

Comme vu en Section [4.4.4](#), par analogie avec PCCA la représentation latente de  $\mathbf{x}^i$  est donnée par l'espérance de l'encodeur  $q_\eta(\mathbf{z}|\mathcal{V}_p(\mathbf{x}^i), \mathbf{A}) = \prod_{m=1}^M q_{\eta_m}(z)|\mathcal{V}_p(\mathbf{x}_M^i), \mathbf{A}$  (Eq. [\(4.36\)](#))) qui est une gaussienne multivariée car chacun des éléments du produit est une gaussienne multivariée (Eq. [\(4.29\)](#)) :

$$q_{\eta_m}(z|\mathcal{V}_p(\mathbf{x}_m^i), \mathbf{A}) = \mathcal{N}(\boldsymbol{\mu}_m^{\text{enc}}(i, :), \text{diag}(\boldsymbol{\sigma}_m^{\text{enc}2}(i, :)))$$

Ainsi :

$$q_\eta(\mathbf{z}|\mathcal{V}_p(\mathbf{x}^i), \mathbf{A}) = \mathcal{N}(\boldsymbol{\mu}^{\text{enc}}(i, :), \text{diag}(\boldsymbol{\sigma}^{\text{enc}2}(i, :)))$$

où d'après [[Petersen et al. 2008](#)]  $\boldsymbol{\sigma}^{\text{enc}}(:, i)$  est tel que :

$$\text{diag}(\boldsymbol{\sigma}^{\text{enc}2}(i, :)) = \left(\sum_m^M \text{diag}(\boldsymbol{\sigma}_m^{\text{enc}2}(i, :))^{-1}\right)^{-1}$$

Donc :

$$\forall k \in \llbracket 1, q \rrbracket, \sigma^{\text{enc2}}(i, k) = \frac{\prod_{m=1}^M \sigma_m^{\text{enc2}}(i, k)}{\sum_{m'=1}^M \prod_{\substack{m=1 \\ m' \neq m}}^M \sigma_m^{\text{enc2}}(i, k)}$$

Et d'après la même référence :

$$\boldsymbol{\mu}^{\text{enc}}(i, :) = \text{diag}(\boldsymbol{\sigma}^{\text{enc2}}(:, i)) \left( \sum_m^M \text{diag}(\boldsymbol{\sigma}_m^{\text{enc2}}(i, :))^{-1} \boldsymbol{\mu}_m^{\text{enc}}(i, :) \right)$$

Donc :

$$\forall k \in \llbracket 1, q \rrbracket, \boldsymbol{\mu}^{\text{enc}}(i, k) = \frac{\sum_{m'=1}^M \boldsymbol{\mu}_{m'}^{\text{enc}}(i, k) / \sigma_m^{\text{enc2}}(i, k)}{\sum_{m'=1}^M 1 / \sigma_m^{\text{enc2}}(i, k)}$$

Ce qui justifie la formule de la représentation latente de l'Equation (4.36).

### A.3.4 Construction d'un graphe synthétique

Les jeux de données UCI et twitter n'ont pas de structure de graphe naturelle, [Chen et al. \[2019\]](#) ont proposé la procédure suivante pour créer le graphe (que nous avons suivi) :

- On choisit un ensemble d'indices  $I \subset \{1, \dots, M\}$  dans l'ensemble des indices des vues disponibles.
- Pour chacune de ces vues  $m \in I$ , on construit une matrice  $\mathbf{W}^m$  :

$$\mathbf{W}_{i,j}^m = \begin{cases} \mathbf{K}_{i,j}^m, & i \in \mathcal{V}_k(j) \text{ ou } j \in \mathcal{V}_k(i) \\ 0 & \text{sinon.} \end{cases} \quad (\text{A.17})$$

où en posant  $h = \frac{2}{n(n-1)} \sum_{\substack{i,j=1 \\ i>j}}^n \|\mathbf{X}_m(i, :) - \mathbf{X}_m(j, :)\|_{\mathcal{F}}^2$  :

$$\mathbf{K}_{i,j}^m = e^{\frac{-\|\mathbf{x}_m(i, :) - \mathbf{x}_m(j, :)\|_{\mathcal{F}}^2}{h}} \quad (\text{A.18})$$

Et  $\mathcal{V}_k(i)$  est l'ensemble des indices des  $k$  plus proches lignes de  $\mathbf{K}^m$  de la ligne  $\mathbf{K}^m(i, :)$ .

- Finalement le graphe pondéré de notre jeu de données multivues est alors donné par la formule suivante :

$$\mathbf{A} = \sum_{m \in I} \mathbf{W}^m \quad (\text{A.19})$$

Les paramètres choisis pour uci7 et uci10 sont :  $I = \{3\}$  et  $k = 50$ . Les paramètres choisis pour twitter lorsque  $n = 2506$  sont :  $I = \{1, 2, 3\}$  et  $k = 50$ . Pour  $n = 12530$  :  $I = \{1, 2, 3\}$  et  $k = 50$ .



## A.4 Chapitre 5 - SGML

### A.4.1 *Graph Metric Learning* dans la littérature

Le terme *Graph Metric Learning* peut prêter à confusion car il est effectivement utilisé pour nommer des tâches différentes de l'AM *entre* graphes et relativement éloignées les unes des autres :

- Ce terme peut se référer à des méthodes qui font de la classification de données *euclidiennes* (notamment des images) en construisant un graphe. A partir d'un jeu de données dont seule une partie des étiquettes sont connues, elles construisent un graphe pondéré. Les poids de ce graphe sont calculés à l'aide d'une distance construite pour refléter la structure des étiquettes des éléments pour lesquelles elles sont connues [Dhillon et al. 2010; Wauquier and Keller 2015; Zhu et al. 2020]. Ce graphe est alors utilisé pour procéder à la classification des éléments du jeu de données non étiquetés. En somme, il s'agit d'une tâche d'apprentissage de métrique sur les données euclidiennes mais qui utilise un graphe comme intermédiaire.
- Ce terme peut également désigner l'apprentissage d'une métrique mais *entre* les nœuds d'un graphe ; en s'appuyant notamment sur les caractéristiques portées par ces derniers et la structure de leur graphe [Heitz et al. 2021]. Encore une fois, il s'agit d'un AM sur données euclidiennes, puisque les informations portées par les nœuds sont essentiellement des données euclidiennes.
- Enfin, ce terme peut également désigner la tâche qui consiste à approximer (notamment via des réseaux de neurones) certaines distances entre graphes difficiles à calculer. Ces méthodes opèrent à partir de la connaissance de la véritable valeur de la distance pour quelques exemples [Riba et al. 2018; Zhang 2020; Li et al. 2019b]. Ces méthodes ciblent notamment la *Graph Edit Distance* – GED – qui est une véritable distance sur les graphes mais très difficile à estimer.

### A.4.2 Processus ponctuels déterminantaux

**Les processus ponctuels déterminantaux.** Ces processus issus de la physique ont été introduits par la physicienne et mathématicienne française Odile Macchi [Macchi 1975] initialement pour décrire les distributions de fermions<sup>2</sup> qui sont des particules obéissant aux principes de Pauli [Pauli 1925] et qui ne peuvent donc pas se trouver dans le même état au même endroit. Ces distributions sont fonctions de  $n$  points dans l'espace  $\mathbb{R}^d$ , et sont proportionnelles au déterminant formé par la matrice de ces  $n$  points.

$$\{\boldsymbol{\theta}_m\}_{m=1}^M \sim \det([\boldsymbol{\theta}_1 \dots \boldsymbol{\theta}_M])$$

Cette propriété implique que les configurations spatiales où des points sont proches dans l'espace sont très peu probables car dans ce cas là, la quasi co-linéarité de leurs vecteurs posi-

---

2. Les fermions sont des particules possédant un spin demi-entier (comme les électrons) en opposition aux bosons (comme les photons) qui possèdent un spin entier. L'une des principales caractéristiques liée à cette différence est que deux fermions (contrairement aux bosons) ne peuvent se trouver aux mêmes endroits avec le même état quantique. Cette propriété est notamment responsable de l'impénétrabilité de la matière.

tions fait tendre la valeur du déterminant vers 0. Les propriétés répulsives des points de cette distribution permettent pour le calcul d'une intégrale par MC une convergence en  $O\left(\frac{1}{\sqrt{M^{1+\frac{1}{q'}}}}\right)$ .

### A.4.3 Quasi Monte-Carlo sur l'hypersphère

Considérons une fonction  $f$  qu'on souhaite intégrer sur  $[0, 1]^{q'}$ , et considérons une suite  $\{\mathbf{x}_m\}_{m=1}^M$  d'éléments de  $[0, 1]^{q'}$ , la discrédance  $D_M$  de cette suite se définit comme :

$$D_M = \max_{\mathbb{Q} \subseteq [0, 1]^{q'}} \left| \frac{|\{\mathbf{x}_m | \mathbf{x}_m \in \mathbb{Q}\}|}{M} - \text{Vol}(\mathbb{Q}) \right| \quad (\text{A.20})$$

où  $\mathbb{Q}$  est un pavé droit dans  $[0, 1]^{q'}$  et  $\text{Vol}(\mathbb{Q})$  son volume. C'est une manière de quantifier l'homogénéité de la répartition des points dans l'espace. Muni de cette grandeur, on sait alors que :

$$\left| \int_{[0, 1]^{q'}} f(\mathbf{x}) d\mathbf{x} - \frac{1}{M} \sum_{m=1}^M f(\mathbf{x}_m) \right| \leq V_{HK}(f) D_M \quad (\text{A.21})$$

où  $V_{HK}(f)$  est la variation de Hardy–Krause qui est une généralisation à plusieurs dimensions (relativement peu utilisée de nos jours) de la variation totale d'une fonction réelle. Si cette constante est difficile à estimer, on peut néanmoins influencer sur la valeur de  $D_M$  pour améliorer la borne. Il suffit de construire une suite de points tels que sa discrédance<sup>3</sup> soit faible. La suite de Hammersley est une telle suite, elle produit pour tout  $M$  un ensemble de points de faible discrédance. Elle se définit ainsi, pour tout  $m \in \llbracket 1, M \rrbracket$  :

$$\mathbf{x}_m = \left[ y_{p_1}(m), y_{p_2}(m), \dots, y_{p_{M-1}}(m), \frac{m}{M} \right]^T \quad (\text{A.22})$$

où  $p_1, \dots, p_{M-1}$  est une suite de nombre premiers<sup>4</sup>. Et  $\forall m \in \llbracket 1, M \rrbracket, p \in \{p_m\}_{m=1}^{M-1}$  :

$$y_p(m) = \sum_{k=0}^{H-1} d_k(m) p^{-k} \quad (\text{A.23})$$

où les  $d_k(m)$  sont les coefficients de la décomposition de  $m$  dans la base  $p$ , i.e :

$$m = \sum_{k=0}^{H-1} d_k(m) p^k \quad (\text{A.24})$$

La discrédance de cette suite est bornée par la quantité  $C \frac{(\log M)^{q'-1}}{M}$  d'où l'inégalité suivante :

$$\left| \int_{[0, 1]^{q'}} f(\mathbf{x}) d\mathbf{x} - \frac{1}{M} \sum_{m=1}^M f(\mathbf{x}_m) \right| \leq C V_{HK}(f) \frac{(\log M)^{q'-1}}{M} \quad (\text{A.25})$$

3. On rappelle que c'est un synonyme de divergence (ou disconvenance).

4. On peut aussi choisir une suite de nombres tel que leur PGCD est 1.

Ce taux de convergence est asymptotiquement meilleur que  $O(\frac{1}{M})$ ; mais pour les  $M$  petits cela dépend grandement de la constante  $V_{HK}(f)$ . Pour bénéficier de la potentielle accélération due à cette approximation pour les fonctions  $f$  de la  $\mathbb{S}^{q'-1}$ , [Rehman and Mandic \[2010\]](#) proposent de construire une suite à discrédance faible sur la sphère à partir de la suite d'Hammersley sur le pavé  $[0, 1]^{q'-1}$ . Pour ce faire, ils établissent simplement une bijection linéaire entre le pavé  $[0, 1]^{q'-1}$  et le pavé  $[0, \pi]^{q'-2} \times [0, 2\pi]$  qui est lui-même naturellement en bijection (modulo les problèmes de bords) avec la sphère  $\mathbb{S}^{q'-1}$  (de  $\mathbb{R}^{q'}$ ). Il suffit alors d'appliquer la transformation ainsi définie aux éléments de la suite d'Hammersley pour construire une suite de discrédance faible sur la sphère  $\mathbb{S}^{q'-1}$ . On a utilisé cette procédure pour l'expérience avec l'estimateur de quasi Monte-Carlo relaté en [Table 5.1](#).



# Bibliographie

- S. Akaho. A kernel method for canonical correlation analysis. In *In Proceedings of the International Meeting of the Psychometric Society (IMPS2001)*. Springer-Verlag, 2001.
- G. Andrew, R. Arora, J. Bilmes, and K. Livescu. Deep canonical correlation analysis. volume 28 of *Proceedings of Machine Learning Research*, pages 1247–1255, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.
- K. Appel and W. Haken. Every planar map is four colorable. part i : Discharging. *Illinois J. Math.*, 21(3) :429–490, 09 1977.
- Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.
- R. Arora and K. Livescu. Multi-view cca-based acoustic features for phonetic recognition across speakers and domains. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7135–7139, 2013.
- László Babai. Graph isomorphism in quasipolynomial time. *CoRR*, abs/1512.03547, 2015.
- László Babai. Lectures on graph isomorphism, university of toronto, dept. of computer science, 1979.
- F. R. Bach and M. I. Jordan. A probabilistic interpretation of canonical correlation analysis. Technical report, Department of Statistics, University of California, Berkeley, 2005.
- Alexandru T Balaban. Applications of graph theory in chemistry. *Journal of chemical information and computer sciences*, 25(3) :334–343, 1985.
- Aurélien Bellet, Amaury Habrard, and Marc Sebban. Good edit similarity learning by loss minimization. *Machine Learning*, 89(1-2) :5–35, 2012.
- Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv :1306.6709*, 2013.
- A. Benton, R. Arora, and M. Dredze. Learning multiview embeddings of twitter users. In *Proc. Annual Meeting Assoc. Comput. Linguistics*, volume 2, Aug. 7-12 2016.

- Adrian Benton, Huda Khayrallah, Biman Gujral, Dee Ann Reisinger, Sheng Zhang, and Raman Arora. Deep generalized canonical correlation analysis. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 1–6, Florence, Italy, August 2019.
- Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Hierarchical representation learning in graph neural networks with node decimation pooling, 2019.
- Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 874–883. PMLR, 13–18 Jul 2020a.
- Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 874–883. PMLR, 13–18 Jul 2020b.
- T. D. Bie, B. D. Moor, and K. Arenberg. On the regularization of canonical correlation analysis. 2002.
- Esben Jannik Bjerrum and Richard Threlfall. Molecular generation with recurrent neural networks (rnns). *arXiv preprint arXiv :1705.04612*, 2017.
- Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and radon wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51(1) :22–45, 2015.
- Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21 (suppl\_1) :i47–i56, 2005.
- Gecia Bravo Hermsdorff and Lee Gunderson. A unifying framework for spectrum-preserving graph sparsification and coarsening. In *Advances in NeurIPS 2019*, pages 7736–7747. 2019.
- Y. Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on Pure and Applied Mathematics*, 44 :375–417, 1991.
- Simon Brezovnik, Niko Tratnik, and Petra Žigert Pleteršek. Weighted wiener indices of molecular graphs with application to alkenes and alkadienes. *Mathematics*, 9(2), 2021. ISSN 2227-7390. doi : 10.3390/math9020153.
- Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014)*, CBLIS, April 2014, 2014.
- Horst Bunke and Kim Shearer. A graph distance metric based on the maximal common subgraph. *Pattern recognition letters*, 19(3-4) :255–259, 1998.

- Louis Béthune, Yacouba Kaloga, Pierre Borgnat, Aurélien Garivier, and Amaury Habrard. Hierarchical and unsupervised graph representation learning with loukas’s coarsening. *Algorithms*, 13(9), 2020a. ISSN 1999-4893. doi : 10.3390/a13090206.
- Louis Béthune, Yacouba Kaloga, Pierre Borgnat, Aurélien Garivier, and Amaury Habrard. Apprentissage de représentations hiérarchiques de graphes avec graph2vec et la réduction de loukas. In *Proceedings de CAp (Conférence sur l’Apprentissage automatique)*, NordiCHI, Juin 2020b.
- Tibério S Caetano, Julian J McAuley, Li Cheng, Quoc V Le, and Alex J Smola. Learning graph matching. *IEEE transactions on pattern analysis and machine intelligence*, 31(6) :1048–1058, 2009.
- Nicola De Cao and Thomas Kipf. Molgan : An implicit generative model for small molecular graphs, 2018.
- J Douglas Carroll. Generalization of canonical correlation analysis to three or more sets of variables. In *Proceedings of the 76th annual convention of the American Psychological Association*, volume 3, pages 227–228. Washington, DC, 1968.
- X. Chang, T. Xiang, and T. M. Hospedales. Scalable and effective deep CCA via soft decorrelation. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 1488–1497. IEEE Computer Society, 2018.
- J. Chen, G. Wang, Y. Shen, and G. B. Giannakis. Canonical correlation analysis of datasets with a common source graph. *IEEE Transactions on Signal Processing*, 66(16) :4398–4408, 2018.
- J. Chen, G. Wang, and G. B. Giannakis. Graph multiview canonical correlation analysis. *IEEE Transactions on Signal Processing*, 67(11) :2826–2838, 2019.
- Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.
- Dan C. Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. High-performance neural networks for visual object classification. *CoRR*, abs/1102.0183, 2011.
- R. T. Collins, A. J. Lipton, H. Fujiyoshi, and T. Kanade. Algorithms for cooperative multisensor surveillance. *Proceedings of the IEEE*, 89(10) :1456–1477, 2001.
- Donatello Conte, Pasquale Foggia, and Mario Vento. Challenging complexity of maximum common subgraph detection algorithms : A performance analysis of three algorithms on a wide database of graphs. *J. Graph Algorithms Appl.*, 11(1) :99–143, 2007.
- Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 39(9) : 1853–1865, 2016.

- Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1) :21–27, 1967.
- Marco Cuturi. Sinkhorn distances : Lightspeed computation of optimal transportation distances, 2013.
- Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *CoRR*, abs/1606.09375, 2016.
- Paramveer S Dhillon, Partha Pratim Talukdar, and Koby Crammer. Inference-driven metric learning for graph construction. In *4th North East Student Colloquium on Artificial Intelligence*, 2010.
- Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, Heidelberg ; New York, fourth edition, 2010.
- Florian Dorfler and Francesco Bullo. Kron reduction of graphs with applications to electrical networks. *IEEE Transactions on Circuits and Systems I : Regular Papers*, 60(1) :150–163, Jan 2013. ISSN 1558-0806.
- Michael Drmota and Robert F Tichy. *Sequences, discrepancies and applications*. Springer, 2006.
- Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs, 2021.
- Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *International Conference on Learning Representations*, 2020.
- Rana Fakhreddine. *Stratified Monte Carlo methods for numerical integration and simulation*. Theses, Université de Grenoble, September 2013.
- Mirtha-Lina Fernandez and Gabriel Valiente. A graph distance metric combining maximum common subgraph and minimum common supergraph. *Pattern Recognition Letters*, 22(6-7) : 753–758, 2001.
- P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing of data structures. *IEEE transactions on neural networks*, 9 5 :768–86, 1998.
- Kunihiko Fukushima, Sei Miyake, and Takayuki Ito. Neocognitron : A neural network model for a mechanism of visual pattern recognition. *IEEE transactions on systems, man, and cybernetics*, (5) :826–834, 1983.
- Vikas K. Garg, Stefanie Jegelka, and Tommi S. Jaakkola. Generalization and representational limits of graph neural networks. *CoRR*, abs/2002.06157, 2020.
- Thomas Gärtner, Peter Flach, and Stefan Wrobel. On graph kernels : Hardness results and efficient alternatives. In Bernhard Schölkopf and Manfred K. Warmuth, editors, *Learning Theory and Kernel Machines*, pages 129–143, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.



- Swarnendu Ghosh, Nibaran Das, Teresa Gonçalves, Paulo Quaresma, and Mahantapas Kundu. The journey of graph kernels through two decades. *Computer Science Review*, 27 :88–111, 2018. ISSN 1574-0137. doi : <https://doi.org/10.1016/j.cosrev.2017.11.002>.
- Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry, 2017.
- Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.
- Jacob Goldberger, Geoffrey E Hinton, Sam Roweis, and Russ R Salakhutdinov. Neighbourhood components analysis. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*. MIT Press.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734 vol. 2, 2005. doi : 10.1109/IJCNN.2005.1555942.
- Willem H. Haemers and Edward Spence. Enumeration of cospectral graphs. *European Journal of Combinatorics*, 25(2) :199–211, 2004. ISSN 0195-6698. In memory of Jaap Seidel.
- William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2017a.
- William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs : Methods and applications, 2017b. Published in the IEEE Data Engineering Bulletin, September 2017.
- David K. Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2) :129–150, 2011. ISSN 1063-5203. doi : <https://doi.org/10.1016/j.acha.2010.04.005>.
- Matthieu Heitz, Nicolas Bonneel, David Coeurjolly, Marco Cuturi, and Gabriel Peyré. Ground metric learning on graphs. *Journal of Mathematical Imaging and Vision*, 63(1) :89–107, 2021.
- Harald Andrés Helfgott. Isomorphismes de graphes en temps quasi-polynomial (d’après babai et luks, weisfeiler-leman...), 2017.
- Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *CoRR*, abs/1506.05163, 2015.
- R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019.

- P. Horst. Generalized canonical correlations and their applications to experimental data. *Journal of clinical psychology*, 17 :331–47, 1961a.
- Paul Horst. Relations among sets of measures. *Psychometrika*, 26(2) :129–149, June 1961b. doi : 10.1007/BF02289710.
- H. Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4) :321–377, 1936. ISSN 00063444.
- E. Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik*, 31 :253–258, 1925.
- Elvin Isufi, Andreas Loukas, Andrea Simonetto, and Geert Leus. Distributed time-varying graph filtering. *CoRR*, abs/1602.04436, 2016.
- Linlin Jia, Benoit Gauzère, Florian Yger, and Paul Honeine. A metric learning approach to graph edit costs for regression. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 238–247. Springer, 2021.
- Yacouba Kaloga, Marion Foare, Nelly Pustelnik, and Pablo Jensen. Discrete mumford–shah on graph for mixing matrix estimation. *IEEE Signal Processing Letters*, 26(9) :1275–1279, 2019.
- Yacouba Kaloga, Pierre Borgnat, Sundeep Prabhakar Chepuri, Patrice Abry, and Amaury Habrard. Multiview variational graph autoencoders for canonical correlation analysis. In *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, June 2021a.
- Yacouba Kaloga, Pierre Borgnat, Sundeep Prabhakar Chepuri, Patrice Abry, and Amaury Habrard. Variational graph autoencoders for multiview canonical correlation analysis. *Signal Processing*, 188 :108182, 2021b. ISSN 0165-1684. doi : <https://doi.org/10.1016/j.sigpro.2021.108182>.
- Yacouba Kaloga, Borgnat. Pierre, and Amaury Habrard. A simple way to learn metrics between graph. Technical report, September 2021c.
- Leonid V Kantorovich. On the translocation of masses. In *Dokl. Akad. Nauk. USSR (NS)*, volume 37, pages 199–201, 1942.
- M. Karami and D. Schuurmans. Variational inference for deep probabilistic canonical correlation analysis, 2020.
- Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning, ICML'03*, page 321–328. AAAI Press, 2003. ISBN 1577351894.
- Kristian Kersting, Nils M. Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. Benchmark data sets for graph kernels, 2016. URL <http://graphkernels.cs.tu-dortmund.de>.

- J.R. Kettenring. Canonical analysis of several sets of variables. *Biometrika*, 58(3) :433–451, 12 1971.
- Diederik P Kingma and Jimmy Ba. Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*, 2014.
- D.P. Kingma and M. Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR*, 2014.
- T. N. Kipf and M. Welling. Variational graph auto-encoders. *arXiv :stat.ML*, 1611.07308, 2016a.
- T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016b.
- Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016c.
- Nils M. Kriege, Pierre-Louis Giscard, and Richard Wilson. On valid optimal assignment kernels and applications to graph classification. In *Advances in NeurIPS 2016*, pages 1623–1631. 2016.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
- Sofia Ira Ktena, Sarah Parisot, Enzo Ferrante, Martin Rajchl, Matthew Lee, Ben Glocker, and Daniel Rueckert. Metric learning with spectral graph convolutions on brain connectivity networks. *NeuroImage*, 169 :431–442, 2018. ISSN 1053-8119.
- Luc Le Magoarou, Rémi Gribonval, and Nicolas Tremblay. Approximate fast graph fourier transforms via multilayer sparse approximations. *IEEE transactions on Signal and Information Processing over Networks*, 4(2) :407–420, 2017.
- Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4) : 541–551, 1989. doi : 10.1162/neco.1989.1.4.541.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998.
- Stan Z. Li and Anil Jain, editors. *LDA (Linear Discriminant Analysis)*, pages 899–899. Springer US, Boston, MA, 2009.
- Y. Li, Ming Yang, and Z. Zhang. A survey of multi-view representation learning. *IEEE Transactions on Knowledge and Data Engineering*, 31 :1863–1883, 2019a.
- Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv :1511.05493*, 2015.

- Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *International conference on machine learning*, pages 3835–3845. PMLR, 2019b.
- Lai Pei Ling and Colin Fyfe. Kernel and nonlinear canonical correlation analysis. *International Journal of Neural Systems*, 10(05) :365–377, 2000.
- SP Lloyd. Least square quantization in pcm. bell telephone laboratories paper. published in journal much later : Lloyd, sp : Least squares quantization in pcm. *IEEE Trans. Inform. Theor.*(1957/1982), 18, 1957.
- Gabriel Loaiza-Ganem and John P Cunningham. The continuous bernoulli : fixing a pervasive error in variational autoencoders. *arXiv preprint arXiv :1907.06845*, 2019.
- D. Lopez-Paz, S. Sra, A. Smola, Z. Ghahramani, and B. Schölkopf. Randomized nonlinear component analysis, 22–24 Jun 2014.
- Andreas Loukas. What graph neural networks cannot learn : depth vs width. In *ICLR*, 2020.
- Andreas Loukas and Pierre Vandergheynst. Spectrally approximating large graphs with smaller graphs. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3237–3246. PMLR, 2018.
- Sitao Luan, Mingde Zhao, Xiao-Wen Chang, and Doina Precup. Break the ceiling : Stronger multi-scale deep graph convolutional networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc.
- T. Divya M. Yamuna. Boiling point of alkanes and alkenes - from graph eccentricity. 2018. doi : 10.5958/0974-360X.2018.00364.5. *Research J. Pharm. and Tech* 2018 ; 11(5) :1962-1970.
- Odile Macchi. The coincidence approach to stochastic point processes. *Advances in Applied Probability*, 7(1) :83–122, 1975.
- James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.
- Hermina Petric Margetic, Mireille El Gheche, Giovanni Chierchia, and Pascal Frossard. GOT : an optimal transport framework for graph comparison. *CoRR*, abs/1906.02085, 2019.
- Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In *Advances in NeurIPS 32*, pages 2156–2167. 2019.
- A. Mednykh and I. Mednykh. Isospectral genus two graphs are isomorphic. *Ars Math. Contemp.*, 10 :223–235, 2016.

- Oren Melamud and Jacob Goldberger. Information-theory interpretation of the skip-gram negative-sampling objective function. In *Proceedings of ACL 2017 (Volume 2 : Short Papers)*, Vancouver, Canada, July 2017.
- Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification : Generalizing to new classes at near-zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11) :2624–2637, 2013. doi : 10.1109/TPAMI.2013.83.
- Alessio Micheli. Neural network for graphs : A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3) :498–511, 2009. doi : 10.1109/TNN.2008.2010350.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.
- Gaspard Monge. *Mémoire sur la théorie des déblais et des remblais*. De l’Imprimerie Royale, 1781.
- Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. *CoRR*, abs/1611.08402, 2016.
- William J Morokoff and Russel E Caffisch. Quasi-monte carlo integration. *Journal of computational physics*, 122(2) :218–230, 1995.
- Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural : Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609, 2019.
- Zak Murez, Soheil Kolouri, David Kriegman, Ravi Ramamoorthi, and Kyungnam Kim. Image to image translation for domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4500–4509, 2018.
- Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec : Learning distributed representations of graphs. *arXiv preprint arXiv :1707.05005*, 2017.
- Michel Neuhaus and Horst Bunke. Automatic learning of cost functions for graph edit distance. *Information Sciences*, 177(1) :239–247, 2007.
- Giannis Nikolentzos, Giannis Siglidis, and Michalis Vazirgiannis. Graph kernels : A survey. *arXiv preprint arXiv :1904.12218*, 2019.
- Sonja Nikolić and Nenad Trinajstić. The wiener index : Development and applications. *Croatica Chemica Acta*, 68(1) :105–129, 1995.

- Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan : Training generative neural samplers using variational divergence minimization. In *Advances in NeurIPS 2016*, pages 271–279. 2016.
- Victor Nzobounsana and Sandrine Gaymard. Les analyses canoniques simple et généralisée linéaires : applications à des données psycho-sociales. *Mathématiques et sciences humaines. Mathematics and social sciences*, (189) :69–101, 2010.
- Francesco Orsini, Paolo Frasconi, and Luc De Raedt. Graph invariant kernels. In *Proceedings of IJCAI'15*, page 3756–3762, 2015.
- Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph signal processing : Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5) :808–828, 2018.
- Wolfgang Pauli. Über den zusammenhang des abschlusses der elektronengruppen im atom mit der komplexstruktur der spektren. *Zeitschrift für Physik*, 31(1) :765–783, 1925.
- Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15) :510, 2008.
- Maria MP Petrou and Costas Petrou. *Image processing : the fundamentals*. John Wiley & Sons, 2010.
- Gabriel Peyré and Marco Cuturi. Computational optimal transport, 2020.
- Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8 : 143–195, 1999.
- Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2008.
- Naveed Rehman and Danilo P Mandic. Multivariate empirical mode decomposition. *Proceedings of the Royal Society A : Mathematical, Physical and Engineering Sciences*, 466(2117) :1291–1302, 2010.
- Pau Riba, Andreas Fischer, Josep Lladós, and Alicia Fornés. Learning graph distances with message passing neural networks. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2239–2244. IEEE, 2018.
- Benjamin Ricaud, Pierre Borgnat, Nicolas Tremblay, Paulo Gonçalves, and Pierre Vandergheynst. Fourier could be a data scientist : From graph fourier transform to signal processing on graphs. *Comptes Rendus Physique*, 20(5) :474–488, 2019.
- Kaspar Riesen and Horst Bunke. *Graph classification and clustering based on vector space embedding*, volume 77. World Scientific, 2010.

- Mark Rowland, Jiri Hron, Yunhao Tang, Krzysztof Choromanski, Tamas Sarlos, and Adrian Weller. Orthogonal estimation of wasserstein distances, 2019.
- J. Rupnik, P. Skraba, J. Shawe-Taylor, and S. Guettes. A comparison of relaxations of multiset canonical correlation analysis and applications. *CoRR*, abs/1302.0974, 2013.
- Miyuki Sakai, Kazuki Nagayasu, Norihiro Shibui, Chihiro Andoh, Kaito Takayama, Hisashi Shirakawa, and Shuji Kaneko. Prediction of pharmacological activities from chemical structures with graph convolutional neural networks. *Scientific reports*, 11(1) :1–14, 2021.
- Aliaksei Sandryhaila and José MF Moura. Discrete signal processing on graphs. *IEEE transactions on signal processing*, 61(7) :1644–1656, 2013.
- Ryoma Sato. A survey on the expressive power of graph neural networks. *arXiv preprint arXiv :2003.04078*, 2020.
- Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1) :61–80, 2009. doi : 10.1109/TNN.2008.2005605.
- Firdovsi Sharifov and Hakan Kutucu. *Network Design Problem with Cut Constraints*, pages 265–292. Springer International Publishing, Cham, 2018.
- Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *JMLR*, 12(Sep) :2539–2561, 2011.
- David I. Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. Signal processing on graphs : Extending high-dimensional data analysis to networks and other irregular data domains. *CoRR*, abs/1211.0053, 2012a.
- David I Shuman, Benjamin Ricaud, and Pierre Vandergheynst. A windowed graph fourier transform. In *2012 IEEE Statistical Signal Processing Workshop (SSP)*, pages 133–136. Ieee, 2012b.
- Martin Simonovsky and Nikos Komodakis. Graphvae : Towards generation of small graphs using variational autoencoders. *CoRR*, abs/1802.03480, 2018.
- A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *IEEE transactions on neural networks*, 8 3 :714–35, 1997.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1) :1929–1958, 2014.
- Juan Luis Suárez-Díaz, Salvador García, and Francisco Herrera. A tutorial on distance metric learning : Mathematical foundations, algorithms, experimental analysis, prospects and challenges (with appendices on mathematical background and detailed algorithms explanation). *arXiv preprint arXiv :1812.05944*, 2018.

- Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. Infograph : Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*, 2020.
- Aynaz Taheri. Learning graph representations with recurrent neural network autoencoders. 2018.
- Hiroshi Takahashi, Tomoharu Iwata, Yuki Yamanaka, Masanori Yamada, and Satoshi Yagi. Variational autoencoder with implicit optimal priors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5066–5073, 2019.
- Nicholas C Tang and Ashutosh Chilkoti. Combinatorial codon scrambling enables scalable gene synthesis and amplification of repetitive proteins. *Nature materials*, 15(4) :419–424, 2016.
- Vayer Titouan, Nicolas Courty, Romain Tavenard, Chapel Laetitia, and Rémi Flamary. Optimal transport for structured data with application on graphs. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 6275–6284. PMLR, 2019.
- Matteo Togninalli, Elisabetta Ghisu, Felipe Llinares-López, Bastian Rieck, and Karsten Borgwardt. Wasserstein weisfeiler-lehman graph kernels. In *Advances in Neural Information Processing Systems 32*, pages 6439–6449. 2019.
- Jakub Tomczak and Max Welling. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223. PMLR, 2018.
- Nicolas Tremblay, Paulo Gonçalves, and Pierre Borgnat. Design of graph filters and filterbanks, 2017.
- Anton Tsitsulin, Davide Mottin, Panagiotis Karras, Alexander Bronstein, and Emmanuel Müller. Netlsd. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, Jul 2018. doi : 10.1145/3219819.3219991.
- Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. *CoRR*, abs/1702.05464, 2017.
- Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.
- Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.
- Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *ICLR*, 2019.



- Jorge R. Vergara and Pablo A. Estévez. A review of feature selection methods based on mutual information. *CoRR*, abs/1509.07577, 2015.
- Cédric Villani. *Topics in optimal transportation*. Number 58. American Mathematical Soc., 2003.
- S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *JMLR*, 11(Apr) :1201–1242, 2010.
- Fei Wang and Changshui Zhang. Feature extraction by maximizing the average neighborhood margin. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007. doi : 10.1109/CVPR.2007.383124.
- W. Wang, R. Arora, K. Livescu, and J. Bilmes. On deep multi-view representation learning. volume 37 of *Proceedings of Machine Learning Research*, pages 1083–1092, Lille, France, 07–09 Jul 2015. PMLR.
- Weiran Wang, H. Lee, and Karen Livescu. Deep variational canonical correlation analysis. *ArXiv*, abs/1610.03454, 2016.
- Pauline Wauquier and Mikaela Keller. Metric learning approach for graph-based label propagation. *arXiv preprint arXiv :1511.05789*, 2015.
- Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(9) :207–244, 2009.
- Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.
- David P Williamson. *Network flow algorithms*. Cambridge University Press, 2019.
- TA Witten Jr and Leonard M Sander. Diffusion-limited aggregation, a kinetic critical phenomenon. *PRL*, 47(19) :1400, 1981.
- Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.
- Eric Xing, Michael Jordan, Stuart J Russell, and Andrew Ng. Distance metric learning with application to clustering with side-information. *Advances in neural information processing systems*, 15 :521–528, 2002.
- Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.
- Y. Yamanishi, J.-P. Vert, A. Nakaya, and M. Kanehisa. Extraction of correlated gene clusters from multiple genomic data by generalized kernel canonical correlation analysis. *Bioinformatics*, 19(suppl. 1) :i323–i330, 07 2003.

- 
- Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in NeurIPS 2018*, pages 4800–4810. 2018.
- Tomoki Yoshida, Ichiro Takeuchi, and Masayuki Karasuyama. Distance metric learning for graph structured data. *Machine Learning*, 110(7) :1765–1811, Jun 2021. ISSN 1573-0565.
- Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. Graph transformer networks, 2020.
- Jiawei Zhang. Graph neural distance metric learning with graph-bert. *arXiv preprint arXiv :2002.03427*, 2020.
- Qi Zhao and Yusu Wang. Learning metrics for persistence-based summaries and applications for graph classification. *CoRR*, abs/1904.12189, 2019.
- Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks : A review of methods and applications, 2021.
- Yuehua Zhu, Muli Yang, Cheng Deng, and Wei Liu. Fewer is more : A deep graph metric learning perspective using fewer proxies. *arXiv preprint arXiv :2010.13636*, 2020.