université djillali Bounaama
Khemis Miliana

département mathematique et informatique

Thème

Utilisation des GCNN pour l'optimisation de recherche dans les algorithmes FSM

Encadrer par :

Mr.Oussama harbouche

Realisé par :

fatiha kerrache

amine bechaa

31 décembre 2020

Résumé

L'apprentissage profond (Deep Learning)sur les graphes est une tâche importante avec des applications allant de la conception de médicaments à la recommandation d'amitié dans les réseaux sociaux. Le principal défi dans ce domaine est de trouver un moyen de représenter ou de coder la structure d'un graphe afin qu'elle puisse être facilement exploitée par des modèles d'apprentissage en profondeur.Parmis les approches relier au deep learning en graphe, on trouve le sous-graphe fréquemment exploité (frequent sub graph mining) FSM, relier avec l'un des domaines de deep learning : les réseaux convolutifs de graphes(graph convolutional noronal netwoks) GCNN l'objet de notre travail c'est l'utilisation d' une stratégie de recherche efficace pour identifier les modèles de sous-graphes de réseau qui apparaissent le plus fréquemment dans un ensemble de données de graphes cibles. Pour cela on a réutilisé une approche SPMiner (Subgraph Pattern Miner) qui est un outil général pour trouver des sous-graphes et intègre des réseaux de neurones graphiques (GNN).

.

.

.

ملخص

يعد التعلم العميق على الرسوم البيانية مهمة كبيرة مع تطبيقات تتراوح من تصميم الأدوية إلى توصيات الصداقة عبر وسائل التواصل الاجتماعي. يتمثل التحدي الرئيسي في هذا المجال في إيجاد طريقة لتمثيل أو ترميز بنية الرسم البياني بحيث يمكن استغلاله بسهولة بواسطة نماذج التعلم العميق. من بين الأساليب المتعلقة بالتعلم العميق في الرسوم البيانية ، يجد الرسم البياني الفرعي الذي يتم استغلاله بشكل متكرر (تنقيب الرسم البياني الفرعي المتكرر) ، لربطه بأحد مجالات التعلم العميق : الشبكات التلافيفية للرسوم البيانية (الرسم البياني للشبكات النورونية التلافيفية). هدف عملنا هو استخدام استراتيجية بحث فعالة لتحديد أنماط الرسم البياني الفرعي للشبكة التي تظهر بشكل متكرر في مجموعة بيانات الرسم البياني المستهدف. لهذا قمنا بإعادة استخدام نهج (اسبي ماينر)وهو أداة عامة للعثور على الرسوم البيانية الفرعية ودمج الشبكات العصبية الرسومية

.

.

abstract

Deep learning on graphs is a big task with applications ranging from drug design to social media friendship recommendation. The main challenge in this field is to find a way to represent or code the structure of a graph so that it can be easily exploited by deep learning models. Among the approaches relating to deep learning in graphs, we find the frequently exploited subgraph (frequent sub graph mining) FSM, to

link with one of the fields of deep learning : convolutional networks of graphs (graph convolutional noronal netwoks) GCNN the object of our work is to use an effective search strategy to identify the network subgraph patterns that appear most frequently in a target graph dataset. For this we have reused an SPMiner (Subgraph Pattern Miner) approach which is a general tool for finding subgraphs and integrates graphic neural networks (GNN).

Table des matières

1 Les Algorithme FSM **4**

1.1 Introduction 4

1.2 Théorie des graphes : 5

1.2.1 Quelques définitions : 5

1.2.1.1 Graphe : 5

1.2.1.2 Un graphe orienté : 6

1.2.1.3 Boucle, chaîne, chemin, cycle et circuit : 7

1.2.1.4 Quelques types de graphes : 8

1.2.1.4.1 Graphe simple : 8

1.2.1.4.2 Multi-graphe : 8

1.2.1.4.3 graphe connexe et non connexe : 9

1.2.1.4.4 Graphe complet ou clique : . 9

1.2.1.4.5 Graphe biparti : 10

1.2.1.4.6 Graphe planaire : 10

1.2.1.4.7 Graphe étiqueté : 10

1.2.1.4.8 Sous-graphe : 10

1.2.1.4.9 Graphe partiel : 11

1.2.1.5 Arbre et de foret : 12

1.2.1.6 Parcours de graphes : 12
1.2.1.7 Représentation de graphe : 13
1.2.1.7.1 Matrices d'incidence et d'adjacence : 13
1.2.1.7.2 Liste d'adjacence : 14
1.2.2 Notions de base sur l'appariement de graphes : . 16
1.2.2.1 Isomorphisme et homomorphisme : . . 16
1.2.2.1.1 Homomorphisme de graphes : 16
1.2.2.1.2 Isomorphisme de graphes : . . 17
1.2.2.1.3 Isomorphisme de sous-graphes : 17
1.2.2.2 Complexité d'appariement de graphes : 17
1.2.3 Notions de similarité entre graphes : 18
1.2.3.1 La distance d'édition de graphes : . . . 18
1.2.3.2 La distance basée sur la notion de sous-graphe commun maximal (SCM) : . . . 19
1.2.3.3 Autres distances pour calculer la similarité entre graphes : 20
1.3 Frequent sub-graphe mining (fsm) : 21
1.3.1 Le problème de découverte des sous-graphes connexes fréquent : 21
1.3.2 Les problèmes de découverte des sous-graphes fréquents : 23
1.3.3 La génération des sous-graphes connexes candidats : 23
1.3.3.1 L'augmentation : 24

1.3.3.2 La jonction : 26
1.3.3.3 La vérification de fréquence : 27
1.3.3.4 Le problème d'isomorphisme de sous-graphes : 28
1.3.4 Les approches des algorithmes FSM : 29
1.3.4.1 Propriété Apriori : 30
1.3.4.2 Approche basée sur Apriori : 30
1.3.4.3 Approche PatternGrowthGraph : . . . 32
1.3.5 Frequent subgraph mining algorithms : 33
1.3.6 FSM inexact : 35
1.3.7 FSM exact : 35
1.4 Conclusion : 36

2 L'IA,Machine Learning,Deep Learning 37
2.1 Introduction : 37
2.2 L'intelligence artificielle 38
2.2.1 Définition : 38
2.2.2 Domaine de l'IA : 38
2.2.2.1 Le symbolisme : 38
2.2.2.2 Le connexionnisme : 39
2.2.2.3 Le comportementalisme : 39
2.2.3 Segmentation : 39
2.2.3.1 Les solutions : 40
2.2.3.2 Les outils : 40
2.2.3.3 Les techniques : 40
2.2.3.4 Les données : 40

2.2.4 Données de l'IA : 41
2.2.4.1 Type de données : 42
2.2.4.1.1 Données d'entraînement : . . 43
2.2.4.1.2 Données de test : 44
2.2.4.1.3 Données de production : . . . 45
2.2.4.1.4 Données de renforcement : . . 45
2.2.4.2 Origine des données : 46
2.3 Machine Learning : 46
2.3.1 Définition : 46
2.3.2 Catégories de machine Learning : 47
2.3.2.1 L'apprentissage supervisé : 47
2.3.2.1.1 Classification 48
2.3.2.1.2 Régression : 49
2.3.2.2 L'apprentissage non supervisé : 50
2.3.2.2.1 Clustering : 50
2.3.2.2.2 Réduction de la dimensionnalité : 51
2.3.2.3 L'apprentissage par renforcement : . . 52
2.3.3 Outils du machine Learning : 53
2.4 Deep Learning : 54
2.4.1 Définition : 54
2.4.2 Principe de Deep Learning : 55
2.4.3 Evolutions du deep learning : 56
2.4.4 Modes d'apprentissage de Deep Learning : . . . 57
2.4.4.1 L'apprentissage supervisé : 57

2.4.4.2 L'apprentissage non supervisé : 58
2.4.4.3 L'apprentissage par renforcement : . . 59
2.4.5 Outils du deep learning : 59
2.5 Conclusion : 61

3 Les méthodes de GCNN 62
3.1 Introduction : 62
3.2 Domaine de données pour ConvNet : 63
3.3 Architecture de graphe convNet 64
3.4 Euclidean ConvNets : 64
3.4.1 Architecture : 64
3.4.2 couches compositionnelles : 64
3.4.3 Domaine de données pour ConvNets : 65
3.5 ConvNets spectrales pour les graphes fixes : 67
3.5.1 Théorie des graphes spectrales : 67
3.5.1.1 Les graphes : 67
3.5.1.2 Graph Laplacian : 67
3.5.1.3 Décomposition spectrale : 68
3.5.1.4 Modes de Fourier : 69
3.5.1.5 Analyse de Fourier : 70
3.5.1.5.1 Analyse de Fourier euclidienne : 70
3.5.1.5.2 Analyse de Fourier sur les graphes : 70
3.5.1.6 Convolution : 71
3.5.1.6.1 Convolution dans un espace euclidien discret : 71
3.5.1.6.2 Convolution sur les graphes . 71

3.5.1.7 Fast graph pooling : 72
3.5.2 Spectral ConvNets : 73
3.5.2.1 SplineNets : 73
3.5.2.1.1 graphe spectral vanille ConvNets : 73
3.5.2.1.2 Convolution du graphe spectral : 73
3.5.2.1.3 SplineNets : 74
3.5.2.1.4 localisation spatial et Lissage spectral 75
3.5.2.2 ChebNets : 75
3.5.2.2.1 Filtres polynomiaux spectraux : 75
3.5.2.2.2 Graphe spectral ConvNets avec filtres polynomiaux : 76
3.5.2.2.3 Chebyshev polynomials : . . 77
3.5.2.2.4 ChebNets : 77
3.5.2.2.5 Graph convolutional nets ChebNets simplifiés : 78
3.6 ConvNets pour Variable Graphs : 79
3.6.1 ConvNets pour Variable Graphes et fixe graphes : 79
3.6.2 Graph neural networks : 79
3.6.3 Graph RNNs : 80
3.6.4 Graph ConvNets : 82
3.7 Conclusion : 83

4 GCNN/FSM **84**

4.1 Introduction : 84
4.2 les outils de l'implementation : 85
4.2.1 Google colab : 85
4.2.1.1 Collaboration facile : 85
4.2.1.2 Bibliothèques préinstallées : 85
4.2.1.3 GPU / TPU gratuit : 85
4.3 Motivation : 86
4.4 Analise du problème : extraction fréquente de sous-graphes : 87
4.4.1 frequence de sous graphe : 88
4.4.2 Définition 1 niveau de graphe 88
4.4.2.1 Définition 2 au niveau du nœud 88
4.4.3 Approches actuelles de l'extraction de sous-graphes : 89
4.5 Notre travail : 90
4.5.1 Notre approche SPMiner : 91
4.5.1.1 Encodeur SPMiner : 92
4.5.1.2 Encodeur SPMiner : architecture GNN : 92
4.5.1.3 SPMiner : commande d'espace d'intégration : 95
4.5.1.4 Procédure de recherche SPMiner : . . . 95
4.6 Expérience : 97
4.6.1 Expérimentations de petits motifs : 98
4.6.2 Expérience : grands motifs : 99
4.7 Conclusion : 102

Table des figures

1.1 Un graphe non oriente à 8 sommets 6
1.2 Un graphe orienté à 4 sommets 7
1.3 Exemple de chemin et cycle 8
1.4 Exemple d'un multi-graphe 9
1.5 Exemple de graphe non connexe possédant 2 composantes connexes. 9
1.6 Exemple de graphe biparti (a) - Exemple de graphe planaire (b) 10
1.7 Exemple de sous graphe et graphe partiel. 11
1.8 Exemple d'arbre et de forêt 12
1.9 Exemple de représentation d'un graphe par sa matrice d'adjacence 13
1.10 Représentation d'un graphe simple orienté par sa matrice d'incidence. 14
1.11 Liste d'adjacence d'un graphe 15
1.12 Liste d'adjacence d'un graphe étiqueté 15
1.13 Un exemple d'une séquence d'opérations d'édition pour transformer G1 à G2 19

1.14 Exemple de sous graphe maximum commun entre deux graphes. 20
1.15 Exemple d'augmentation d'un sous-graphe par l'ajout d'un sommet. 25
1.16 Exemple d'augmentation d'un sous-graphe par l'ajout d'une arête. 26
1.17 Exemple de jonction de deux graphes connexes à base de sommets. 28
1.18 Exemple d'isomorphisme de sous-graphes. 29
1.19 Algorithme apriori 31
1.20 Approche basé sur apriori 31
1.21 Algorithme de croissance de modèle 32
1.22 Approche basé sur la croissance de modèle 33

2.1 les données de l'IA 41
2.2 les types de données 42
2.3 régression linéaire et non linéaire 49
2.4 choix d'un modèle 51
2.5 catégorie de machine Learning 53
2.6 Outils de machine Learning 54
2.7 principe de Deep Learning 55
2.8 perceptron de rosenblatt 57
2.9 Outils de Deep Learning 60

3.1 Standard ConvNets Spectral graph ConvNets de domaine fixe et variable 63
3.2 architecture de graphe convNet 64

3.3 architecture d'apprentissage de dimension élevées . . . 65
3.4 couches compositionnelle 65
3.5 Domaine de données Euclidien de ConvNet 66
3.6 Domaine de données non Euclidien de ConvNet 66
3.7 un graphe 68
3.8 premier eigenvectors de 1D Euclidean laplacien =base standard de fourier 69
3.9 Premier Laplacien eigenvectors d'un graphe 70
3.10 Couche convolutionelle 73
3.11 filtre du graphe spatial 74
3.12 filtre de graphe spectral 74
3.13 localisation spatial et fonction de filtre de smooth spectral 75
3.14 application d'un filtre spectral au feature signal f . . . 76
3.15 Chebyshev polynomials 78
3.16 Brain connectivity network (sMRI) Fixed graph G . . 80
3.17 Molecules Variable graphs Gk 80
3.18 graphe arbitraire 81
3.19 graphe variable 81
3.20 Vanilla graph ConvNets 82

4.1 modèles d'interaction des protéines dans les voies de désaccord 86
4.2 modèles d'interaction dans les réseaux sociaux 87
4.3 groupes fonctionnels pour les ensembles de données moléculaires 87

4.4 fréquence de sous graphe au niveau graphe 88
4.5 fréquent de sous graphe au niveau de nœuds 89
4.6 nœuds encodé 90
4.7 Approche SPMiner 91
4.8 Encodeur SPMiner 92
4.9 Encodeur SPMiner : architecture GNN 93
4.10 structure de graph neural networks à deux tours 94
4.11 SPMiner : commande d'espace d'intégration 95
4.12 Procédure de recherche SPMiner(1) 96
4.13 Procédure de recherche SPMiner(2) 97
4.14 Procédure de recherche SPMiner dans un espace d'integration 98
4.15 Expérimentations de petits motifs 100
4.16 Expérience : grands motifs(1). 101
4.17 Expérience : grands motifs(2). 101

Introduction générale

La recherche de motifs de sous-graphes ou de motifs de réseaux qui se reproduisent fréquemment dans un jeu de données de graphes est importante pour identifier les propriétés structurelles interprétables de réseaux complexes. Exemples d'applications : Biologie : le comptage de sous-graphes est hautement prédictif pour les voies de la maladie, l'interaction des gènes et les connectome , Sciences sociales : les schémas sous-graphiques sont des indicateurs de l'équilibre et du statut social, Chimie : les sous-structures communes sont essentielles pour prédire les propriétés moléculaires Cependant, l'extraction fréquente de sous-graphes a une complexité de calcul extrêmement élevée. Une approche traditionnelle de l'extraction de motifs consiste à énumérer tous les motifs possibles Q d'une taille allant jusqu'à k, puis à compter les apparitions de Q dans un ensemble de données donné. Le comptage fréquent de sous-graphes est très difficile en termes de calcul car il nécessite la résolution de deux problèmes de recherche insolubles :

(1) Comptage de la fréquence d'un motif Q donné dans G.

(2) Recherche sur tous les motifs possibles Q pour identifier les motifs fréquents.

Le problème (1) est NP-difficile, tandis que le problème (2) est

également difficile car le nombre de graphes possibles Q augmente de façon super-exponentielle avec leur taille. SPMiner est la première approche neuronale à identifier approximativement les sous-graphes les plus fréquents, surpassant les heuristiques existantes et les algorithmes d'approximation basés sur la recherche.l'objet de notre travail c'est l'utilisation d' une stratégie de recherche efficace pour identifier les modèles de sous-graphes de réseau qui apparaissent le plus fréquemment dans un ensemble de données de graphes cibles. Pour cela on a réutilisé une approche SPMiner (Subgraph Pattern Miner) qui est un outil général pour trouver des sous-graphes et intègre des réseaux de neurones graphiques (GNN). SPMiner (Subgraph Pattern Miner) est un outil général pour trouver des sous-graphiques fréquents dans un grand graphe cible FSM , et une stratégie de recherche efficace pour identifier les modèles de sous-graphiques de réseau qui apparaissent le plus fréquemment dans un ensemble de données de graphiques cibles. Le mémoire est organisé en quatre chapitres comme suit :

Chapitre01 : on va définir les notions de base liées aux graphes et à la théorie des graphes. Et on a présenté des algorithmes et des techniques d'exploration de sous-graphes fréquents (FSM).

Chapitre02 : est composé on trois parties. Partie 01 : on va définir les notions de base de l'intelligence artificielle.

Partie 02 : on on va définir les déférents concepts de machine Learning.

Partie 03 : se présente principalement les défirents concepts de Deep Learning.

Chapitre 03 : Ce chapitre définit les déférentes méthodes de GCNN graphe convolutional noronal networks : Euclidean convNet , les concepts de ConvNets spectrales pour les graphes fixes et les concepts de ConvNets spatial pour Variable Graphs.

Chapitre 04 : on va définir les outils de l'implementation, , la configuration de notre broblème avec les approches actuelles de l'extraction de sous-graphes FSM ,ensuite on va détailler l'approche qu'on va utiliser dans notre travail (SPminer)avec l'architecture GCNN et on va terminer notre chapitre avec la partie d'experience.

Chapitre 1

Les Algorithme FSM

1.1 Introduction

Ce chapitre définit en détail le concept de l'extraction de graphes et se concentre principalement sur l'exploration de sous-graphes fréquent (FSM), il est organisé en deux parties :

La partie 01 : est principalement consacrée à présenter, de manière simplifiée, les notions de base liées aux graphes et à la théorie des graphes.

La partie 02 : se concentre sur la présentation de «l'état de l'art» des algorithmes et des techniques d'exploration de sous-graphes fréquents (FSM). Un aperçu des recherches actuelles dans le domaine et des solutions pour résoudre les principaux problèmes de recherche sont également présentés

1.2 Théorie des graphes :

L'histoire de la théorie des graphes débute avec les travaux de Léonard Euler avec le problème des ponts de Königsberg. Euler a modélisé ce problème par un graphe : Un sommet est associé à chaque parcelle de terre délimitée par la rivière et une arête est associée à chaque pont les reliant (cf. ref. [32]). Depuis, cette théorie a donné naissance à de multiples techniques dans de nombreux domaines, en mathématique, chimie, physique, biologie, informatique, etc. Cette théorie a deux principaux objectifs : Le premier est d'offrir un modèle pour représenter le problème, généralement l'ensemble des possibilités. Dans ce cas, le graphe est une représentation des différents éléments et des relations binaires entre eux. Le deuxième objectif de cette théorie est d'offrir des outils pour permettre de trouver les meilleures possibilités dans le but de résoudre un problème (cf. ref. [45]) . Dans notre contexte, nous nous intéressons aux outils qui permettent d'apparier deux graphes.

1.2.1 Quelques définitions :

1.2.1.1 Graphe :

Un graphe fini G = (V, E) est défini par l'ensemble fini V =v1, v2, . . . , vn dont les éléments sont appelés sommets (vertices en anglais), et par l'ensemble fini E=e1, e2, . . . , em dont les éléments sont appelés arêtes (edges en anglais). Une arête e de l'ensemble E est définie par une paire non ordonnée de sommets, appelés les extrémités de e. Si l'arête e relie les sommets a et b, on dira que ces sommets sont adjacents, ou

incidents avec e, ou bien que l'arête e est incidente avec les sommets a et b. Le graphe de la figure 1.1 représente un graphe non orienté à 8 sommets, nommés a à h, comportant 10 arêtes. Ce graphe est définit de la façon suivante : G= (V, E) V= a, b, c, d, e, f, g, h E= (a, d), (b, c), (b, d), (d, e), (e, c), (e, h), (h, d), (f, g), (d, g), (g, h) .(cf. ref. [61])(cf. ref. [26])(cf. ref. [33])(cf. ref. [57])

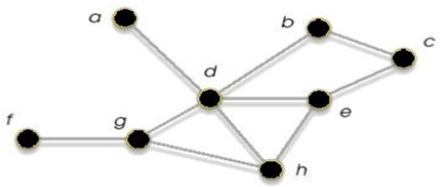


FIGURE 1.1: Un graphe non oriente à 8 sommets

1.2.1.2 Un graphe orienté :

Est un graphe dont les arêtes sont orientées et par conséquent, le couple (x, y) est différent du couple (y, x) puisque leur orientation est différente. Les arcs d'un graphe orienté, sont représentés sous forme de flèches. On les appelle arcs pour les distinguer des arêtes non orientées. (cf. ref. [61])(cf. ref. [26])(cf. ref. [33])(cf. ref. [57])

Ordre, degré et distance d'un graphe : On appelle ordre d'un graphe le nombre de sommets n de ce graphe, il est noté |X|. Le degré d'un sommet x d'un graphe est le nombre d'arêtes incidentes à x. Il est noté d(x). Pour un graphe simple le degré de x correspond également au nombre de sommets adjacents à x. La distance entre deux sommets u et v est le nombre d'arêtes dans le plus court (la plus petite longueur) chemin entre u et v (cf. ref. [32]) (b)

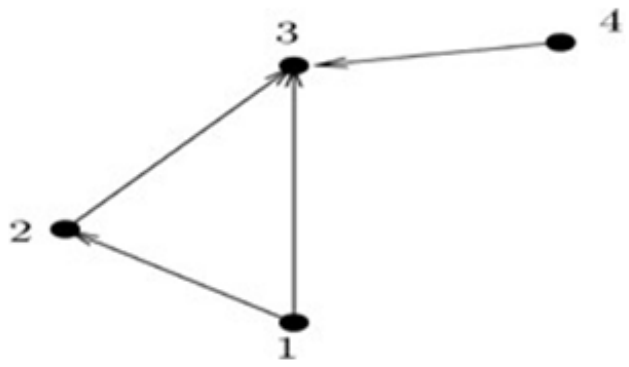


FIGURE 1.2: Un graphe orienté à 4 sommets

1.2.1.3 Boucle, chaîne, chemin, cycle et circuit :

Une arête dont les extrémités sont identiques est appelée une boucle. Une chaîne simple ou élémentaire est une chaîne ne pouvant passer qu'une seule fois au maximum par arc et par conséquent, ayant une longueur maximale égale au nombre d'arcs d'un graphe. Un chemin est une chaîne dont tous les arcs sont orientés dans le même sens. La figure 1.3 montre deux exemples de chemin, le chemin simple (f, g, d, b) en rouge et le chemin (f, g, d, h, e, d, b) qui n'est pas simple car le sommet d est visité deux fois. Un cycle ou circuit est une suite d'arcs partant et finissant au même sommet. Notons également qu'un cycle simple sera automatiquement une chaîne simple, dans la figure 1.3, le chemin (d, h, e, d) en vert représente un cycle.(cf. ref. [61])(cf. ref. [26])(cf. ref. [33])(cf. ref. [57]). Remarque : Les termes ' chemin' et 'circuit' s'emploient en propre pour les graphes orientés. Pour les graphes non orientés, on parle de chaine et de cycle. Cependant la définition formelle est exactement la même dans les deux cas, seule change la structure (graphe orienté ou non) sur laquelle ils sont définis.

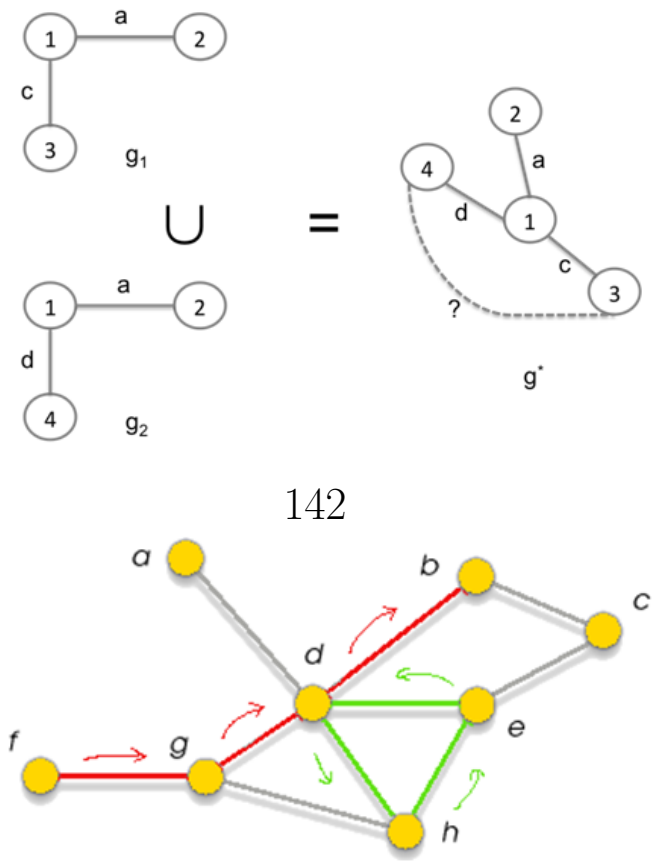


FIGURE 1.3: Exemple de chemin et cycle

1.2.1.4 Quelques types de graphes :

Il existe plusieurs familles de graphes, nous citons ci-après, quelques familles simples et importantes de graphes, notons que le choix d'un type de graphe dépend du problème à résoudre :

1.2.1.4.1 Graphe simple : Un graphe est simple si au plus une arête relie deux sommets et s'il n'y a pas de boucle sur un sommet.(cf. ref. [61])(cf. ref. [26])(cf. ref. [33])(cf. ref. [57]).

1.2.1.4.2 Multi-graphe : On peut imaginer des graphes avec une arête qui relie un sommet à lui-même (une boucle), ou plusieurs

arêtes reliant les deux mêmes sommets. On appelle ces graphes des multi-graphes. (cf. ref. [61])(cf. ref. [26])(cf. ref. [33])(cf. ref. [57]).

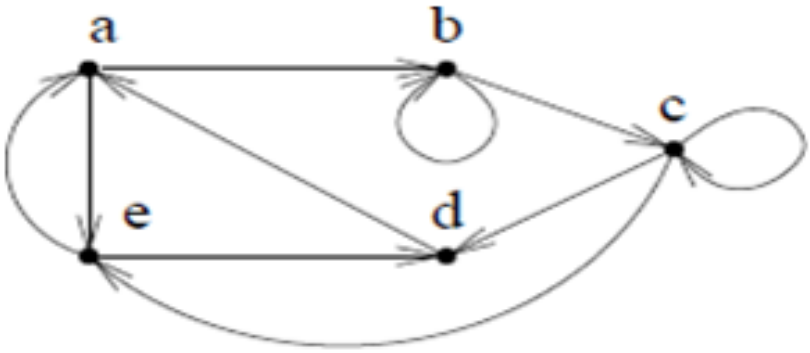


FIGURE 1.4: Exemple d'un multi-graphe

1.2.1.4.3 graphe connexe et non connexe : Un graphe est connexe s'il est possible, à partir de n'importe quel sommet, de rejoindre tous les autres en suivant les arêtes. Un graphe non connexe se décompose en composantes connexes.

FIGURE 1.5: Exemple de graphe non connexe possédant 2 composantes connexes.

1.2.1.4.4 Graphe complet ou clique : Un graphe est complet si chaque sommet du graphe est relié directement à tous les autres sommets. (cf. ref. [61])(cf. ref. [26])(cf. ref. [33])(cf. ref. [57]).

1.2.1.4.5 Graphe biparti : Un graphe est biparti si ses sommets peuvent être divisés en deux ensembles X et Y, de sorte que toutes les arêtes du graphe relient un sommet dans X à un sommet dans Y comme illustré dans la figure 1.6 (a) : (cf. ref. [61])(cf. ref. [26])(cf. ref. [33])(cf. ref. [57]).

1.2.1.4.6 Graphe planaire : Un graphe planaire est un graphe qui a la particularité de pouvoir se représenter dans un plan sans qu'aucune arête, courbe ou rectiligne, ne se croise (voir l'exemple de la figure 1.6 (b).(cf. ref. [**?**])(cf. ref. [**?**])(cf. ref. [**?**])(cf. ref. [**?**]).

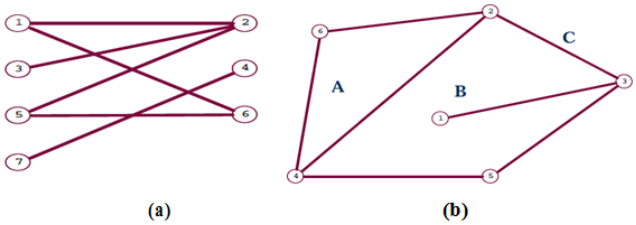


FIGURE 1.6: Exemple de graphe biparti (a) - Exemple de graphe planaire (b) .

1.2.1.4.7 Graphe étiqueté : Un graphe étiqueté est un graphe dont les arêtes et/ou les nœuds sont affectés d'étiquettes. (cf. ref. [61])(cf. ref. [26])(cf. ref. [33])(cf. ref. [57]).

1.2.1.4.8 Sous-graphe : Un sous graphe est une partie d'un graphe. Toutefois, dans un sous-graphe, il n'est pas obligatoire que toutes les parties du graphe principal y apparaissent. Dans le cas où le

sous-graphe est parfait, c'est-à-dire qu'il représente fidèlement et complètement une partie du graphe, ce sous-graphe est appelé sous-graphe engendré. Autrement dit, un sous-graphe de G est un graphe H= (V ; A(V)) tel que V est un sous ensemble de X, et A(V) sont les arêtes induites par A sur V, c'est-à-dire les arêtes de A dont les deux extrémités sont des sommets de V.(cf. ref. [61])(cf. ref. [26])(cf. ref. [33])(cf. ref. [57]).

1.2.1.4.9 Graphe partiel : Soit G=(E,V) un graphe. Le graphe G1= (E1,V1) est un graphe partiel de G, si V1 est inclus dans V. Autrement dit, on obtient G1 en enlevant une ou plusieurs arêtes du graphe G. Etant donné le graphe G=(E,V) de la figure 1.7 où e=1, 2, 3, 4, 5, 6 et V= (1,3), (1,4), (1,5), (1,6), (2,3), (3,5) ,(5,6). Le graphe partiel G1= (E1,V1) de G est induit par les sommets E1=1, 2, 3, 4, 5, 6 et les arrêtes V1= (1,3), (1,4), (1,6), (2,3),(5,6). Le sous-graphe G2= (E2,V2) de G est induit par les sommets E2=1, 2, 4, 6 et les arrêtes V2=(1,3),(1,4),(1,6)(cf. ref. [61])(cf. ref. [26])(cf. ref. [33])(cf. ref. [57]).

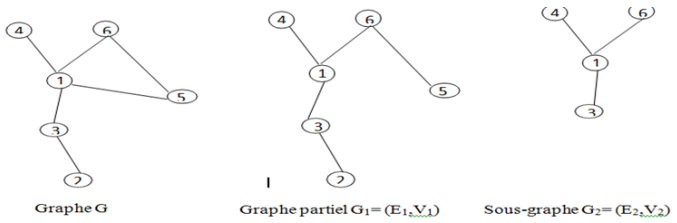


FIGURE 1.7: Exemple de sous graphe et graphe partiel.

1.2.1.5 Arbre et de foret :

On appelle arbre tout graphe connexe sans cycle. Un graphe sans cycle mais non connexe est appelé une forêt. Figure 1. 6 Exemple de sous graphe et graphe partiel(cf. ref. [61])(cf. ref. [26])(cf. ref. [33])(cf. ref. [57])..

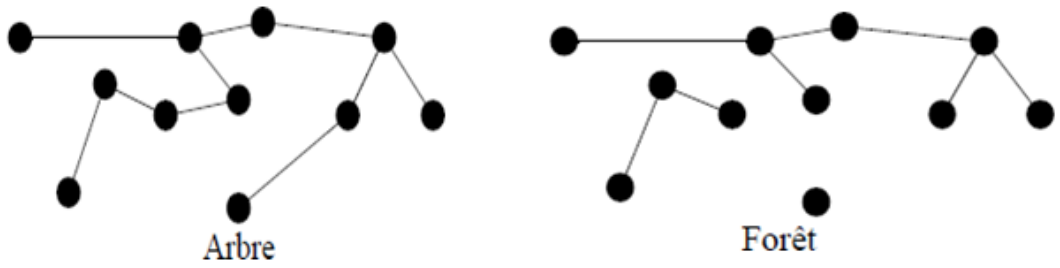


FIGURE 1.8: Exemple d'arbre et de forêt

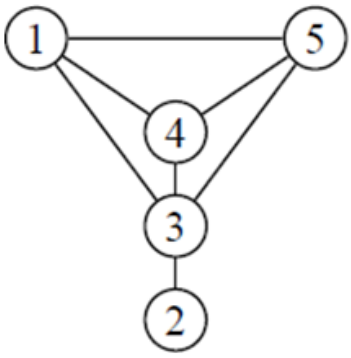
1.2.1.6 Parcours de graphes :

Beaucoup de problèmes sur les graphes nécessitent que l'on parcourt l'ensemble des sommets et des arcs/arêtes du graphe. Deux principales stratégies d'exploration existent : Le parcours en largeur consiste à explorer les sommets du graphe niveau par niveau, à partir d'un sommet donné ; Le parcours en profondeur consiste, à partir d'un sommet donné, à suivre le chemin le plus loin possible puis à faire des retours en arrière pour reprendre tous les chemins ignorés précédemment. (cf. ref. [61])(cf. ref. [26])(cf. ref. [33])(cf. ref. [57]).

1.2.1.7 Représentation de graphe :

Tout graphe peut être représenté par une matrice d'incidence et d'adjacence ou par une liste d'adjacence.

1.2.1.7.1 Matrices d'incidence et d'adjacence : Bien que le dessin soit un moyen pratique pour définir un graphe, il n'est pas adapté ni au stockage des graphes dans la mémoire d'un ordinateur, ni à l'application des méthodes mathématiques pour étudier leurs propriétés. Pour ces usages, il existe deux matrices associées à un graphe : sa matrice d'incidence et sa matrice d'adjacence. On peut représenter un graphe simple par une matrice d'adjacences. Une matrice (nm) est un tableau de n lignes et m colonnes. (i, j) désigne l'intersection de la ligne i et de la colonne j. Dans une matrice d'adjacences, les lignes et les colonnes représentent les sommets du graphe. Un « 1 » à la position (i, j) signifie que le sommet i est adjacent au sommet j.



$$M = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

FIGURE 1.9: Exemple de représentation d'un graphe par sa matrice d'adjacence

Cette matrice a plusieurs caractéristiques : 1. Elle est carrée : il y a autant de lignes que de colonnes. 2. Il n'y a que des zéros sur la

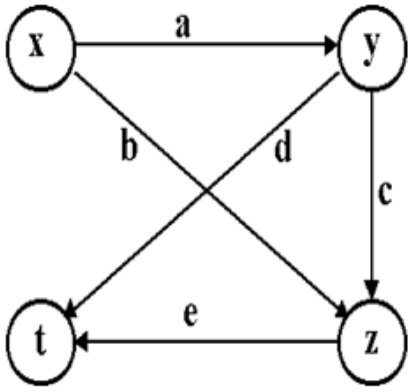
diagonale allant du coin supérieur gauche au coin inférieur droit. Un « 1 » sur la diagonale indique qu'il existe une boucle. 3. Elle est symétrique : mij =mji. On peut dire que la diagonale est un axe de symétrie. 4. Une fois que l'on fixe l'ordre des sommets, il existe une matrice d'adjacences unique pour chaque graphe. La matrice d'incidence sommets-arcs d'un graphe G est la matrice M = (mij) de taille nm, définie comme suit :

mij

* 1si i est le sommet initoal de j,

*-1si i est le sommet terminal de j,

* 0 dans les autres cas. (cf. ref. [61])(cf. ref. [26])(cf. ref. [33])(cf. ref. [57]).



	a	b	c	d	e
x	+1	+1	0	0	0
y	-1	0	+1	+1	0
z	0	-1	-1	0	+1
t	0	0	0	-1	-1

FIGURE 1.10: Représentation d'un graphe simple orienté par sa matrice d'incidence.

1.2.1.7.2 Liste d'adjacence : Une autre idée pour représenter les graphes en mémoire : les listes d'adjacences. Dans ce type de structure, on a un tableau de liste de sommets. Le tableau comporte autant de cases qu'il y a de sommets. Chacune des cases pointe vers une liste de sommets. Cette liste n'est rien d'autre que les successeurs du som-

met considéré. La figure 1.11 montre un exemple de graphe et sa liste d'adjacence associée :

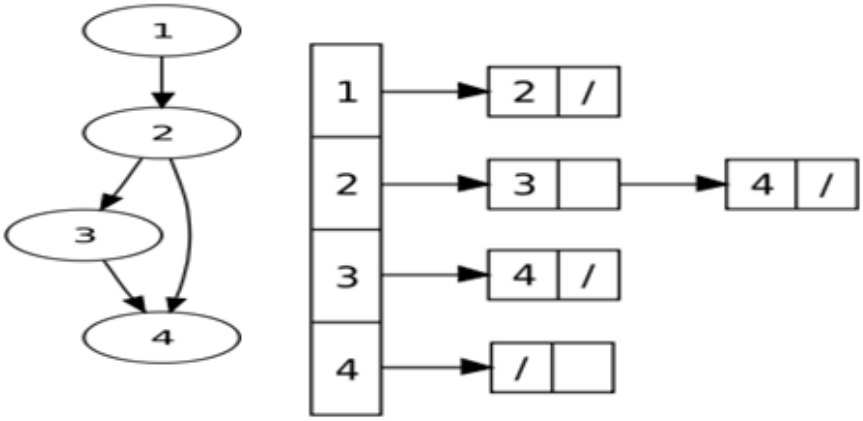


FIGURE 1.11: Liste d'adjacence d'un graphe

Pour un graphe étiqueté, il faut mémoriser, dans les nœuds, de la liste la valeur du nœud. On peut donc avoir le graphe étiqueté précédent et sa liste d'adjacence associée comme suit : (cf. ref. [61])(cf. ref. [26])(cf. ref. [33])(cf. ref. [57]).

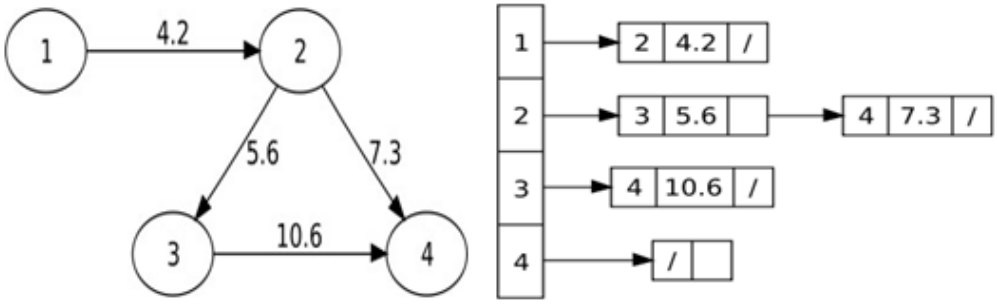


FIGURE 1.12: Liste d'adjacence d'un graphe étiqueté

1.2.2 Notions de base sur l'appariement de graphes :

La recherche d'appariements dans un graphe vise à trouver des couples de sommets compatibles dans des graphes. Un appariement est un ensemble d'arêtes indépendantes Ei dans un graphe G = (V,E). L'indépendance d'arêtes est définie par l'absence de sommets incidents communs entre elles. Il s'agit d'un appariement deV0 V si tous les sommets de V0 sont couverts par Ei, c'est-à-dire s'il existe une arête dans Ei qui relie chaque sommet dans Vi à un autre (cf. ref. [12]).

1.2.2.1 Isomorphisme et homomorphisme :

L'isomorphisme de graphes est une relation univoque des nœuds entre deux graphes qui respecte leurs attributs et ceux des arêtes. Dans le cas d'isomorphisme de sous-graphes, on considère deux graphes G1 et G2 de taille différente, |G2|= n2 |G1|=n1. Afin de déterminer si G2 est un sous-graphe de G1, une telle relation est recherchée entre les nœuds du graphe G1 et tous les sous-graphes du graphe G2 induits par n1 nœuds. De manière générale, on essaie de déterminer si deux graphes donnés sont égaux. Généralement, les notions d'homomorphisme et d'isomorphisme sont définies comme suit :

1.2.2.1.1 Homomorphisme de graphes : Soient G1=(V1,E1) et G2=(V2,E2) deux graphes. (dans ce document, nous désignons par G1 un graphe de données et G2 un graphe modèle ou graphe requête). S'il existe une relation univoque m : V1 V2 telle que (v1,v2) E (m(v1),m(v2)) E1, alors G1 et G2 sont homomorphes.[7].

1.2.2.1.2 Isomorphisme de graphes : :Soient G1 = (V1,E1) et G2 = (V2,E2) deux graphes. S'il existe une relation univoque m : V1 V2 telle que (v1,v2) E (m(v1), m(v2)) E1, alors G1 et G2 sont isomorphes. Par conséquent, un homomorphisme est un isomorphisme lorsque la relation m est bijective. En effet, un homomorphisme de G1 vers G2 permet qu'il y ait une arête entre deux sommets dans G2 faisant partie de la relation, sans qu'il y en ait entre leurs antécédents dans G1. Dans le cas de l'isomorphisme, une telle situation est impossible. (cf. ref. [12]).

1.2.2.1.3 Isomorphisme de sous-graphes : Le problème d'isomorphisme de sous-graphes consiste à trouver une application partielle entre les sommets de deux graphes, telle que les conditions définies précédemment soient satisfaites. Soient G1 et G2 deux graphes avec n2 n1, on cherche un sous-graphe de G1 induit par n1 sommets. Si l'on considérait des sous-graphes en général au lieu de sous-graphes induits par des sommets, on ne traiterait pas le problème d'isomorphisme mais celui d'homomorphisme.(cf. ref. [12]).

1.2.2.2 Complexité d'appariement de graphes :

L'appariement de graphes est considéré comme l'un des problèmes les plus complexes. Cette complexité est due à sa nature combinatoire. Les problèmes d'appariement exact de graphes ont été jusqu'à présent classifiés comme étant dans P ou NP complet. Quelques articles dans la littérature essayent de prouver qu'elle est NP complet quand

les deux graphes à apparier sont de types particuliers ou possèdent quelques contraintes spécifiques. D'autre part, pour quelques types de graphes, ils ont démontré que la complexité du problème d'isomorphisme est polynomiale, par exemple pour l'isomorphisme des graphes planaires(cf. ref. [27]). Cependant, certains types de graphes spécifiques peuvent avoir une complexité moins importante. Par exemple, le cas particulier, où le grand graphe est une forêt et le petit est un arbre. Finalement, dans l'appariement inexact de graphes, la complexité a été prouvée comme étant NP – complet.(cf. ref. [49]).

1.2.3 Notions de similarité entre graphes :

Il n'existe pas une règle pour calculer la similarité entre les graphes. Cela dépend généralement du type de l'application. Nous présentons, ci-après, les mesures les plus utilisées pour déterminer les similarités entre graphes.

1.2.3.1 La distance d'édition de graphes :

La distance d'édition de graphes proposée par Bunke (cf. ref. [10]). recherche l'ensemble d'opérations d'édition (insertion, suppression ou substitution des nœuds et des arêtes) nécessaires pour transformer, avec un coût minimal, un graphe en un autre graphe. Si les graphes ne sont pas étiquetés, seules les opérations de suppression et d'insertion sont utilisées. Chaque opération est associée à une fonction de coût qui varie selon la quantité de distorsion introduite par la transformation. L'ensemble d'opérations le moins couteux permettant de transformer

G1 en un graphe isomorphe à G2 définit la distance d'édition entre G1 et G2. La figure 1.13 représente un exemple illustrant l'ensemble d'opérations d'éditions qui permettent de transformer le graphe G1 en G2.

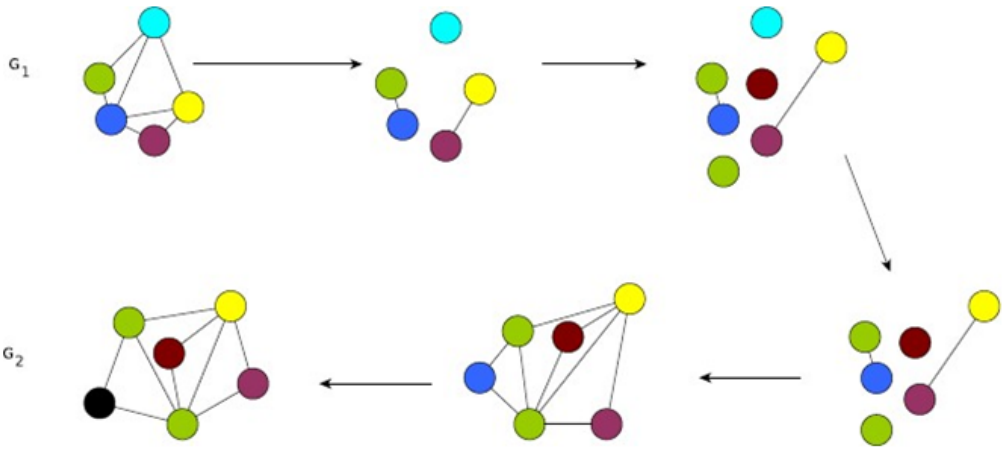


FIGURE 1.13: Un exemple d'une séquence d'opérations d'édition pour transformer G1 à G2

1.2.3.2 La distance basée sur la notion de sous-graphe commun maximal (SCM) :

L'avantage principal de cette approche basée sur le SCM est la non utilisation des fonctions de coût, palliant ainsi l'inconvénient principal de distance basée sur les séquences d'opérations d'édition. Il a été montré par Bunke et Shearer(cf. ref. [11])qu'une telle mesure est une métrique et conduit à une distance ayant des valeurs dans l'intervalle [0,1]. suit : Soit deux graphes G1 et G2, la similarité basée sur le SCM est définie comme sming

$$SimSCM(G1, G2) = \frac{|SCM(G1, G2)|}{|\max(G1, G2)|}$$

Où |max(G1,G2)|= max(|G1|,|G2|) et |SCM(G1,G2)| est le nombre d'arête dans SCM(G1,G2) (cf. ref. [37]).. La figure 1.14 montre un exemple d'un sous graphe maximum commun entre deux graphes G1 et G2 colorié en rouge,

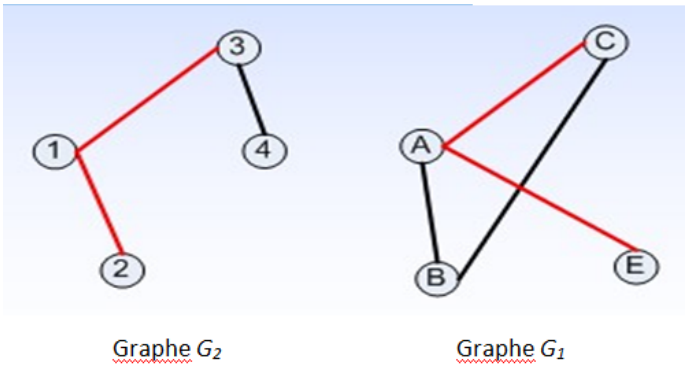


FIGURE 1.14: Exemple de sous graphe maximum commun entre deux graphes.

1.2.3.3 Autres distances pour calculer la similarité entre graphes :

Parmi les autres distances qui existent pour calculer la similarité entre graphes, on peut citer la distance de Wallis et all (cf. ref. [63]). qui est une mesure basée sur l'union de graphes (UG). Cette distance est utilisée pour modéliser la taille du problème. Elle est définit comme suit :

$$SimUG(G1, G2) = \frac{|SCM(G1, G2)|}{|G1| + |G2| - SCM(G1, G2)|}$$

Une autre distance présentée par Fernandez et al.(cf. ref. [**?**]). qui ont proposé de combiner le sous-graphe commun maximal et le super-graphe minimal commun pour calculer la distance entre deux graphes. Selon cette distance, pour que deux graphes G1 et G2 soient similaires, il faut que le sous-graphe commun maximal et le super-graphe minimal commun de G1 et G2 soient similaires (cf. ref. [37]).

1.3 Frequent sub-graphe mining (fsm) :

1.3.1 Le problème de découverte des sous-graphes connexes fréquent :

Le problème de découverte des sous-graphes connexes fréquents peut être défini de deux différentes façons : 1. Soit un ensemble de graphes DG, le but est de récupérer tous les sous-graphes connexes qui sont présents au moins sur graphes Gi DG. 2. Soit un seul grand graphe G, le but est de récupérer tous les sous-graphes qui oc- current au moins fois dans ce grand graphe.

De façon générale, pour les deux cas un algorithme de découverte des sous-graphes connexes fréquents doit faire face principalement à l'isomorphisme de sous graphes pour tester la fréquence, ainsi que l'isomorphisme de graphes afin d'éviter la génération de duplicatas. Encore, cet algorithme doit aussi faire attention à l'optimisation de nombre d'augmentations (la jonction pour les algorithmes à l'apriori (c.-à-d. ajouter une arête ou un sommet à un sous-graphe fréquent pour créer un nouveau sous-graphe connexe candidat). Pour le cas d'un seul grand

graphe, il faut être très attentif par rapport à la propriété de fermeture descendante du support, parce qu'en cas de présence de chevauchement entre des plongements différents (sous graphes différents qui partagent au moins une même arête ou un même sommet) cette propriété sera violée. Il existe beaucoup de méthodes de découverte des sous-graphes fréquents qui peuvent être catégorisées selon : 1) la stratégie d'exploration de l'espace de recherche. 2) la façon dont ces méthodes gèrent les problèmes d'isomorphisme de graphes et de sous-graphes.

Pour le cas d'un seul grand graphe, comme nous l'avons évoqué précédemment, les algorithmes de découvertes sont concernés par une tâche supplémentaire pour préserver la fermeture descendante du support. Nous citons deux algorithmes très connus : HSIGRAM et VSIGRAM . Ces deux algorithmes définissent le support comme la taille de l'ensemble d'indépendances maximales des sous-graphes chevauchés, ce qui permet de préserver la propriété de fermeture descendante. Ces deux algorithmes par- courent l'espace de recherche de deux manières différentes : soit en largeur pour le premier et en profondeur pour le deuxième. Il existe d'autres algorithmes pour la découverte des sous-graphes connexes au sien d'un seul grand graphe . Par ailleurs, le cas de découverte des sous-graphes connexes fréquents par rapport à une base de graphes est plus populaire . Ces algorithmes explorent l'espace de recherche de différentes manières en largeur et en profondeur, ils utilisent des techniques exactes ainsi que d'autres approchées, soit pour la restriction de l'espace de recherche, et/ou même pour résoudre le problème d'isomorphisme de sous- graphes, et suivent aussi différents

schémas de canonisation pour tester l'isomorphisme de graphes afin d'éviter la génération de candidats redondants. Dans la suite de ce manuscrit, nous allons considérer que le cas de découverte des sous-graphes connexes fréquents sur une base de graphes.(cf. ref. [39]).

1.3.2 Les problèmes de découverte des sous-graphes fréquents :

Comme tout algorithme de découverte des motifs fréquents, les principales tâches de la découverte des sous-graphes connexes fréquents sont la génération des candidats, et le calcul de la fréquence. Malheureusement, ces tâches sont beaucoup plus difficiles en raison de la complexité structurelle des graphes. La résolution de ces deux tâches consiste en la résolution d'isomorphisme de graphes et de sous-graphes pour chacun des candidats explorés, sachant qu'ils sont deux problèmes difficiles. La tâche de génération d'un sous- graphe plus spécifique à partir d'un plus général nécessite un nombre important de tests d'augmentations en raison du grand nombre de possibilités que cela soit de relier par une nouvelle arête deux sommets déjà existants, ou d'ajouter un nouveau sommet et le relier avec l'un des sommets déjà existant.(cf. ref. [39]).

1.3.3 La génération des sous-graphes connexes candidats :

La génération d'un nouveau sous-graphe connexe à k-sommets (k-arêtes) consiste géné- ralement à le produire :

*à partir d'un autre à k-1-sommets (k-1-arêtes) en lui rajoutant un sommet (une arête), ou un sous-graphe à k+1-sommets (k+1-arêtes) en éliminant un sommet (une arête).

*à partir d'une jonction de deux sous-graphes à k-1-sommets (k-1-arêtes) qui partagent un sous-graphe connexe en commun à k-2-sommets (k-2-arêtes) par l'union de leurs sommets et arêtes.

*Même à partir de l'intersection d'un en- semble de graphes Gi D qui contiennent un certain sous-graphe générateur inductif g. Cette opération d'intersection produit le sous-graphe maximal en commun qui peut être non connexe, alors l'ensemble de ces sous-graphes connexes maximaux est récupéré. Cette dernière technique de génération est beaucoup plus coûteuse, mais elle permet de réduire considérablement le nombre des candidats à explorer.(cf. ref. [39]).

1.3.3.1 L'augmentation :

Cette opération d'augmentation consiste à ajouter une arête ou un sommet à un sous- graphe à k-1-arêtes(k-1-sommets) connexe qui a déjà été exploré afin de construire nouveau sous-graphe connexe à k-arêtes (k-sommets). Contrairement au cas des itemsets où la tâche d'augmentation se résume à ajouter un nouvel item à un ensemble d'items fréquent déjà exploré, pour les graphes connexes, cette opération d'ajouter soit un sommet ou une arête est beaucoup plus compliquée. La difficulté de cette opération à l'égard des graphes est due au nombre important de possibilités de jonctions. Afin d'ajouter un sommet, tous les sommets du graphe sont des cibles potentielles pour

les lier au nouveau sommet. De plus, il en est de même pour le choix du sommet à relier au nouveau sommet. Une étiquette d'arête parmi toutes les étiquettes possibles doit être associée à ce nouveau lien. Pour ajouter une nouvelle arête, l'opération d'augmentation consiste à chercher deux sommets qui partagent les mêmes étiquettes des sommets de l'arête à ajouter et les lier. S'il n'existe qu'un seul des deux sommets alors un nouveau sommet est ajouté avec l'arête au sous-graphe.(cf. ref. [39]).

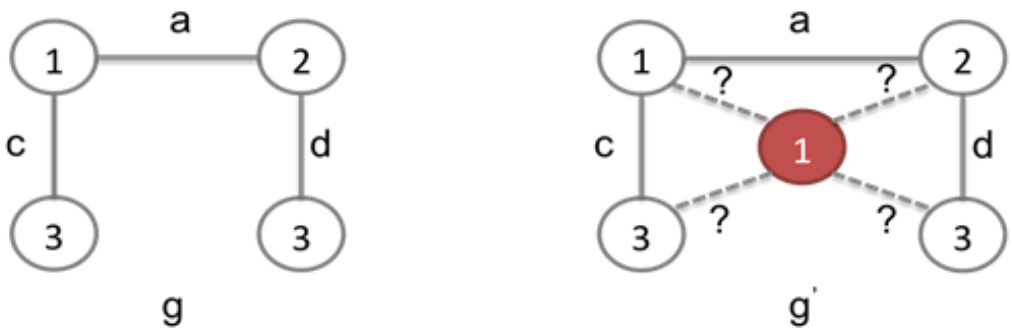


FIGURE 1.15: Exemple d'augmentation d'un sous-graphe par l'ajout d'un sommet.

Pour l'ajout d'un nouveau sommet, il est possible de le relier à chacun des sommets du sous-graphe. De même pour le cas de l'ajout d'une arête où il peut y avoir un nombre important de sommets ayant le même étiquetage que celui de l'arête à ajouter. La figure 3.1 illustre un exemple d'augmentation d'un sous-graphe g par l'ajout d'un sommet, le graphe gJ produit où ce nouveau sommet (rouge) avec l'étiquette 1 peut être relié à cha- cun des sommets du sous-graphe ce qui permet de générer jusqu'à (24 1) sous-graphes connexes différents.Alors que la figure 1.15 illustre un exemple par rapport à l'ajout d'une arête e au

sous-graphe g. L'opération d'augmentation dans ce cas est réalisée soit en reliant deux sommets déjà existants soit en ajoutant un nouveau sommet. Par conséquent, comme l'illustre le graphe gJ il est possible d'avoir 6 sous-graphes connexes différents à l'issue de cette augmentation.

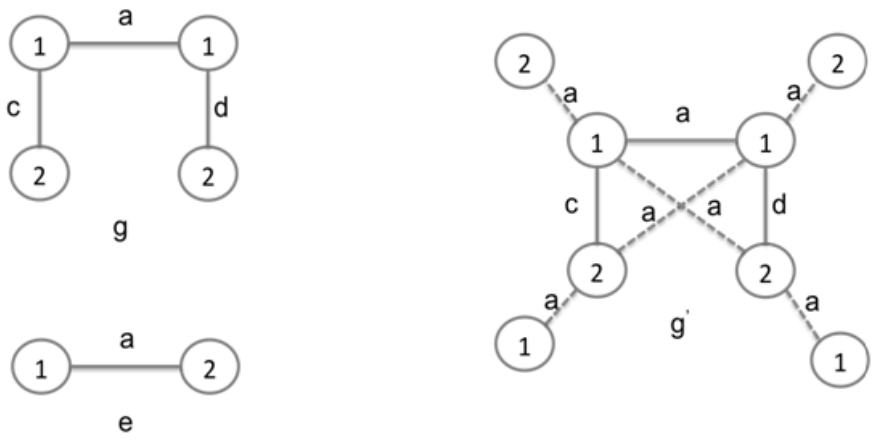


FIGURE 1.16: Exemple d'augmentation d'un sous-graphe par l'ajout d'une arête.

1.3.3.2 La jonction :

Utilisée la première fois par , la jonction est l'une des principales opérations caractéristiques de l'exploration à l'apriori. Cette opération de jonction consiste en l'union des composants de deux motifs d'un même niveau de spécification (c.-à-d. car- dinal, taille, nombre de chemins disjoints, etc.) ayant un seul élément de différence (c.-à-d. item, sommet, arête, etc.) qui ont été déjà explorés et classés fréquents. Pour le cas des itemsets, la jonction est l'union des items de deux sous-ensembles de k-1-items à la différence d'un item, afin de générer un nouvel ensemble d'items à k-items. Alors que cette opération semble

être simple pour le cas des itemsets, due à la notion d'identité, la jonction entre un couple de sous-graphes connexes fréquents à k-1-sommets (k-1-arêtes) est beaucoup plus complexe.

La jonction de deux sous-graphes à k-1-sommets (k-1-arêtes) nécessite qu'ils partagent un sous-graphe connexe de k-2-sommets (k-2-arêtes) ce qui est difficile à vérifier (NP complet) sans avoir recours à de nombreux tests d'isomorphisme de sous-graphes. À partir du sous-graphe en commun, les deux sommets (arêtes) différents sont ajoutés, cependant est-ce que les deux sommets ajoutés seront reliés ou non ? S'ils sont considérés reliés, qu'elle est l'étiquette suggérée à cette arête ? Pour deux arêtes ajoutées ayant deux sommets avec la même étiquette, est-ce que les deux sommets seront fusionnés pour un seul sommet ou les considérer comme deux sommets différents ? Pour le premier cas, soit deux sous-graphes connexes g1 et g2, pour ETE(g1) et ETE(g2) des ensembles contenant les étiquettes des arêtes de g1 et g2, |ETE(g1)) ETE(g2)j + 1 est le nombre de valeurs possibles pour l'étiquette de l'arête entre les deux sommets différents (la valeur supplémentaire est pour le cas où les deux sommets ne sont pas reliés.

1.3.3.3 La vérification de fréquence :

Malgré la complexité de la phase de génération des motifs, la phase de vérification (c.-à-d. de calcul, et de validation du prédicat) de fréquence reste la phase clé pour un processus de découverte de motifs fréquents. Alors, pour un processus de découverte des sous-graphes connexes fréquents, le calcule de la fréquence consiste à tester l'exis-

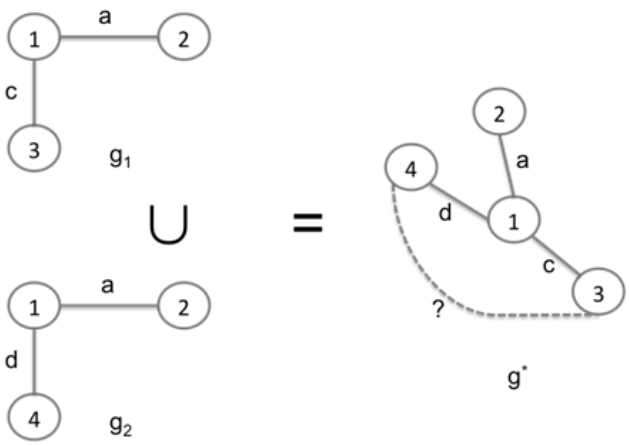


FIGURE 1.17: Exemple de jonction de deux graphes connexes à base de sommets.

tence d'un isomorphisme de sous-graphes de chacun des candidats qui ont été générés par rapport à chacun des graphes connexes de la base de graphes Gi D.(cf. ref. [39]).

1.3.3.4 Le problème d'isomorphisme de sous-graphes :

Encore plus difficile que l'isomorphisme de graphes, l'isomorphisme de sous-graphes est connu comme étant un problème de complexité NP complet . Cette difficulté est due à la complexité de la structure manipulée, ce qui nécessite l'explo- ration des deux graphes pour vérifier les différentes substitutions des sommets et des arêtes possibles. Pour tester si un graphe G1 de taille t1 est un sous-graphe d'un graphe G2 de taille t2 > t1, il faut tester l'isomorphisme de graphes entre G1 et chacun des sous-graphes de G2 de taille t1. Si l'un des sous-graphes de G2 est isomorphe à G1 alors le test est validé.

La figure 1.17. illustre trois graphes X1, X2 et X3 où X1 est un sous-graphe isomorphe de X2 dû au fait qu'il est isomorphe à g1 l'un

des sous-graphes de X2 g1,g2 et g3. Alors que X1 n'est isomorphe à aucun des sous-graphes de X3 (c.-à-d. g4, g5, g6 et g7) ce qui signifier que X1 n'est pas un sous-graphe de X3. Autrement dit, le graphe G2 contient le graphe G1, noté G1 G2, s'il existe un isomorphisme de G1 à un sous-graphe de G2 (c.-à-d. une bijection entre G2 et l'un des sous-graphes de G1).(cf. ref. [39]).

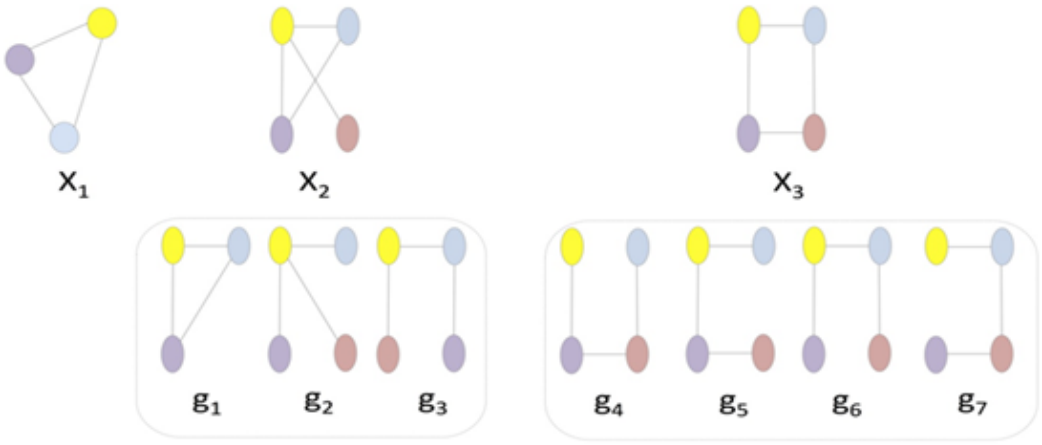


FIGURE 1.18: Exemple d'isomorphisme de sous-graphes.

1.3.4 Les approches des algorithmes FSM :

Cette section fournit un aperçu générique du processus de FSM, il est largement admis que les techniques de FSM peuvent être divisées en deux catégories : (i) Approches basées sur les Apriori. (ii) Modèles basés sur la croissance de modèle. Ces deux catégories sont similaires dans leur esprit à leurs homologues trouvés dans Association Rule Mining (ARM), à savoir l'algorithme Apriori (cf. ref. [3]). et l'algorithme de croissance FP (cf. ref. [34]). respectivement. Avant de définir chacun d'eux, examinons d'abord la propriété apriori.

1.3.4.1 Propriété Apriori :

La propriété Apriori, également connue sous le nom de propriété de fermeture vers le bas (DCP), exprime une diminution monotone d'un critère d'évaluation accompagnant l'avancement d'une exploration séquentielle. Il est activé afin de découvrir efficacement tous les modèles séquentiels fréquents. En termes simples, cette propriété impose que si un graphe est fréquent, alors tous ses sous-graphes seront également fréquents, donc la fréquence ou le support d'un graphe séquentiel diminue toujours ou restant constant, et n'augmente jamais pour dépasser le support de ses sous-graphes parents.(cf. ref. [7]).

1.3.4.2 Approche basée sur Apriori :

L'approche basée sur Apriori se déroule de manière à générer et tester en utilisant une stratégie de recherche en largeur (BFS) pour explorer le réseau sous-graphe de la base de données. Par conséquent, avant de considérer (k + 1) les sous-graphes, cette approche doit d'abord considérer tous les k sous-graphes. (cf. ref. [29]). Le cadre général des algorithmes basés sur Apriori est décrit dans l'algorithme AprioriGraph ci-dessous : Soit sk ensemble de sous-graphes fréquent de taille k l'algorithme AprioriGraph adopte une méthodologie d'exploration de niveaux. À chaque itération, la taille des sous-graphes fréquents récemment découverts est augmentée d'une unité. Ces nouveaux sous-graphes sont d'abord générés en joignant deux sous-graphes fréquents similaires découverts lors du dernier appel de l'algorithme. Les graphes nouvellement formés sont ensuite vérifiés pour leur fréquence. Les plus

```
    procedure AprioriGraph(D, min_sup, S_k)
1       S_{k+1} ← ∅;
2       foreach frequent g_i ∈ S_k do
3           foreach frequent g_j ∈ S_k do
4               foreach size (k+1) graph g formed by merge(g_i, g_j) do
5                   if g is frequent in D and g ∉ S_{k+1} then
6                       insert g into S_{k+1};
7       if S_{k+1} ≠ ∅ then
8           AprioriGraph(D, min_sup, S_{k+1});
9       return;
```

FIGURE 1.19: Algorithme apriori

fréquents sont utilisés pour générer de plus gros candidats au prochain tour. La principale complexité de conception des algorithmes basés sur Apriori provient de l'étape de génération du candidat. Bien que la génération candidate pour l'exploration fréquente d'éléments soit simple, le même problème dans le contexte de l'exploration de graphes est beaucoup plus difficile, car il existe de nombreuses façons de fusionner deux graphes(cf. ref. [7]).

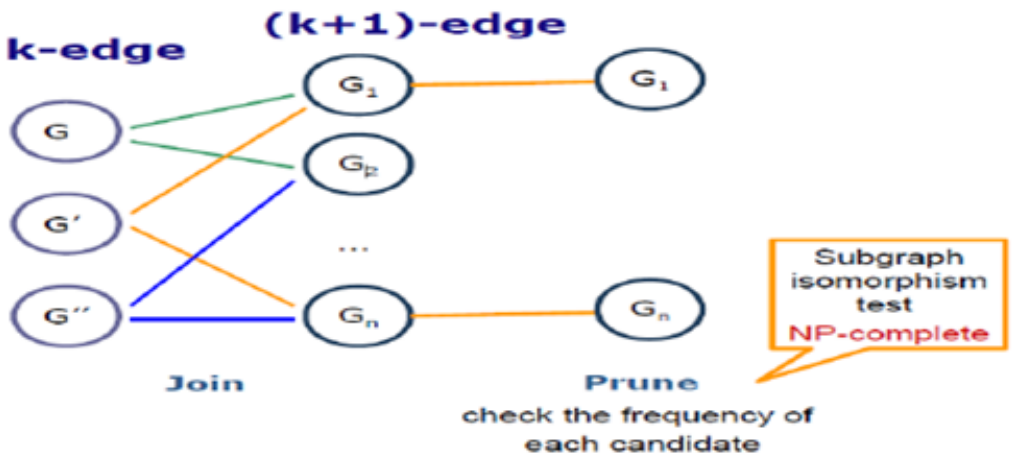


FIGURE 1.20: Approche basé sur apriori

1.3.4.3 Approche PatternGrowthGraph :

Le modèle basé sur PatternGrowthGraph adopte des stratégies DFS ou BFS pour explorer le réseau de sous-graphe, et pour chaque sous-graphe découvert g le sous-graphe est étendu de manière récursive jusqu'à ce que tous les sous graphes fréquents de g soient découverts. Algorithm PatternGrowthGraph illustre un cadre d'algorithmes d'exploration de graphes fréquents basés sur la croissance de motifs. Un graphe g eut être étendu en ajoutant un nouveau bord e Le graphe nouvellement formé est noté g+*xe . L'arête e peut ou non introduire un nouveau sommet dans g Pour chaque graphe découvert g PatternGrowth effectue des extensions récursivement jusqu'à ce que tous les graphes fréquents avec g incorporé soient découverts. La récursivité s'arrête une fois qu'aucun graphe fréquent ne peut plus être généré.

```
    procedure PatternGrowthGraph(g, D, min_sup, S)
1       if g ∈ S then return;
2       else insert g into S;
3       scan D once, find all edges e that g can be extended to g◊ₓe;
4       foreach frequent g◊ₓe do
5           PatternGrowthGraph(g◊ₓe, D, min_sup, S);
6       return;
```

FIGURE 1.21: Algorithme de croissance de modèle

L'algorithme ci-dessus est simple, mais pas efficace. Le goulot d'étranglement est à l'inefficacité de l'extension d'un graphe. Le même graphe peut être découvert plusieurs fois. Par exemple, il peut exister n différents (n- 1) graphes d'arêtes qui peuvent être étendus au même graphe

à n arêtes. La découverte répétée du même graphe est inefficace sur le plan des calculs. Un graphe qui est découvert à la deuxième fois est appelé graphe en double. Bien que la ligne 1 de l'algorithme supprime les graphes en double, la génération et la détection de graphes en double peuvent entraîner des charges de travail supplémentaires. Afin de réduire la génération de graphes en double, chaque graphe fréquent doit être étendu de la manière la plus conservatrice possible, en ce sens que toutes les parties de chaque graphe ne doivent pas être étendues. (cf. ref. [7]).

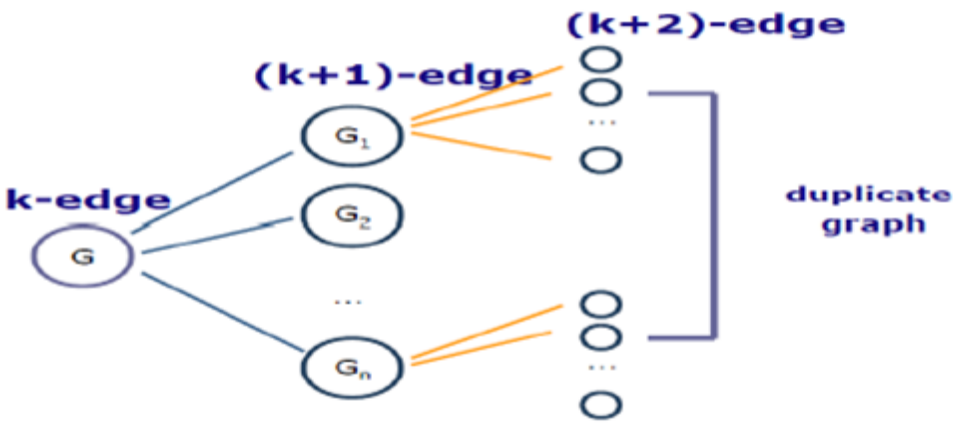


FIGURE 1.22: Approche basé sur la croissance de modèle

1.3.5 Frequent subgraph mining algorithms :

Il existe de nombreux algorithmes pour résoudre la version en mémoire des tâches fréquentes d'extraction de sous-graphes, les plus notables d'entre eux étant AGM (cf. ref. [36])., FSG (cf. ref. [27])., gSpan (cf. ref. [1])., Gaston (cf. ref. [52]). et DMTL (cf. ref. [16]). Ces méthodes supposent que l'ensemble de données est petit et que la tâche

d'exploration se termine dans un laps de temps raisonnable à l'aide d'une méthode en mémoire. Pour prendre en compte le scénario de données volumineuses, quelques algorithmes traditionnels d'extraction de graphes basés sur des bases de données, tels que DB-Subdue (cf. ref. [14]), DB-FSG (cf. ref. [15]). et OO-FSG (cf. ref. [60]). sont également proposés. Il existe également quelques travaux qui explorent les sous-graphiques qui sont fréquents compte tenu des occurrences induites de ces sous-graphiques dans un seul grand graphique (cf. ref. [66])..,(cf. ref. [48]).. Wu et coll. développé un algorithme d'extraction de sous-graphes distribués(cf. ref. [66]). qui nécessite le diamètre du graphe et le nombre de sommets pour l'appariement de motifs. Liu et coll. a proposé un algorithme MRPF (cf. ref. [48]). qui trouve des motifs dans le réseau de compatibilité de prescription. Ces deux derniers travaux considèrent l'extraction à partir d'un seul grand graphe, mais FSM-H extrait un grand ensemble de graphes de taille moyenne pouvant tenir dans une mémoire. Pour les tâches d'extraction de graphes à grande échelle, les chercheurs ont envisagé des algorithmes parallèles à mémoire partagée pour l'exploitation fréquente de sous-graphes. Cook et coll. ont présenté une version parallèle de leur algorithme d'extraction de sous-graphes Subdue (cf. ref. [17]). Wang et coll. développé une boîte à outils parallèle(cf. ref. [64]). pour leur algorithme MotifMiner (cf. ref. [53]). Meinl et coll. a créé un logiciel nommé Parmol (cf. ref. [50]). qui inclut l'implémentation parallèle de Mofa, gSpan, FFSG et Gaston. ParSeMis (cf. ref. [54]). est un autre outil de ce type qui fournit une implémentation parallèle de l'algorithme gSpan. Pour faire face

au problème d'évolutivité causé par la taille des graphiques d'entrée, il existe quelques travaux notables, PartMiner (cf. ref. [65]). et Part-GraphMining (cf. ref. [51])., qui sont basés sur l'idée de partitionner les données du graphique.(cf. ref. [5]). En plus d'être classés comme des algorithmes parallèles en mémoire, basés sur une base de données et à mémoire partagée, les algorithmes FSM pourraient être classés en tenant compte de différentes considérations.

1.3.6 FSM inexact :

Les algorithmes FSM basés sur une recherche inexacte utilisent une mesure approximative pour comparer la similitude de deux graphes, c'est-à-dire que deux sous-graphes ne doivent pas être entièrement identiques pour contribuer au nombre de supports, mais qu'un sous-graphe peut contribuer au nombre de supports pour un sous-graphe candidat s'il est en quelque sorte similaire au candidat. La recherche inexacte n'est bien sûr pas garantie de trouver tous les sous-graphes fréquents, mais la nature de la comparaison approximative des graphes conduit souvent à des gains d'efficacité de calcul. Exemple SUBDUE(cf. ref. [35]). GREW (cf. ref. [6]).

1.3.7 FSM exact :

Les algorithmes FSM exacts sont beaucoup plus courants que les algorithmes FSM basés sur une recherche inexacte. Ils peuvent être appliqués dans le contexte de l'extraction basée sur des transactions de graphes ou de l'exploration basée sur un seul graphique. Une carac-

téristique fondamentale des algorithmes basés sur la recherche exacte est que l'extraction est terminée, c'est-à-dire que les algorithmes d'exploration sont garantis pour trouver tous les sous-graphiques fréquents dans les données d'entrée. Cependant, des algorithmes d'exploration de données complets ne fonctionnent efficacement que sur des graphiques clairsemés avec une grande quantité d'étiquettes pour les sommets et les arêtes. En raison de cette restriction d'exhaustivité, ces algorithmes effectuent une comparaison approfondie des isomorphismes de sous-graphes, soit explicitement soit implicitement, ce qui entraîne une surcharge de calcul importante. Exemples AGM(cf. ref. [36]). FSG(cf. ref. [41]). MoFa(cf. ref. [42]). gSpan(cf. ref. [1]). GASTON(cf. ref. [52]).

1.4 Conclusion :

Nous avons introduit le chapitre en donnant quelques notions préliminaires de graphes et de théorie des graphes. Nous nous sommes concentrés dans ce chapitre sur l'exploitation des algorithmes (FSM) exploration des sous graphe fréquent (FSM). Nous avons donné les principales opérations qu'un algorithme FSM doit effectuer pour trouver des sous-graphes, par exemple la génération de candidats, la vérification d'isomorphisme et le comptage de support et présenté les techniques associées pour réaliser chacun d'entre eux. Nous avons présenté les approches algorithmiques FSM, Apriori et Pattern Growth. Dans le chapitre suivant on va présenter l'intelligence artificielle, machine learning et le deep Learning .

Chapitre 2

L'IA,Machine Learning,Deep Learning

2.1 Introduction :

Ce chapitre se concentre sur les concepts de base de l'intelligence artificielle, machine Learning, et le Deep Learning il est organiser en trois partie Partie 01 : présente principalement les notions de base de l'intelligence artificielle tel que la définition de l'IA, les trois grandes domaines de l'IA, la segmentation de l'IA et le lien entre la qualité des solutions d'IA et celle des données qui les alimentent. Partie 02 : se concentre sur les défirents concepts de machine Learning tel que la définition da machine Learning, les défirent catégories : superviser, non superviser, par renforcement chacun avec ces défirent types, les outils de machine Learning. Partie 03 : se présente principalement les

défirent concept de Deep Learning : la définition de Deep Learning, son principe, l'évolution de celle-ci les défirent catégories de D eep Learning les outils de Deep Learning.

2.2 L'intelligence artificielle

2.2.1 Définition :

L'IA est un ensemble de techniques permettant de résoudre des problèmes complexes en s'inspirant de mécanismes cognitifs humains, agissant de manière rationnelle en fonction de faits, données et expériences, et capables d'atteindre de manière optimale un ou plusieurs objectifs donnés. La rationalité n'est pas l'omniscience mais la capacité à agir en fonction des informations disponibles, y compris celles qui sont ambigües. Cette rationalité est habituellement limitée par notre volonté, le poids émotionnel de notre cerveau limbique et notre capacité d'optimisation. (cf. ref. [28]).

2.2.2 Domaine de l'IA :

A haut niveau, on peut découper l'IA en trois grands domaines :

2.2.2.1 Le symbolisme :

Qui se focalise sur la pensée abstraite et la gestion des symboles, l'algorithmique et la logique. Le symbolisme modélise notamment les concepts sous la forme d'objets reliés entre eux par des prédicats logiques (appartient à, etc). C'est une approche « macro » de résolution

de problèmes. C'est dans cette catégorie que l'on peut ranger les systèmes experts et moteurs de règles qui les font fonctionner, et dans une certaine mesure, le web sémantique. (cf. ref. [28]).

2.2.2.2 Le connexionnisme :

Qui se focalise sur la perception, dont la vision, la reconnaissance des formes et s'appuie notamment sur les réseaux neuronaux artificiels qui reproduisent à petite échelle et de manière approximative le fonctionnement générique du cerveau. C'est une vision « micro » et probabiliste de la résolution des problèmes. C'est ici que l'on peut ranger le Deep Learning utilisé dans la vision artificielle ou le traitement de la parole. Cette IA est aussi associable aux méthodes stochastiques et heuristiques du machine Learning.(cf. ref. [28]).

2.2.2.3 Le comportementalisme :

Qui s'intéresse aux pensées subjectives de la perception. C'est dans ce dernier domaine que l'on peut intégrer l'informatique affective (ou affective computing) qui étudie les moyens de reconnaître, exprimer, synthétiser et modéliser les émotions humaines. C'est une capacité qu'IBM Watson est censé apporter au robot Pepper de Softbank Robotics (ex Aldebaran). (cf. ref. [28]).

2.2.3 Segmentation :

Cette segmentation relie entre eux quatre domaines de manière plus hiérarchique

2.2.3.1 Les solutions :

Que l'on va directement utiliser dans les entreprises ou chez les particuliers avec les chatbots, les véhicules autonomes, les robots, les systèmes de recommandation, les outils de segmentation client, le marketing prédictif ou les solutions de cybersécurité.(cf. ref. [28]).

2.2.3.2 Les outils :

Qui aident à créer ces solutions, comme la vision artificielle, la reconnaissance de la parole, la traduction automatique, les systèmes experts, les outils de prévision ou de segmentation automatiques. on'y a ajouté les réseaux multi-agents qui coordonnent l'action des différents outils d'une solution d'IA. (cf. ref. [28]).

2.2.3.3 Les techniques :

Sur lesquelles sont construits ces outils, avec les méthodes de machine lear-ning, les réseaux de neurones, les nombreuses méthodes de deep learning et les moteurs de règles. (cf. ref. [28]).

2.2.3.4 Les données :

Les sources de données correspondantes et les capteurs associés qui jouent un rôle indispensable, notamment pour les approches connexionnistes, le machine Learning et le Deep learning.(cf. ref. [28]).

Cela rappelle que les solutions à base d'IA sont des assemblages de diverses briques logicielles et matérielles selon les besoins. Ces briques

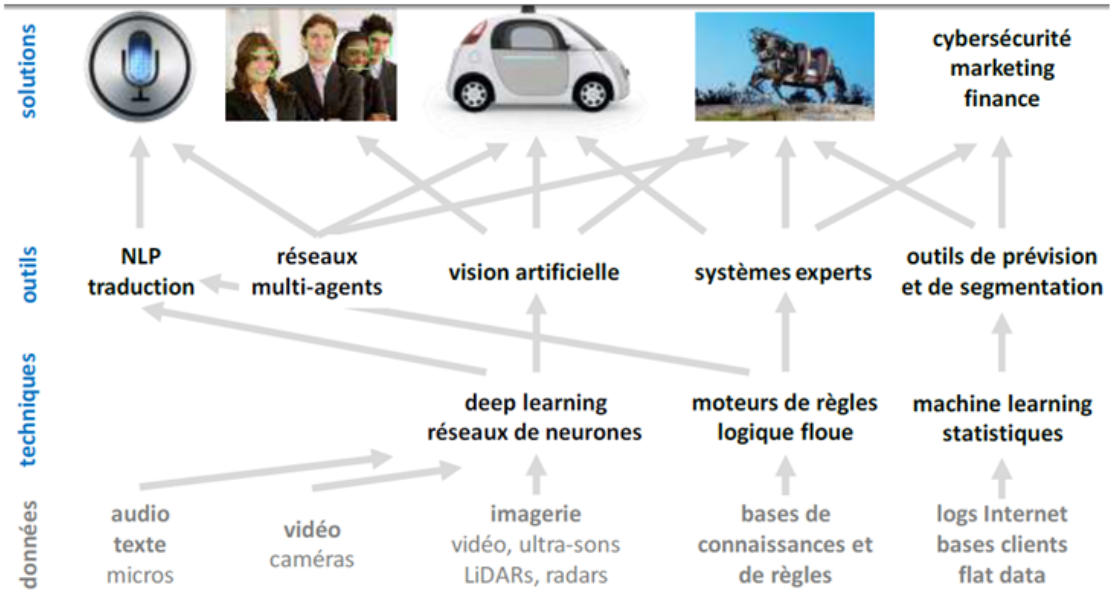


FIGURE 2.1: les données de l'IA

sont des plus nombreuses. A tel point que leur intégration est un en-jeu technique et métier de taille, peut-être le plus complexe à relever. Quand une startup indique qu'elle a créé « une IA » pour faire ceci ou cela, cela signifie qu'elle a assemblé des techniques, paramétré des outils, en général assez standards, pour exploiter des don-nées, et les a appliqués pour créer une solution. L'originalité est rarement dans la création des briques mais plutôt dans leur sélection, leur combinaison, leur assemblage et le problème métier traité !

2.2.4 Données de l'IA :

Le point commun du machine Learning et du Deep Learning est d'exploiter des données pour l'entraînement de modèles probabilistes. Avec les algorithmes/logiciels et le matériel, les données sont la troi-sième composante clé de la plupart des IA du jour. Il faut donc se poser la question du lien entre la qualité des solutions d'IA et celle des

données qui les alimentent. (cf. ref. [28]).

2.2.4.1 Type de données :

On distingue généralement trois types de données pour entraîner un système de machine Learning et de Deep Learning : les données d'entraînement, les données de test et les données de production. [(cf. ref. [28]).

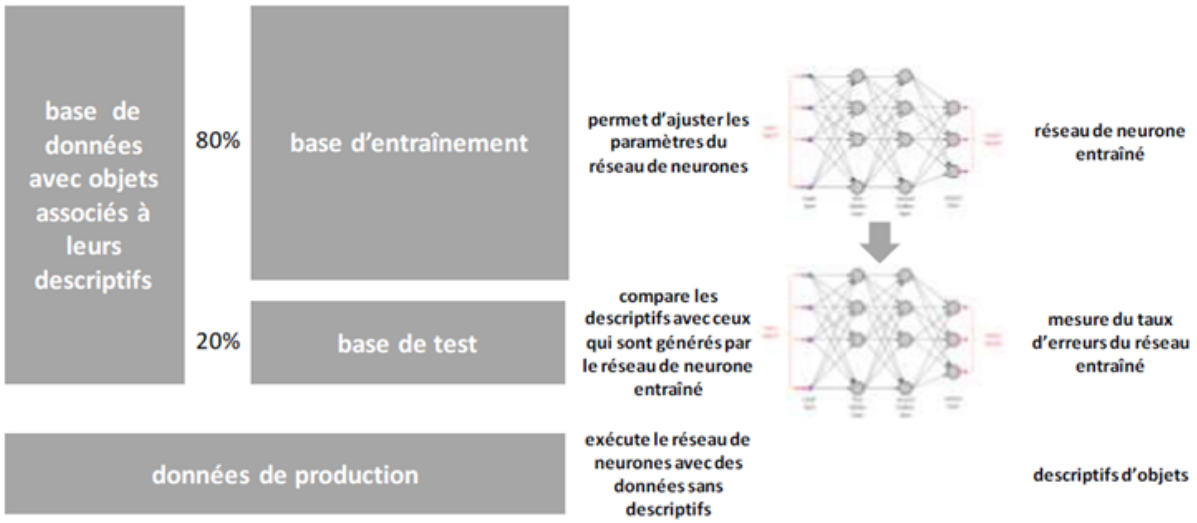


FIGURE 2.2: les types de données

Dans le machine et le deep learning supervisé, les données d'entraînement et de tests contiennent leur label, à savoir, l'information qui doit être générée par le système à entraîner. C'est un jeu de test doté d'une bonne répartition de l'espace du possible de l'application. On le découpe arbitrairement en deux sous-ensembles, l'un pour l'entraînement et l'autre pour les tests de qualification du réseau de neurones entraîné qui déterminent un taux d'erreur de reconnaissance. En général, la part de la base taggée dédiée à l'entraînement est plus grande

que celle qui est dédiée aux tests et dans un rapport 3/4 et 1/4. Les données d'entraînement et de tests sont indispensables pour la grande majorité des systèmes d'IA à base de machine learning, que ce soit pour de l'apprentissage supervisé ou non supervisé. L'apprentissage par renforcement utilise une plus faible quantité de données mais s'appuie en général sur des modèles déjà entraînés au préalable. On pourrait y ajouter les données de renforcement qui servent aux apprentissages par renforcement. On peut considérer qu'il s'agit de nouveaux jeux de données d'entraînement qui permettent d'ajuster celui d'un réseau de neurones déjà entraîné.

2.2.4.1.1 Données d'entraînement : Ce sont les jeux de données qui vont servir à entraîner un modèle de machine Learning ou de Deep Learning pour en ajuster les paramètres. Dans le cas de la reconnaissance d'images, il s'agira d'une base d'images avec leurs tags correspondants qui décrivent leur contenu. Plus la base est grande, meilleur sera l'entraînement du système, mais plus il sera long. Si vous n'avez pas de données déjà taggées pour entraîner un modèle de machine Learning ou de Deep Learning, vous n'irez pas bien loin ! Les bases d'entraînement d'images ont une taille qui dépend de la diversité des objets à détecter. Dans l'imagerie médicale, des bases d'entraînement de pathologies spécialisées peuvent se contenter de quelques centaines de milliers d'images pour détecter quelques dizaines ou centaines de pathologies. A l'autre extrémité de la complexité, la base d'entraînement d'images de Google Search s'appuie sur plusieurs centaines de millions d'images et permet la détection de plus de 20 000 objets différents.

L'entraînement d'un système de 50 000 images dure environ un quart d'heures dans des ressources en Cloud mais cela dépend des ressources allouées de ce côté-là. Lorsque l'on passe à des centaines de millions d'images, il faudra des milliers de serveurs et jusqu'à plusieurs semaines pour l'entraînement ! Dans la pratique, les jeux d'entraînement de solutions de deep Learning sont limités en taille par la puissance de calcul nécessaire. On peut aussi réaliser un entraînement incrémental au gré de l'ajout de données avec des techniques de transfert de réseau de neurones. Il est évidemment nécessaire de disposer de données d'entraînement de qualité, ce qui nécessite souvent un gros travail de filtrage, de nettoyage et dédoublonnage préalable à l'ingestion des données, une tâche existant déjà dans le cadre d'applications de big data. (cf. ref. [28]).

2.2.4.1.2 Données de test : Ce sont les données, également taggées, qui serviront à vérifier la qualité de l'entraînement d'un système. Ces données doivent avoir une distribution statistique voisine des données d'entraînement, au sens où elles doivent être bien représentatives de la diversité des données que l'on trouve dans la base d'entraînement et que l'on aura dans les données de production. Les données de tests sont un sous-ensemble d'un jeu de départ dont une partie sert à l'entraînement et une autre partie, plus limitée, sert aux tests. Elles seront injectées dans le système entraîné et on en comparera les tags résultants avec les tags de la base. Cela permettra d'identifier le taux d'erreur du système. On passera à l'étape suivante lorsque le taux d'erreur sera considéré comme acceptable pour la mise en pro-

duction de la solution. Le niveau de taux d'erreur acceptable dépend de l'application. Son maximum généralement accepté est le taux d'erreur de la reconnaissance humaine. Mais comme on est généralement plus exigeant avec les machines, le taux véritablement accepté est très inférieur à celui de l'Homme. (cf. ref. [28]).

2.2.4.1.3 Données de production : Il s'agit des données non taggées qui alimenteront le système lors de son utilisation en production pour faire une prévision des tags manquants. Alors que les données d'entraînement sont normalement anonymisées pour l'entraînement du système, les données de production peuvent être nominatives ainsi que les prévisions associées générées par la solution. La nouvelle réglementation RGPD de l'Union Européenne exige que les entreprises conservent les données personnelles des utilisateurs ainsi que les données générées. Cela concerne donc à priori les données générées par les systèmes à base d'IA. Une donnée personnelle générée artificiellement reste une donnée personnelle ! Et son origine artificielle doit être connue et traçable en cas d'audit. (cf. ref. [28]).

2.2.4.1.4 Données de renforcement : Cette expression pour décrire les données qui servent à l'apprentissage par renforcement. Dans un chatbot, cela sera par exemple les données de réactivité des utilisateurs aux réponses des chatbots permettant d'identifier celles qui sont les plus appropriées. En quelque sorte, ce sont des résultats d'A/B testing réalisés sur les comportements d'agents à base d'IA. Tout ce qui pourra être capté sur la réaction du monde réel aux agissements d'un

agent à base d'IA permettra potentiellement d'en ajuster le comportement par réentraînement. L'apprentissage par renforcement est finalement une sorte d'apprentissage supervisé incrémental car on l'utilise pour faire évoluer par petites touches impressionnistes des systèmes déjà entraînés. (cf. ref. [28]).

2.2.4.2 Origine des données :

Les données alimentant les systèmes d'IA proviennent de l'intérieur et/ou de l'extérieur de l'entreprise. Elles sont issues de toutes sortes de capteurs divers : des objets connectés, du plus simple (thermomètre connecté) aux plus sophistiqués (machine outil, Smartphone, ordinateur personnel). Comme pour les applications de Big data habituelles, les sources de données doivent être fiables et les données bien extraites et préparées avant d'être injectées dans les systèmes à base de machine comme de Deep Learning. Les solutions les plus avancées exploitent conjointement des données ouvertes externes et les croisent aux données que l'entreprise est seule à maîtriser. C'est un bon moyen de créer des solutions différenciées. (cf. ref. [28]).

2.3 Machine Learning :

2.3.1 Définition :

Le vaste domaine du machine learning, ou apprentissage automatique, est défini comme le champ de l'IA qui utilise des méthodes probabilistes pour apprendre à partir des données sans être pro-grammé

explicitement. D'un point de vue pratique, le machine learning vise notamment à identifier des tendances, à faire des prévisions sur des données (régressions linéaires et non linéaires), à découvrir des corrélations entre données et événements (comme pour déterminer si un logiciel est un virus, si un client risque de quitter un service sur abonnement ou au contraire, s'il sera intéressé par telle ou telle offre ou qu'un tableau clinique d'un patient est symptomatique de l'émergence d'une pathologie de longue durée), à segmenter des jeux de données (comme une base clients), à reconnaître des objets (des lettres, des objets dans des images), le tout en exploitant des données d'entraînement. L'apprentissage automatique s'appuie sur des données existantes. Elles lui permettent de produire des prévisions, des segmentations ou des labels à partir de la généralisation d'observations. (cf. ref. [28]).

2.3.2 Catégories de machine Learning :

2.3.2.1 L'apprentissage supervisé :

Avec la classification qui permet de labelliser des objets comme des images et la régression qui permet de réaliser des prévisions sur des valeurs numériques. L'apprentissage est supervisé car il exploite des bases de données d'entraînement qui contiennent des labels ou des données contenant les réponses aux questions que l'on se pose. En gros, le système exploite des exemples et acquiert la capacité à les généraliser ensuite sur de nouvelles données de production.(cf. ref. [28]).

2.3.2.1.1 Classification Il s'agit de pouvoir associer une donnée complexe comme une image ou un profil d'utilisateur à une classe d'objet, les différentes classes possibles étant fourni a priori par le concepteur. La classification utilise un jeu de données d'entraînement associé à des descriptifs (les classes) pour la détermination d'un modèle. Cela génère un modèle qui permet de prédire la classe d'une nouvelle donnée fournie en entrée. Dans les exemples classiques, nous avons la reconnaissance d'un simple chiffre dans une image, l'appartenance d'un client à un segment de clients ou pouvant faire partie d'une typologie particulière de clients (mécontents, pouvant se désabonner à un service, etc) ou la détection d'un virus en fonction du comportement ou de caractéristiques d'un logiciel. Il existe plusieurs méthodes de classification dont voici les principales. * Les arbres de décision

* Les Support Vector Machines (SVM)(cf. ref. [55]). (cf. ref. [13]).

* les méthodes des ensembles(cf. ref. [58]). Un modèle mathématique de machine Learning est entraîné avec un jeu de données d'apprentissage. Cet entraînement consiste à déterminer la bonne méthode à utiliser ainsi que les paramètres mathématiques du modèle retenu. Les spécialistes du machine Learning testent habituellement différentes méthodes de classification pour identifier celle qui est la plus efficace compte-tenu du jeu de données d'entraînement, c'est-à-dire, celle qui génère un maximum de bonnes réponses pour un test réalisé avec un jeu de données en entrées qui sont déjà classées mais qui n'ont pas servi à l'entraînement du modèle.

2.3.2.1.2 Régression : La régression permet de prédire une valeur numérique y en fonction d'une valeur x à partir d'un jeu d'entraînement constitué de paires de données (x, y). On peut par exemple prédire la valeur d'un bien immobilier ou d'une société en fonction de divers paramètres les décrivant. Les schémas ci-dessous qui illustrent ce concept utilisent uniquement une donnée en entrée et une en sortie. Dans la pratique, les régressions utilisent plusieurs paramètres en entrée. Les jeux de donnée en entrée comprennent plusieurs variables (x, y, z, ...). Il existe différentes formes de régression, notamment linéaire et non linéaire. S'y ajoute aussi la notion d'overfitting et d'underfitting, qui décrit les méthodes de régression qui suivent plus ou moins de près les variations observées. Il faut éviter les deux et trouver le juste milieu ! C'est le travail des data scientists. Les régressions peuvent être aussi réalisées avec des arbres de décision (CART), des modèles SVM, des réseaux de neurones, etc.

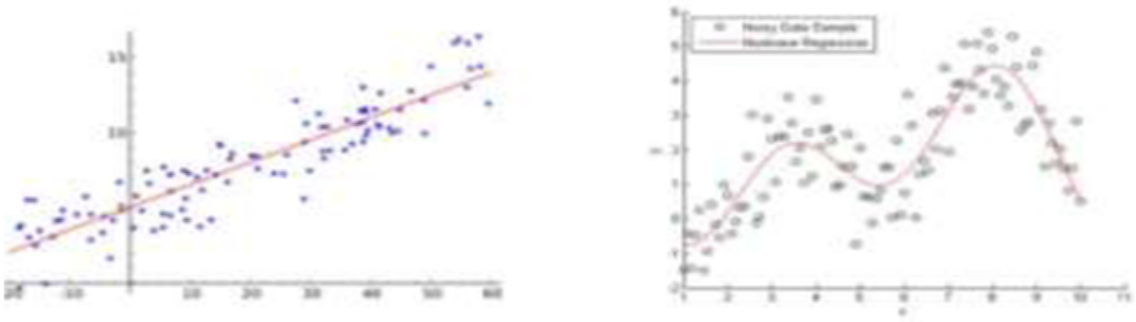


FIGURE 2.3: régression linéaire et non linéaire

2.3.2.2 L'apprentissage non supervisé :

Avec le clustering et la réduction de dimensions. Il exploite des bases de données non labellisées. Ce n'est pas un équivalent fonctionnel de l'apprentissage supervisé qui serait automatique. Ses fonctions sont différentes. Le Clustering permet d'isoler des segments de données spatialement séparés entre eux, mais sans que le système donne un nom ou une explication de ces clusters. La réduction de dimensions (ou embedding) vise à réduire la dimension de l'espace des données, en choisissant les dimensions les plus pertinentes. Du fait de l'arrivée des Big data, la dimension des données a explosé et les recherches sur les techniques d'embedding sont très actives. (cf. ref. [28]).

2.3.2.2.1 Clustering : Le clustering ou la segmentation automatique est une méthode d'apprentissage non supervisé qui permet à partir d'un jeu de données non labellisé d'identifier des groupes de données proches les unes des autres, les clusters de données. La technique la plus répandue est l'algorithme des k-moyennes (k-means). Les méthodes de Clustering permettent d'identifier les paramètres discriminants de ces différents segments. Elles servent ensuite à prévoir l'appartenance à un segment d'une nouvelle donnée entrée dans le système. Là encore, si le clustering peut être automatisé, en mode non supervisé, le choix du modèle de clustering ne l'est pas nécessairement pour autant sauf dans des outils avancés comme ceux de DataRobot et Prevision.io. Le machine Learning à base de réseaux de neurones permet de son côté de segmenter des données avec une répartition quasi-arbitraire

alors que les méthodes élémentaires ci-dessus sont limitées de ce point de vue-là. (cf. ref. [28]).

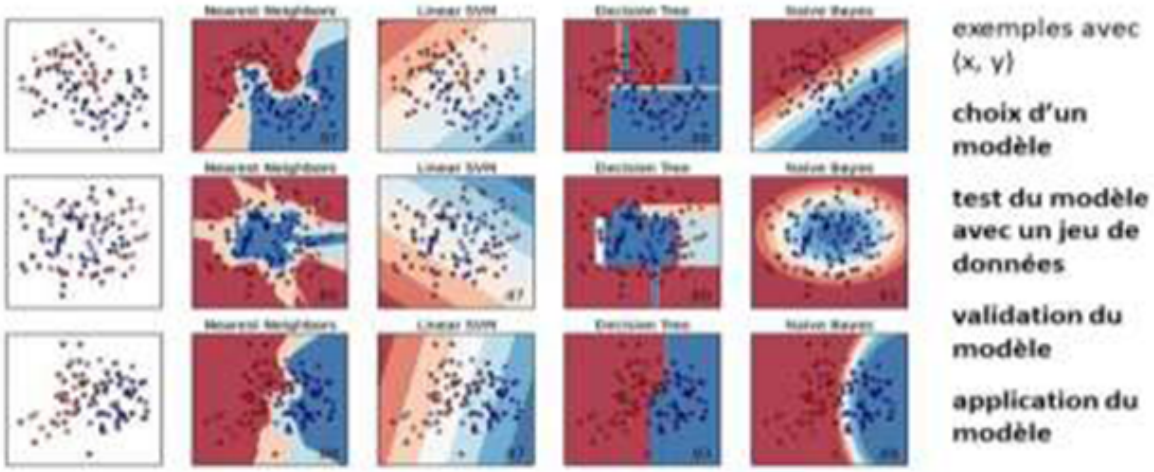


FIGURE 2.4: choix d'un modèle

2.3.2.2.2 Réduction de la dimensionnalité : La dimension des données devient de plus en plus grande à cause de la variété des Big data. Un bon nombre d'algorithmes souffrent de la malédiction des grandes dimensions (« curse of dimensionality »). Il existe donc des techniques de nombreuses méthodes de réduction de dimension. Les les plus classiques consistent à plonger les données (on parle d'embedding) dans un espace de plus faible dimension, de façon à préserver certaines propriétés. Parmi ces techniques : • l'Analyse en Composantes Principales (ou PCA, « Principal Component Analysis ») • l'Analyse Discriminante (Linear Discriminant Analysis : LDA) La réduction du nombre de variables utilisées va aussi réduire la consommation de ressources machines. Mais attention, les variables discriminantes ou facteurs de corrélation ne sont pas forcément des facteurs de causalité.

Ces derniers peuvent être externes aux variables analysées Les techniques de réduction de dimension, et notamment la PCA, sont très largement utilisées dans le machine Learning et le Deep Learning (qui a sa propre version de la PCA calculée par un réseau de neurones, dit auto-enccoder). La réduction du nombre de variables utilisées va aussi réduire la consommation de ressources machines. Mais attention, les variables discriminantes ou facteurs de corrélation ne sont pas forcément des facteurs de causalité. Ces derniers peuvent être externes aux variables analysées.(cf. ref. [4]).

2.3.2.3 L'apprentissage par renforcement :

Pour l'ajustement de modèles déjà entraînés en fonction des réactions de l'environnement. C'est une forme d'apprentissage supervisé incrémental qui utilise des données arrivant au fil de l'eau pour modifier le comportement du système. C'est utilisé par exemple en robotique, dans les jeux ou dans les chatbots capables de s'améliorer en fonction des réactions des utilisateurs. Et le plus souvent, avec le sous-ensemble du machine Learning qu'est le Deep Learning. L'une des variantes de l'apprentissage par renforcement est l'apprentissage su-pervisé autonome notamment utilisé en robotique où l'IA entraîne son modèle en déclenchant d'elle-même un jeu d'actions pour vérifier ensuite leur résultat et ajuster son comportement. (cf. ref. [28]).

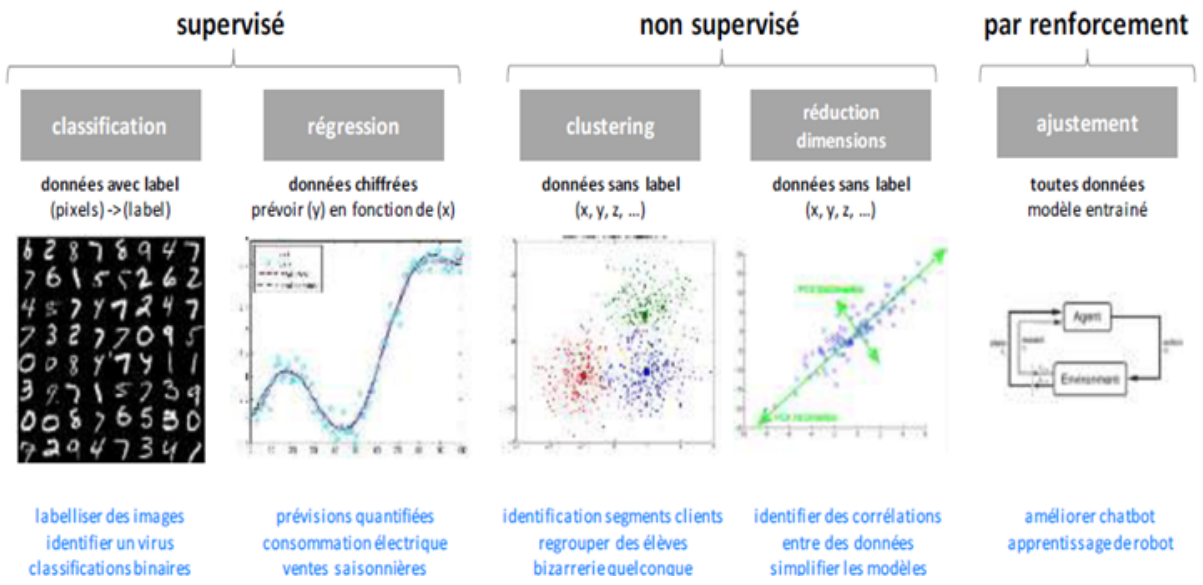


FIGURE 2.5: catégorie de machine Learning

2.3.3 Outils du machine Learning :

Il existe un très grand nombre d'outils de machine learning. Ils combinent plusieurs types de logiciels :

* Des langages de programmation comme Python, Java, C++ ou autres qui sont utilisés conjointement avec des bibliothèques de calcul spécialisées dans le machine Learning.

* Des bibliothèques associées, comme scikitlearn, qui permettent de développer les modèles d'apprentissage et de les mettre ensuite en production. Ces outils tournent sur poste de travail et dans le Cloud.

* Des environnements de travail, ou IDE pour Integrated Development Environment, qui per-mettent de paramétrer ses systèmes et de visualiser les résultats, souvent de manière graphique. Ils servent à tester différentes méthodes de classification, régression et clustering pour définir les modèles à appliquer.

* Des outils d'automatisation de la recherche de méthodes d'apprentissage comme DataRobot.

* Des outils destinés aux utilisateurs pour leur permettre d'analyser leurs données et de produire des rapports graphiques pertinents en se passant théoriquement de data scientists.

* Des outils pour la création de solutions de machine learning pour les objets connectés. (cf. ref. [28]).

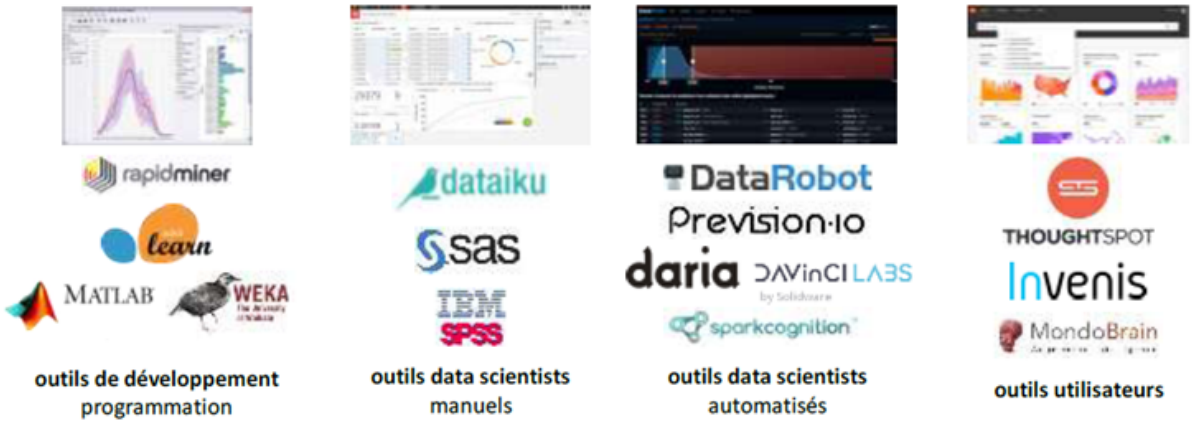


FIGURE 2.6: Outils de machine Learning

2.4 Deep Learning :

2.4.1 Définition :

Le deep learning est un sous-ensemble des techniques de machine Learning à base de réseaux de neurones qui s'appuient sur des réseaux de neurones à plusieurs couches dites cachées. Celles-ci permettent par exemple de décomposer de manière hiérarchique le contenu d'une donnée complexe comme de la voix ou une image pour la classifier ensuite :

identifier des mots pour la voix ou associer des tags descriptifs à des images. (cf. ref. [28]).

2.4.2 Principe de Deep Learning :

C'est le principe de l'une des grandes catégories de réseaux de neurones de deep learning, les réseaux convolutionnels ou convolutifs (schéma ci-dessous). Un réseau peut être profond mais aussi large si le nombre de neurones est élevé dans chaque couche. Le deep learning remplace les méthodes antérieures du machine learning dites à base de « handcraft features » qui consistaient à défi-nir à la main les éléments à rechercher dans les objets (formes dans les images, etc). Dans le deep learning, notamment pour la détection d'images, le réseau de neurones découvre tout seul ces composantes avec des niveaux d'abstraction évoluant de bas en haut et de couche en couche.

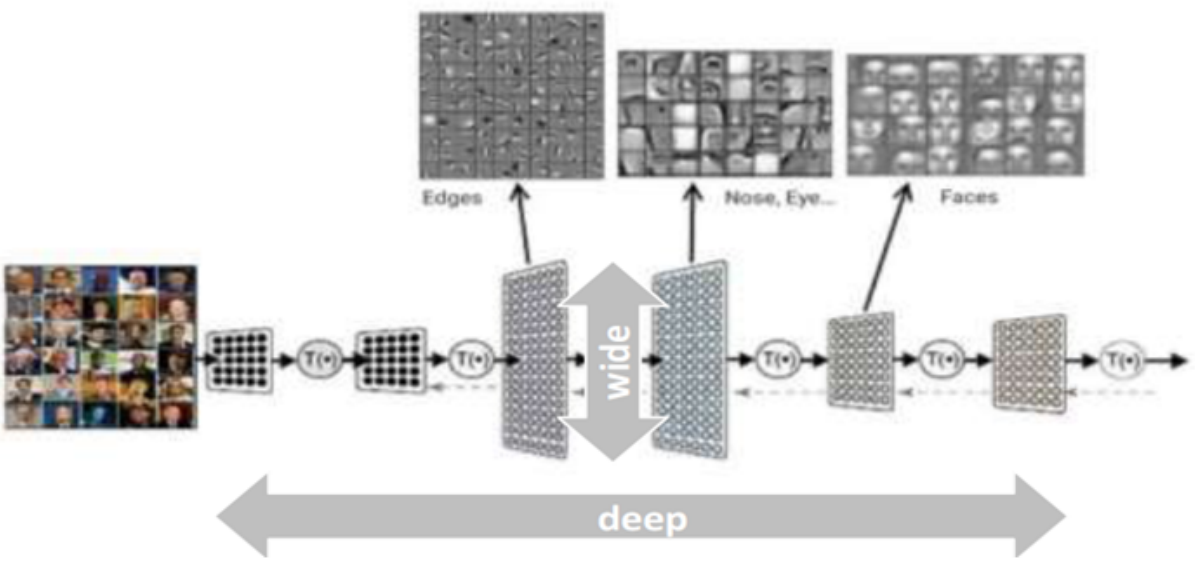


FIGURE 2.7: principe de Deep Learning

Chaque niveau d'un réseau de neurones profond réalise une fonction de classification d'un type d'objet en un type d'objet supérieur (pixels => forme=> nez=> visage => individu) Un réseau de neurones profond de type convolutionnel comprend plusieurs couches « cachées » qui transforme les données en entrée en données ayant un niveau d'abstraction supérieur

Le Deep Learning sert le plus souvent au traitement du langage, de la parole, du bruit, de l'écriture et des images. Il a d'autres usages dans les outils d'aide à la décision, dans les jeux tels que le Go avec AlphaGo et même dans l'exploitation de données structurées, dans la cybersécurité et d'une manière générale dans la recherche scientifique comme en génomique. (cf. ref. [28]).

2.4.3 Evolutions du deep learning :

Les outils de Deep Learning s'appuient sur différentes variantes de réseaux de neurones pour leur mise en œuvre pratique. Leur histoire remonte aux perceptrons de Franck Rosenblatt de 1957.

* perceptrons 1957 * multi-layeredperceptron 1969 * back propagation 1974 * recurrent neural networks 1982(cf. ref. [25]).(cf. ref. [38]).(cf. ref. [62]).(cf. ref. [62]).(cf. ref. [46]). * neocognitrons1983 * error propagation 1986 * restricted boltzmannmachine1986 * time delay neural networks 1989(cf. ref. [18]).(cf. ref. [23]).(cf. ref. [24]). * forward propagation199X * convolutional neural networks1998(cf. ref. [**?**]).(cf. ref. [20]).(cf. ref. [19]).(cf. ref. [21]).(cf. ref. [22]). * deepbeliefnetworks2006(cf. ref. [59]). * stackedautoencoders2007 * googleimage-

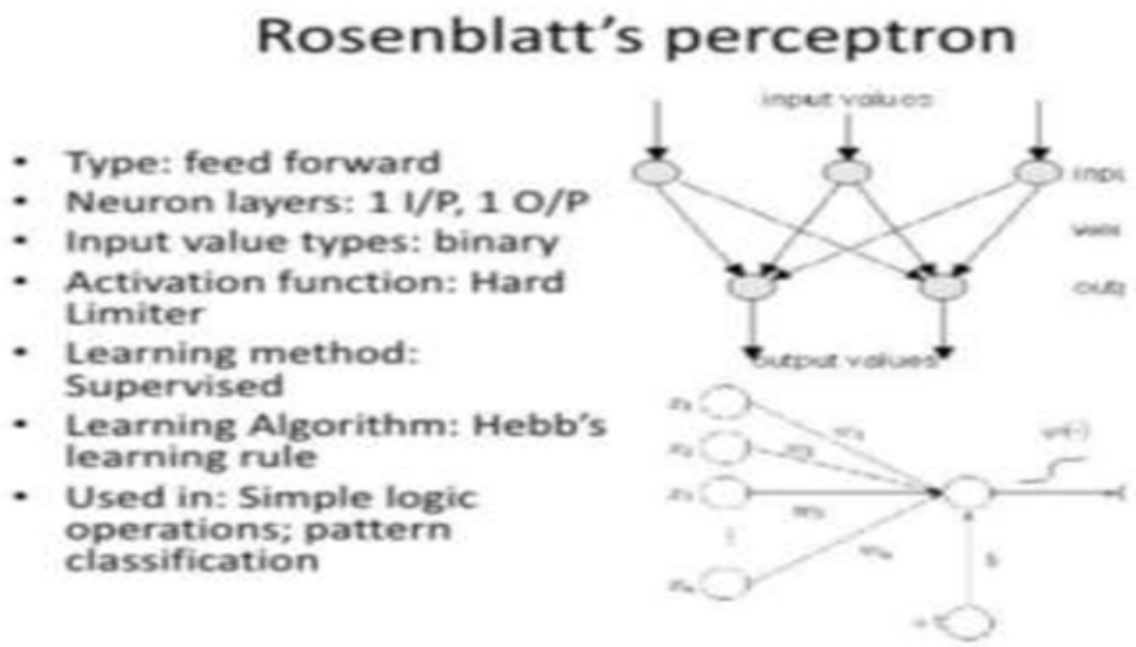


FIGURE 2.8: perceptron de rosenblatt

net2012

2.4.4 Modes d'apprentissage de Deep Learning :

Comme pour la machine Learning, l'apprentissage de solutions de Deep Learning suit l'une des approches suivantes :

2.4.4.1 L'apprentissage supervisé :

Qui repose sur l'entraînement d'un réseau avec un jeu de données d'entraînement qui est associé à une donnée de résultat souhaité. Pour la reconnaissance d'images, il s'agit des descriptifs d'objets contenus par les images ou leur classe. Pour de la traduction automatique, ce sont des couples de phrases traduites d'une langue à l'autre. La labellisation est généralement manuelle et d'origine humaine. Tout du moins, dès

lors que l'on exploite du langage. Le tagging de données peut exploiter des sources non humaines, comme des capteurs sensoriels divers. (cf. ref. [28]).

2.4.4.2 L'apprentissage non supervisé :

Qui est utilisé dans certains types de réseaux de neurones de Deep Learning ou certaines parties de réseaux, comme les stacked autoencoders qui permettent d'identifier automatiquement des patterns dans des objets et de réaliser du clustering automatique d'objets. C'est un moyen de découvrir des classes d'objets alors que dans l'apprentissage supervisé, on entraîne le modèle avec des classes préétablies manuellement. Cet apprentissage non supervisé ne va pas identifier automatiquement le nom des classes identifiées. L'apprentissage totalement non supervisé est plus que rare. Le non supervisé est souvent le complément du supervisé. L'apprentissage non supervisé semble indiquer qu'une machine peut faire preuve de créativité ou d'intuition. En fait, il n'en est rien. L'apprentissage non supervisé permet d'identifier auto-matiquement des variables discriminantes d'un élément spécifique. Dans les chatbots et le traitement du langage, l'apprentissage non supervisé est une forme d'apprentissage par renforcement. C'est le cas si le chatbot s'entraîne à mieux répondre en examinant la manière dont évoluent les conversations avec les utilisateurs en fonction de leurs réponses. Cela implique donc encore une boucle avec des humains.

2.4.4.3 L'apprentissage par renforcement :

Qui consiste à faire évoluer un modèle en fonction de retours externes, en général avec le monde physique. Cette méthode vise à optimiser une récompense quantitative (reward) obtenue par le système en fonction de ses interactions avec son environnement. C'est une technique qui est par exemple utilisée pour optimiser le réalisme des dialogues de chatbots, dans les jeux vidéo ou en robotique. Elle l'est dans les robots qui apprennent à éviter les obstacles ou à réaliser des tâches mécaniques en tâtonnant. L'agent à entraîner par renforcement cherche à maximiser par itérations successives une récompense qui est incarnée par sa performance, telle que le temps pour réaliser une tâche donnée ou la qualité de cette tâche. L'apprentissage par renforcement peut nécessiter un grand nombre d'essais et de données. (cf. ref. [28]).

2.4.5 Outils du deep learning :

Pour suivons cette partie sur le Deep Learning en évoquant l'offre des outils de création des solutions les mettant en œuvre. Il s'agit d'outils de développement exploitant des langages déclaratifs comme Python. Ils permettent de créer des modèles de réseaux de neurones avec leurs différentes couches. La programmation consiste surtout à définir la structure du réseau de neurones : le nombre de couches cachées, la taille des filtres et des feature maps pour les réseaux de neurones convolutionnels, les fonctions de pooling (réduction de résolution), puis à déclencher son entraînement avec une boucle de programme qui va scanner un jeu d'entraînement taggé et faire de la rétropropagation de

gradient dans le réseau de neurones. Une fois entraîné, on évalue le taux d'erreurs généré sur un jeu de test et on affine tous les éléments ci-dessus dans ce que l'on appelle l'optimisation des hyperpa-ramètres. Les outils disponibles pour créer des solutions de deep learning sont le plus souvent disponibles en open source, installables sur les machines et serveurs des utilisateurs ou accessibles via des ressources serveur en cloud. Les grands acteurs du numérique proposent tous leurs frameworks et outils open source de création de réseaux de neurones : TensorFlow chez Google, Torch chez Facebook, Cortana NTK chez Microsoft, la plateforme Watson chez IBM ou encore DSSTNE chez Amazon. Mais les startups ne sont pas en reste, comme H2O, même si les outils de développement open source semblent avoir largement pris le dessus comme c'est le cas pour le développement Internet depuis 1995.(cf. ref. [28]).

FIGURE 2.9: Outils de Deep Learning

2.5 Conclusion :

Nous avons introduit le chapitre en donnant quelques notions de base de l'intelligence artificielle tel que la définition de l'IA, les trois grandes domaines de l'IA, la segmentation et les données de l'IA. Nous avons donné les principales concepts de machine Learning tel que la définition da machine Learning, les défirentes catégories de celle-ci les outils de machine Learning. Nous avons présenté les défirents concepts de Deep Learning : la définition de Deep Learning, son principe, l'évolution de celle-ci les défirentes catégories de Deep Learning les outils de Deep Learning. Dans le chapitre suivant en va présenter les défirent méthodes de GCNN graphe convolutionnel neuronal network.

Chapitre 3

Les méthodes de GCNN

3.1 Introduction :

Ce chapitre définit les déférentes méthodes de GCNN graphe convolutional noronal networks :Euclidean space et non euclidean space de domaine fixe et non euclidean space de domaine variable.il est organisé en trois parties : Partie 1 : représente les déférente concepts d' Euclidean convNet : l'architecture, les couches compositionnelles, domaine des données.... Partie 2 : représente les concepts de ConvNets spectrales pour les graphes fixes ; Graph Laplacian, Analyse de Fourier, SplineNets, ChebNet.... Partie 3 : représente les concepts de ConvNets spatial pour Variable Graphs : Graph neural networks, Vanilla graph ConvNets....

3.2 Domaine de données pour ConvNet :

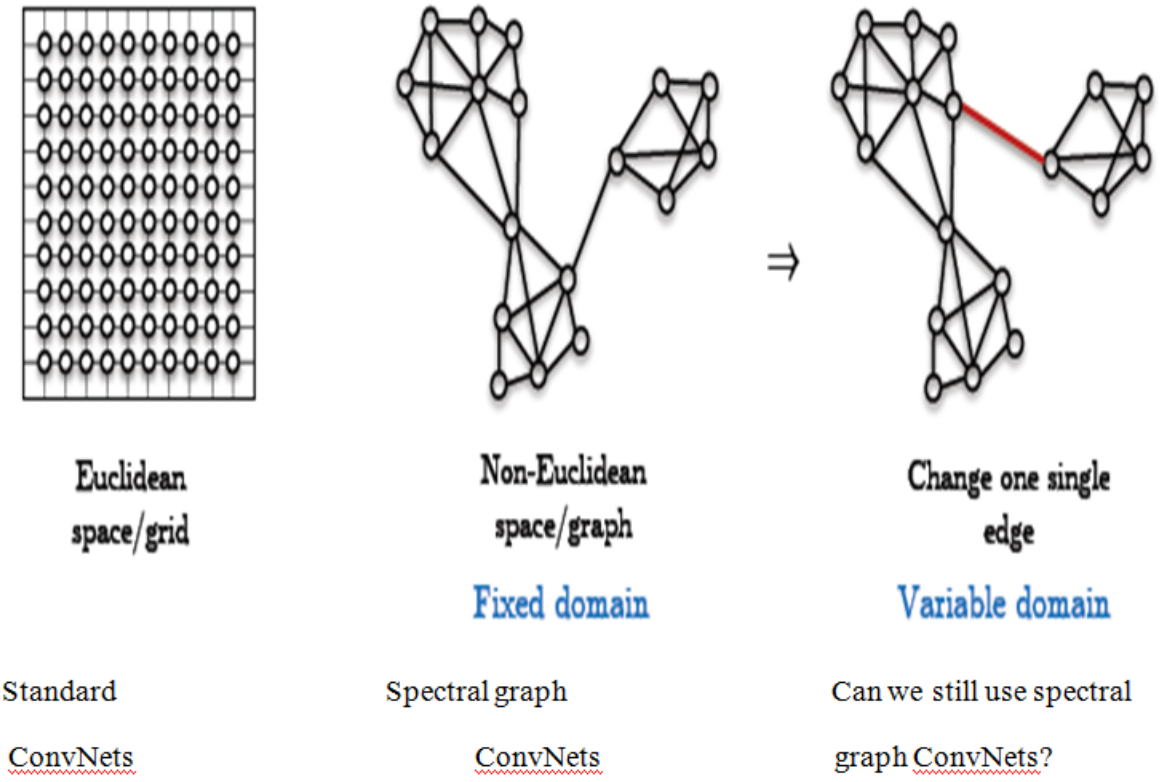


FIGURE 3.1: Standard ConvNets Spectral graph ConvNets de domaine fixe et variable

Les NN spectraux offrent une richesse familles de filtres spectraux Sont des filtres spectraux transférable ? (cf. ref. [8]).

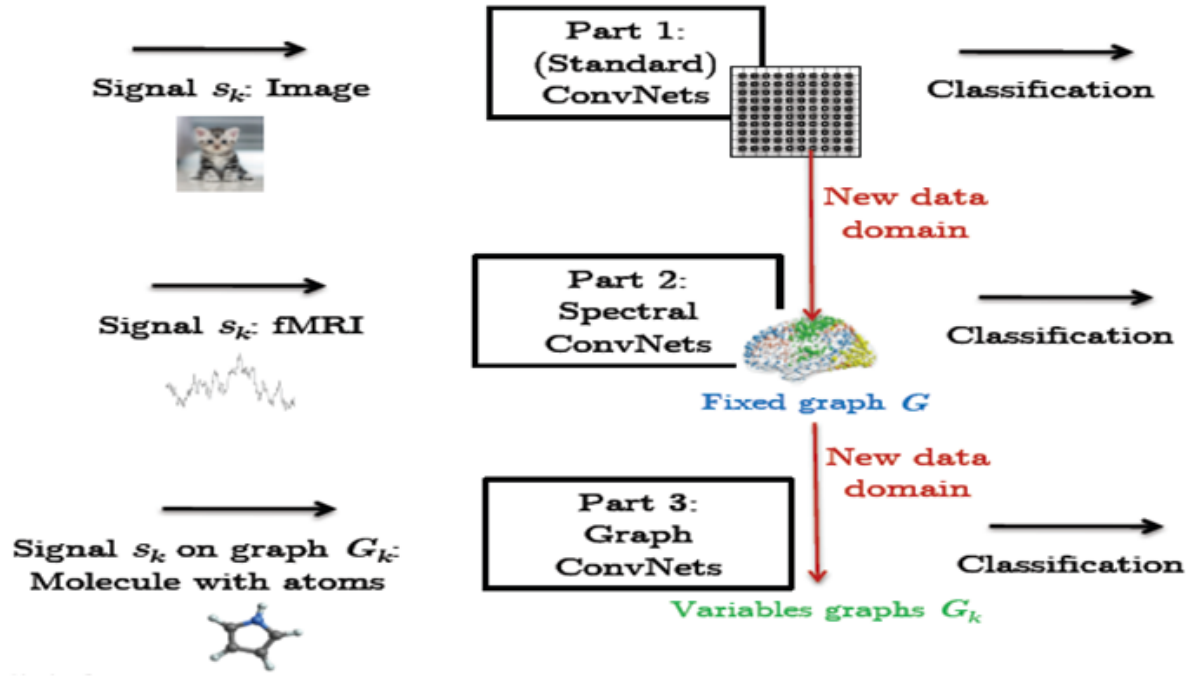


FIGURE 3.2: architecture de graphe convNet

3.3 Architecture de graphe convNet

3.4 Euclidean ConvNets :

3.4.1 Architecture :

Une architecture pour un apprentissage de grande dimension ConvNets sont puissants pour résoudre des problèmes d'apprentissage de dimensions élevées.(cf. ref. [8]).

3.4.2 couches compositionnelles :

$$\begin{aligned} \mathbf{f}_l &= l\text{-th image feature (R,G,B channels)}, \dim(\mathbf{f}_l) = n \times 1 \\ \mathbf{g}_l^{(k)} &= l\text{-th feature map}, \dim(\mathbf{g}_l^{(k)}) = n_l^{(k)} \times 1 \end{aligned}$$

Compositional features se composent de plusieurs couches convolu-

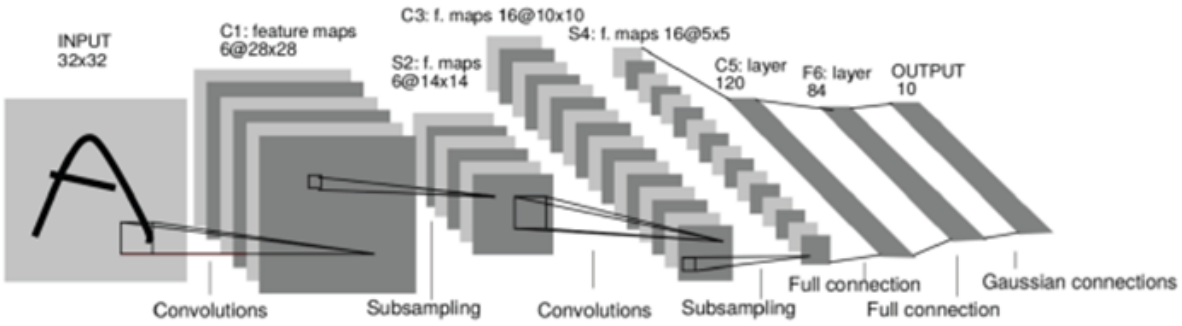


FIGURE 3.3: architecture d'apprentissage de dimension élevées

tionnelles et couche de pooling

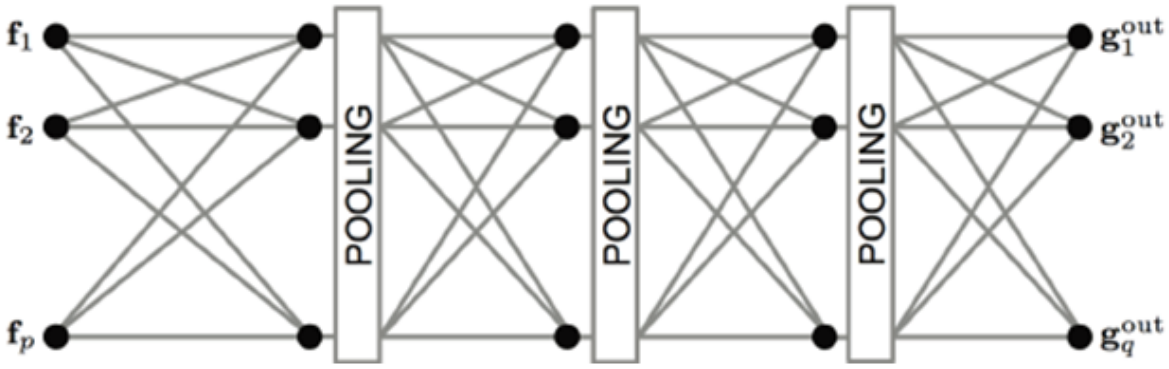


FIGURE 3.4: couches compositionnelle

Couche convolutionelle $\mathbf{g}_l^{(k)} = \xi\left(\sum_{l'=1}^{q_{k-1}} \mathbf{W}_{l,l'}^{(k)} \star \xi\left(\sum_{l'=1}^{q_{k-2}} \mathbf{W}_{l,l'}^{(k-1)} \star \xi\left(\cdots \mathbf{f}_{l'}\right)\right)\right)$

Activation $\xi(x) = \max\{x, 0\}$ rectified linear unit (ReLU)

Pooling $\mathbf{g}_l^{(k)}(x) = \|\mathbf{g}_l^{(k-1)}(x') : x' \in \mathcal{N}(x)\|_p \quad p = 1, 2, \text{ or } \infty$

3.4.3 Domaine de données pour ConvNets :

Image, volume, video : 2D,3D, 2D+1 Euclidean domains. Text , mot, son 1D : Euclidean domains.

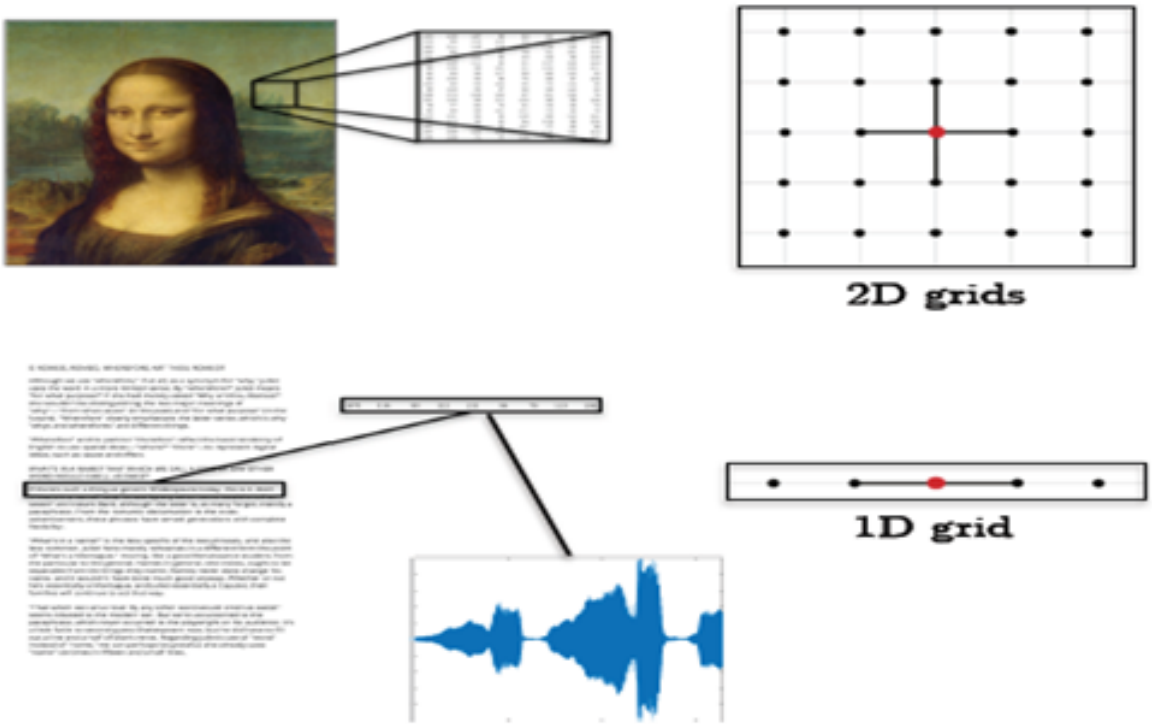


FIGURE 3.5: Domaine de données Euclidien de ConvNet

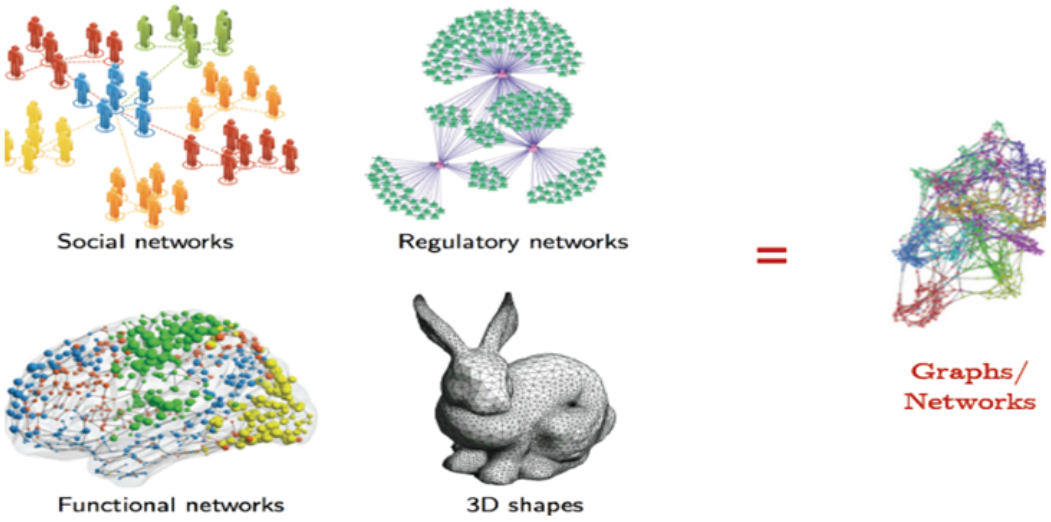


FIGURE 3.6: Domaine de données non Euclidien de ConvNet

3.5 ConvNets spectrales pour les graphes fixes :

3.5.1 Théorie des graphes spectrales :

3.5.1.1 Les graphes :

Graphe $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

Sommets $\mathcal{V} = \{1, \ldots, n\}$

Bords $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$

Poids des sommets $b_i > 0$ for $i \in \mathcal{V}$

Poids des bords $a_{ij} \geq 0$ for $(i, j) \in \mathcal{E}$

Champs de sommet $L^2(\mathcal{V}) = \{f : \mathcal{V} \to \mathbb{R}^h\}$

Représenté comme $\mathbf{f} = (f_1, \ldots, f_n)$

Hilbert space avec *inner product* $\langle f, g \rangle_{L^2(\mathcal{V})} = \sum_{i \in \mathcal{V}} a_i f_i g_i$

(cf. ref. [8]).

3.5.1.2 Graph Laplacian :

Opérateur laplacien

$$\Delta : L^2(\mathcal{V}) \to L^2(\mathcal{V})$$

$$(\Delta f)_i = \frac{1}{b_i} \sum_{j:(i,j) \in \mathcal{E}} a_{ij}(f_i - f_j)$$

Différence entre f et sa moyenne locale (2e dérivée sur les graphes) Core

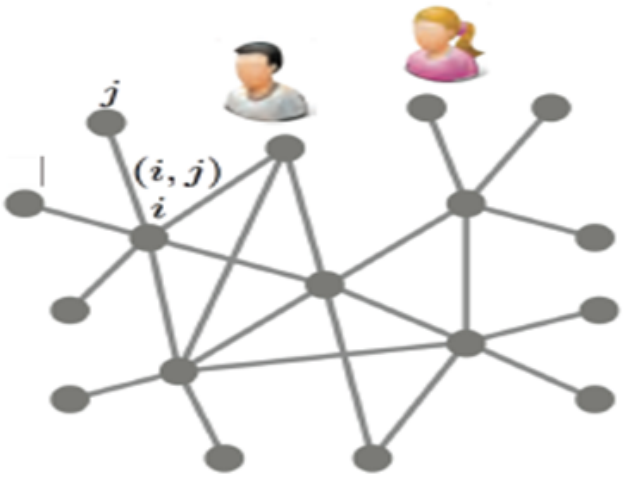


FIGURE 3.7: un graphe

operator dans la théorie des graphes spectraux. Représenté comme une matrice n n semi-définie positive

- Laplacien non normalisé $\Delta = \mathbf{D} - \mathbf{A}$
- Laplacien normalisé $\Delta = \mathrm{I} - \mathrm{D}^{-1/2}\mathrm{A}\mathrm{D}^{-1/2}$
- *Random walk Laplacian* $\Delta = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$

Tel que $\mathbf{A} = (a_{ij})$ et $\mathbf{D} = \mathrm{diag}(\sum_{j \neq i} a_{ij})$

(cf. ref. [8]).

3.5.1.3 Décomposition spectrale :

Un laplacien d'un graphe de n sommets admet n vecteurs propres :

$$\Delta \phi_k = \lambda_k \phi_k, \qquad k = 1, 2, \ldots$$

Les vecteurs propres sont réels et orthonormés (self-adjointness)

$$\langle \phi_k, \phi_{k'} \rangle_{L^2(\mathcal{V})} = \delta_{kk'}$$

Les valeurs propres ne sont pas négatives (positive-semidefiniteness)

$$0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$$

Les vecteurs étrangers laplaciens sont également appelés fonctions / modes de base de Fourier. Eigendecomposition du graphe laplacien :

$$\Delta = \Phi^T \Lambda \Phi$$

Telque $\Phi = (\phi_1, \ldots, \phi_n)$et $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$

(cf. ref. [8]).

3.5.1.4 Modes de Fourier :

Euclidean domain :

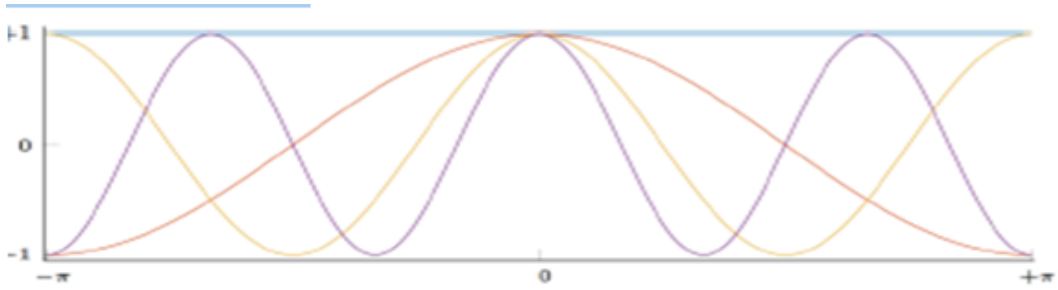


FIGURE 3.8: premier eigenvectors de 1D Euclidean laplacien =base standard de fourier

Graph domain :

Lap eigenvectors liés à la géométrie du graphe

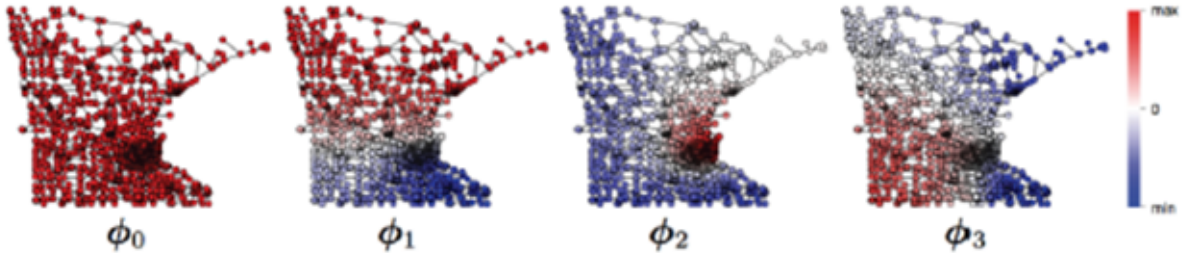


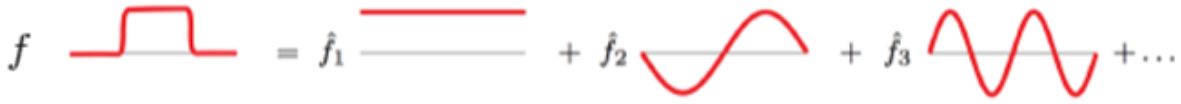
FIGURE 3.9: Premier Laplacien eigenvectors d'un graphe

3.5.1.5 Analyse de Fourier :

3.5.1.5.1 Analyse de Fourier euclidienne :

Une fonction $f : [-\pi, \pi] \to \mathbb{R}$ peut s'écrire en série de Fourier:

$$f(x) = \sum_{k \geq 0} \underbrace{\frac{1}{2\pi} \int_{-\pi}^{\pi} f(x') e^{-ikx'} dx'}_{\hat{f}_k = \langle f, e^{-ikx} \rangle_{L^2([-\pi,\pi])}} e^{-ikx}$$



Base de fourier e^{-ikx} = *Laplace-Beltrami eigenfunctions:*

$$-\Delta \phi_k = k^2 \phi_k$$

$$\begin{cases} \phi_k & = & \text{Fourier mode} \\ k & = & \text{frequency of Fourier mode} \end{cases}$$

3.5.1.5.2 Analyse de Fourier sur les graphes :

Une fonction $f : \mathcal{V} \to \mathbb{R}$ peut s'écrire comme série de Fourier :

$$f_i = \sum_{k=1}^{n} \underbrace{\langle f, \phi_k \rangle_{L^2(\mathcal{V})}}_{\hat{f}_k} \phi_{k,i}$$

$\hat{f}_k$ est le **k**ième coefficient de graphique du Fourier.

En notation matrice-vecteur, avec la matrice de Fourier n x n $\Phi = [\phi_1, \ldots, \phi_n]$

$$\hat{\mathbf{f}} = \Phi^\top \mathbf{f} \text{ and } \mathbf{f} = \Phi \hat{\mathbf{f}}$$

Fourier transform ← → Inverse Fourier transform

Graphe de la base de Fourier ϕ_k = Laplacian eigenvectors :

$$\Delta \phi_k = \lambda_k \phi_k \quad \textbf{with} \quad \begin{cases} \phi_k & = & \text{graph Fourier mode} \\ \lambda_k & = & \text{(square) frequency} \end{cases}$$

3.5.1.6 Convolution :

3.5.1.6.1 Convolution dans un espace euclidien discret :

Convolution de deux vecteurs $\mathbf{f} = (f_1, \ldots, f_n)^\top$ et $\mathbf{g} = (g_1, \ldots, g_n)^\top$

$$(\mathbf{f} \star \mathbf{g})_i = \sum_m g_{(i-m) \bmod n} \cdot f_m$$

$$\mathbf{f} \star \mathbf{g} = \underbrace{\begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix}}_{\text{Circulant matrix}} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

diagonalised by Fourier basis (Toeplitz)

$$= \Phi \begin{bmatrix} \hat{g}_1 & & \\ & \ddots & \\ & & \hat{g}_n \end{bmatrix} \Phi^\top \mathbf{f} = \Phi(\Phi^\top \mathbf{g} \circ \Phi^\top \mathbf{f})$$

3.5.1.6.2 Convolution sur les graphes

La convolution spectrale de $f, g \in L^2(\mathcal{V})$ peut être définie par analogie

$$(f \star g)_i = \underbrace{\sum_{k \geq 1} \underbrace{\langle f, \phi_k \rangle_{L^2(\mathcal{V})} \langle g, \phi_k \rangle_{L^2(\mathcal{V})}}_{\text{product in the Fourier domain}} \phi_{k,i}}_{\text{inverse Fourier transform}}$$

En notation matrice-vecteur :

$$\begin{aligned} \mathbf{f} \star \mathbf{g} &= \mathbf{\Phi}(\mathbf{\Phi}^\top \mathbf{g} \circ \mathbf{\Phi}^\top \mathbf{f}) \\ &= \underbrace{\mathbf{\Phi}\,\mathrm{diag}(\hat{g}_1, \ldots, \hat{g}_n)\mathbf{\Phi}^\top}_{\mathbf{G}} \mathbf{f} \\ &= \mathbf{\Phi}\hat{g}(\mathbf{\Lambda})\mathbf{\Phi}^\top \mathbf{f} = \hat{g}(\mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\top)\mathbf{f} = \hat{g}(\mathbf{\Delta})\mathbf{f} \end{aligned}$$

Non shift-invariant (G n'a pas de structure circulante).

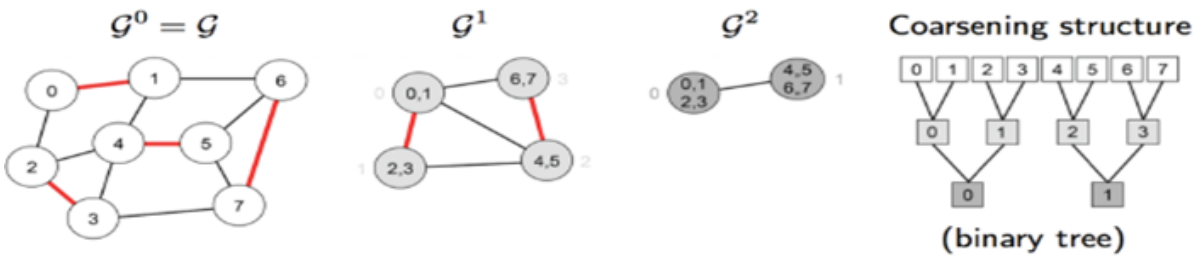
Les coefficients de filtrage dépendent de la base $\phi_1, \ldots, \phi_n$.

Calcul coûteux (pas de FFT): O (n^2).

(cf. ref. [8]).

3.5.1.7 Fast graph pooling :

Pooling structuré Disposition de l'indexation des nœuds telle que les nœuds adjacents sont fusionnés hiérarchiquement au niveau le plus grossier suivant.



Aussi efficace que le 1D-Euclidean grid pooling.

3.5.2 Spectral ConvNets :

3.5.2.1 SplineNets :

3.5.2.1.1 graphe spectral vanille ConvNets : couche convolutionnelle de graphe fl = l-ième data feature sur les graphes, dim (fl) = n* 1 gl = l-ème feature map, dim (gl) = n*1.

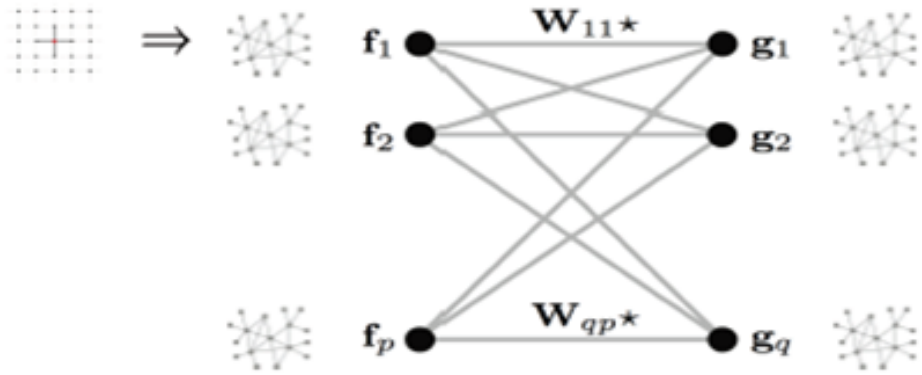


FIGURE 3.10: Couche convolutionelle

Activation, par ex $\xi(x) = \max\{x, 0\}$ unité linéaire rectifiée (ReLU).

3.5.2.1.2 Convolution du graphe spectral : Couche convolutionelle dans le domaine spatial :

Où W l,l =(n* n) matrice diagonale du filtre de graphe spectral. Nous désignerons le filtre spectral sans le symbole du chapeau, c'est-à-dire W l,l(cf. ref. [8]).

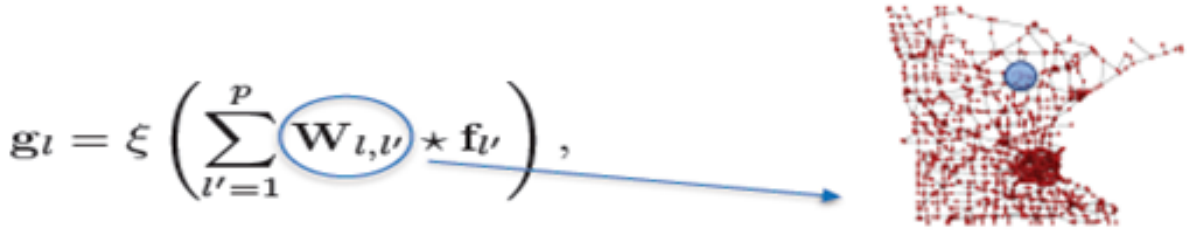


FIGURE 3.11: filtre du graphe spatial

Où $W_{l,l'}$ = matrice du filtre du graphe spatial,

Peut également s'exprimer dans le domaine spectral (using $g \star f = \Phi\, \hat{g}(\Lambda)\Phi^{\top} f$)

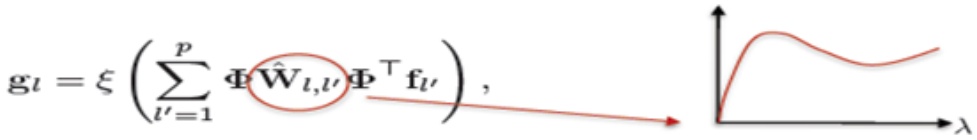


FIGURE 3.12: filtre de graphe spectral

3.5.2.1.3 SplineNets : Série de couches consolatives spectrales :

$$g_l^{(k)} = \xi\left(\sum_{l'=1}^{q^{(k-1)}} \Phi W_{l,l'}^{(k)} \Phi^{\top} g_{l'}^{(k-1)}\right)$$

Avec des coefficients spectraux W l,l à apprendre à chaque couche. Première architecture du graphe spectral CNN Aucune garantie de localisation spatiale des filtres Paramètres O (n) par couche et O (n2) computation of forward et transformées de Fourier inverse , T (pas de FFT sur les graphes) Les filtres dépendent de la base ne se généralise pas entre les graphiques(cf. ref. [8]).

3.5.2.1.4 localisation spatial et Lissage spectral

Dans le cadre euclidien (par l'identité de Parseval)

$$\int_{-\infty}^{+\infty} |x|^{2k} |w(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{w}(\lambda)}{\partial \lambda^k} \right|^2 d\lambda$$

Localisation dans l'espace = smoothness dans le domaine fréquentiel(cf. ref. [8])

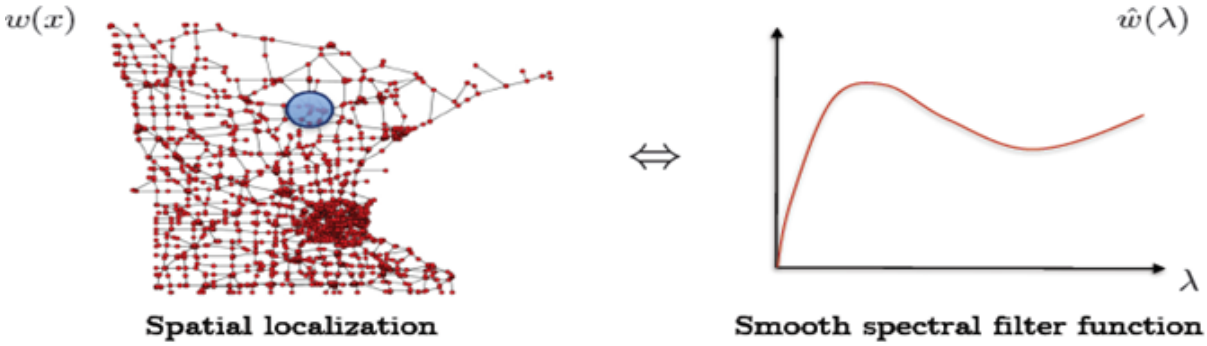


FIGURE 3.13: localisation spatial et fonction de filtre de smooth spectral

3.5.2.2 ChebNets :

3.5.2.2.1 Filtres polynomiaux spectraux :

Représenter des fonctions spectrales smooth avec des polynômes de Laplacian eigenvalues :

$$w_{\boldsymbol{\alpha}}(\lambda) = \sum_{j=0}^{r} \alpha_j \lambda^j$$

où $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r)^\top$ est le vecteur des paramètres de filtre.

Couche convolutionnelle: appliquez un filtre spectral au *feature signal f*

$$w_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} = \sum_{j=0}^{r} \alpha_j \boldsymbol{\Delta}^j \mathbf{f}$$

Observation clé : chaque laplacien opération augmente le soutien d'un fonction par 1 saut Contrôle exact la taille des filtres à base de laplacien.

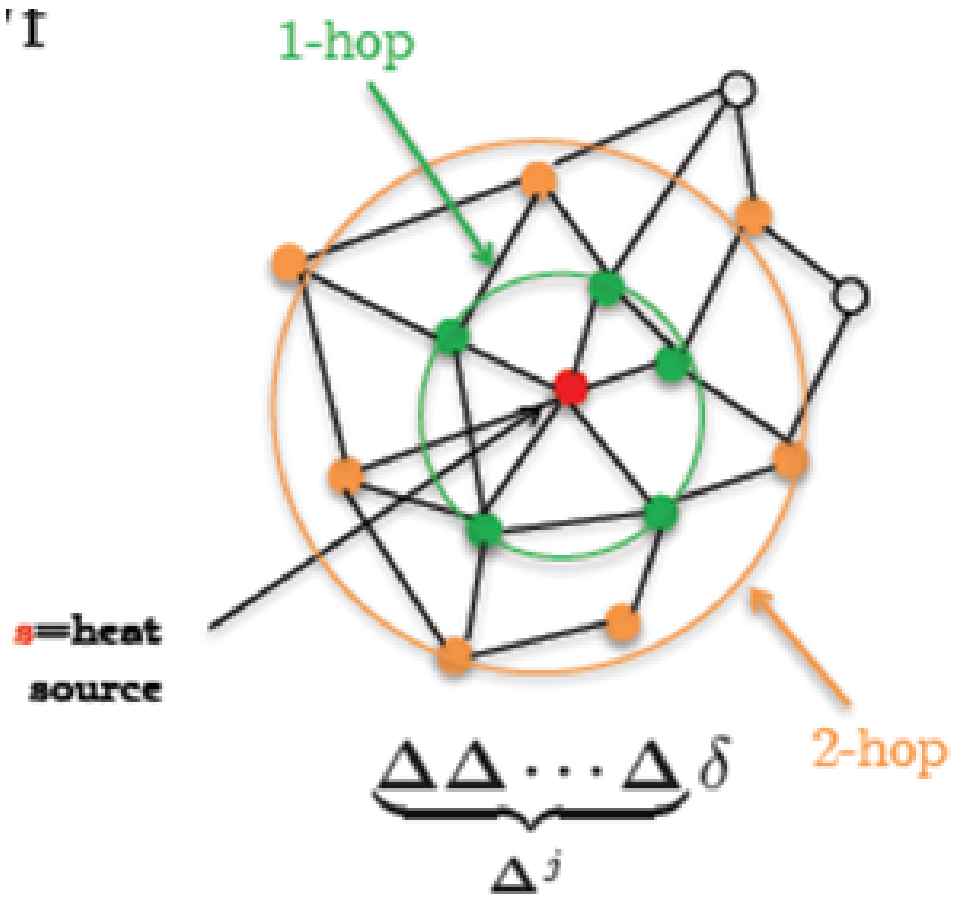


FIGURE 3.14: application d'un filtre spectral au feature signal f

3.5.2.2.2 Graphe spectral ConvNets avec filtres polynomiaux : Série de couches consolatives spectrales :

$$\mathbf{g}_i^{(k)} = \xi\left(\sum_{i'=1}^{q^{(k-1)}} \mathbf{\Phi}\mathbf{W}_{i,i'}^{(k)}\mathbf{\Phi}^\top \mathbf{g}_{i'}^{(k-1)}\right)$$

Avec des coefficients polynomiaux spectraux w l ,l à apprendre à chaque couche. Les filtres sont exactement localisés dans le support

de r-hops O (1) paramètres par couche Pas de computation de , T O (n) complexité de computation calcul (en supposant des graphes sparsely-connected) Instabilité sous perturbation de coefficients (difficile à optimiser) Les filtres dépendent de la base ne se généralise pas entre les graphes.(cf. ref. [8]).

$$a^2 + b^2 = c^2$$

3.5.2.2.3 Chebyshev polynomials : Graphe de convolution avec base monomiale (non orthogonale) 1, x, x2 ...

$$w_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} = \sum_{j=0}^{r} \alpha_j \boldsymbol{\Delta}^j \mathbf{f}$$

Graphe de convolution avec des polynômes de chebychev (orthogonaux).

$$w_{\boldsymbol{\alpha}}(\tilde{\boldsymbol{\Delta}})\mathbf{f} = \sum_{j=0}^{r} \alpha_j T_j(\tilde{\boldsymbol{\Delta}})\mathbf{f}$$

Orthonormal sur

$$L^2([-1,+1]) \text{ w.r.t. } \langle f, g\rangle = \int_{-1}^{+1} f(\tilde{\lambda}) g(\tilde{\lambda}) \frac{d\tilde{\lambda}}{\sqrt{1-\tilde{\lambda}^2}}$$

Stable sous perturbation de coefficients :

3.5.2.2.4 ChebNets : Application du filtre avec le scaled laplacien

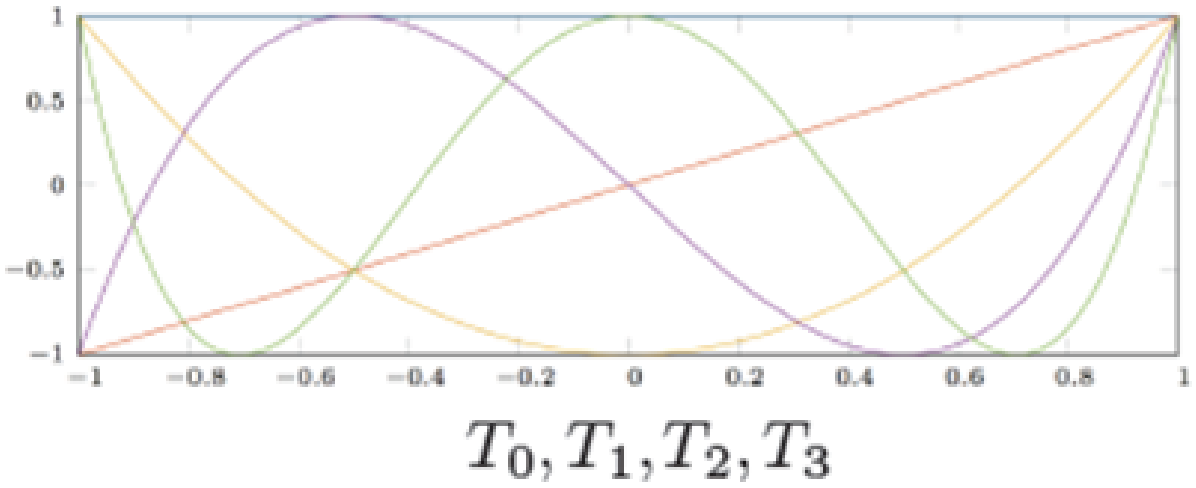


FIGURE 3.15: Chebyshev polynomials

$$\tilde{\mathbf{\Delta}} = 2\lambda_n^{-1}\mathbf{\Delta} - \mathbf{I}$$

$$w_{\boldsymbol{\alpha}}(\tilde{\mathbf{\Delta}})\mathbf{f} = \sum_{j=0}^{r} \alpha_j T_j(\tilde{\mathbf{\Delta}})\mathbf{f} = \sum_{j=0}^{r} \alpha_j \mathbf{X}^{(j)}$$

Avec

$$\begin{aligned} \mathbf{X}^{(j)} &= T_j(\tilde{\mathbf{\Delta}})\mathbf{f} \\ &= 2\tilde{\mathbf{\Delta}}\mathbf{X}^{(j-1)} - \mathbf{X}^{(j-2)}, \qquad \mathbf{X}^{(0)} = \mathbf{f}, \quad \mathbf{X}^{(1)} = \tilde{\mathbf{\Delta}}\mathbf{f} \end{aligned}$$

Les filtres sont exactement localisés dans le support de r-hops O (1) paramètres par couche Pas de computation de , T O (n) complexité de computation (en supposant des sparsely-connected graphes) Stabilité sous perturbation de coefficients Les filtres dépendent de la base ne se généralise pas entre les graphes.(cf. ref. [8]).

3.5.2.2.5 Graph convolutional nets ChebNets simplifiés :

Utiliser des polynômes de Chebychev de degré r = 2 et supposer $\lambda_n \approx 2$

$$\begin{aligned} w_\alpha(\Delta)\mathbf{f} &= \alpha_0\mathbf{f} + \alpha_1(\Delta - \mathbf{I})\mathbf{f} \\ &= \alpha_0\mathbf{f} - \alpha_1\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}\mathbf{f} \end{aligned}$$

Contrainte supplémentaire $\alpha = \alpha_0 = -\alpha_1$ pour obtenir un filtre à paramètre unique.

eigenvalues de $\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ sont maintenant dans [0,2]

⇒ application répétée des résultats de filtrage *in numerical instability*

appliquer une renormalisation

$$w_\alpha(\Delta)\mathbf{f} = \alpha\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{W}}\tilde{\mathbf{D}}^{-1/2}\mathbf{f}$$

$$\text{with} \quad \tilde{\mathbf{W}} = \mathbf{W} + \mathbf{I} \text{ and } \tilde{\mathbf{D}} = \text{diag}(\textstyle\sum_{j \neq i} \tilde{w}_{ij})$$

3.6 ConvNets pour Variable Graphs :

3.6.1 ConvNets pour Variable Graphes et fixe graphes :

ConvNets spectraux : étant donné fixe graphe (s) G, et un ensemble de signaux sk on G à analyser avec ConvNets : ConvNets pour les graphes arbitraires : Étant donné un ensemble de graphes Gk et de signaux sk sur Gk à analyser avec ConvNets :

3.6.2 Graph neural networks :

Technique NN spatiale pour traiter des graphes arbitraires . Inner-Structures minimales : Invariant par réindexation des sommets (aucun graphe matching n'est obligatoire) Localité (seuls neighbors sont pris en compte) Partage de poids (opérations convolutionnelles) Indépendance

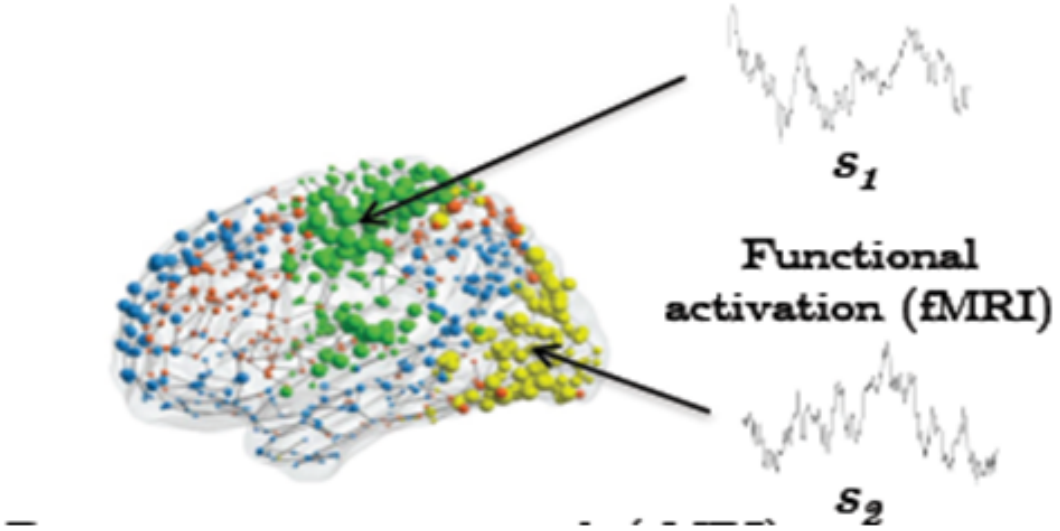


FIGURE 3.16: Brain connectivity network (sMRI) Fixed graph G

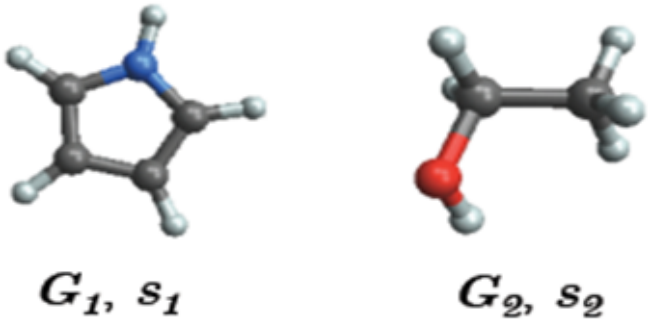


FIGURE 3.17: Molecules Variable graphs Gk

w.r.t. taille du graphe(cf. ref. [8]).

$$h_i = f_{\text{GNN}}\left(\{h_j : j \rightarrow i\}\right)$$

Quelle instanciation de f ?

3.6.3 Graph RNNs :

• Graph RNN : Multilayer perceptron • Graph GRU (Gated Re-

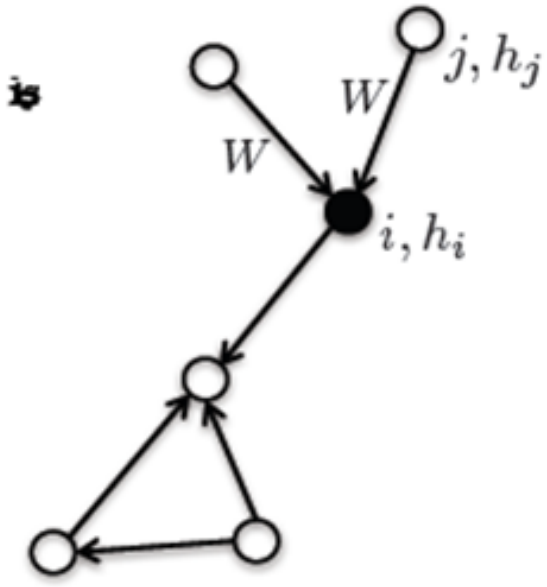


FIGURE 3.18: graphe arbitraire

$$h_i = \sum_{j \to i} \mathcal{C}_{\text{G-MLP}}(x_i, h_j) = \sum_{j \to i} A\sigma(B\sigma(Ux_i + Vh_j))$$

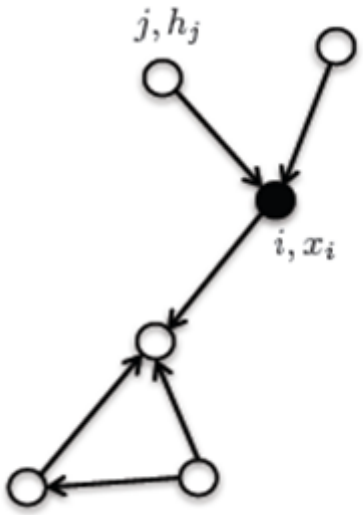


FIGURE 3.19: graphe variable

current Unit)

$$h_i = \mathcal{C}_{\text{G-GRU}}(x_i, \sum_{j \to i} h_j)$$

$$\begin{aligned}
\bar{h}_i^t &= \sum_{j \to i} h_j^t, \quad h_i^{t=0} = x_i \\
z_i^{t+1} &= \sigma(U_z h_i^t + V_z \bar{h}_i^t) \\
r_i^{t+1} &= \sigma(U_r h_i^t + V_r \bar{h}_i^t) \\
\tilde{h}_i^{t+1} &= \tanh\big(U_h(h_i^t \odot r_i^{t+1}) + V_h \bar{h}_i^t\big) \\
h_i^{t+1} &= (1 - z_i^{t+1}) \odot h_i^t + z_i^{t+1} \odot \tilde{h}_i^{t+1}
\end{aligned}$$

3.6.4 Graph ConvNets :

Vanilla graph ConvNets :

$$\begin{aligned}
h_i^{\ell+1} &= \mathcal{C}_{\text{G-VCN}}\Big(h_i^\ell, \sum_{j \to i} h_j^\ell\Big), \quad h_i^{\ell=0} = x_i \\
&= \mathbf{ReLU}\Big(U^\ell h_i^\ell + V^\ell \sum_{j \to i} h_j^\ell\Big), \quad h_i^{\ell=0} = x_i
\end{aligned}$$

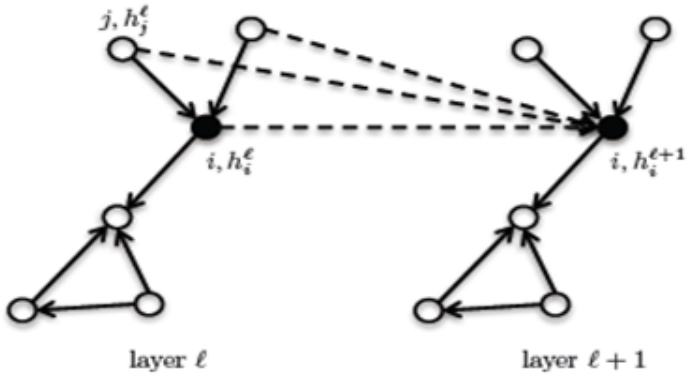


FIGURE 3.20: Vanilla graph ConvNets

3.7 Conclusion :

Nous avons introduit le chapitre en donnant l'architecture et le domaine de données de graphe convNet Nous nous sommes concentrés dans ce chapitre sur les diférentes méthodes de GCNN graphe convolutional noronal networks. Nous avons donné les principaux concepts de méthode Euclidean ConvNets. Nous avons présenté la méthode spectral de graphe fixe. Nous avons présenté la méthode spatial de graphe variable. Dans le chapitre suivant on va présenter la partie de l'implémentation qui va étudier l'utilisation des méthodes GCNN pour l'optimisation de recherche dans les algorithmes FSM

Chapitre 4

GCNN/FSM

4.1 Introduction :

Ce chapitre se concentre sur la partie implementation de notre mémoire en introduisant notre chapitre par les outils de l'implementation, on va présenter les différents domaines d'utilisation des FSM avec les réseau de neurone ,on va étudier les approches actuelles de l'extraction de sous-graphes,ensuite on va détailler l'approche qu'on va utiliser dans notre travail (SPminer)(cf. ref. [56])avec l'architecture GCNN et on va terminer notre chapitre avec la partie experimentale, qui se décopmpose en trois experiences : de petit motif, motif planté synthétique et large motif.

4.2 les outils de l'implementation :

4.2.1 Google colab :

4.2.1.1 Collaboration facile :

Comme son nom l'indique, Google Colab est un outil qui a été publié par Google et, de la même manière que d'autres produits Google, il est disponible dans le cloud et permet l'édition collaborative. Cela signifie que vous pouvez facilement partager le bloc-notes avec d'autres personnes, leur permettre d'afficher le code ou en faire des collaborateurs.(cf. ref. [40])

4.2.1.2 Bibliothèques préinstallées :

Google Colab vous permet d'écrire et d'exécuter du code Python de manière interactive de la même manière que Jupyter Notebook. De plus, il est livré avec la plupart des bibliothèques Data Science préinstallées, ce qui en fait l'outil idéal pour les débutants souhaitant commencer immédiatement les projets Machine Leaning. Des bibliothèques telles que pandas, numpy, Tensorflow, Keras, OpenCV sont déjà installées, il n'est donc pas nécessaire d'exécuter «pip install» et de se débattre avec l'installation de l'environnement local.(cf. ref. [40])

4.2.1.3 GPU / TPU gratuit :

De plus, il offre un GPU / TPU gratuit ! Cela signifie que la formation des réseaux de neurones sera plus rapide et accessible aux personnes ne disposant pas de machines très puissantes. Même si vous

disposez d'une machine puissante, Google Colab vous permettra de libérer votre machine des gros travaux et de la déléguer à un cloud.(cf. ref. [40])

4.3 Motivation :

une extraction fréquente de sous-graphes est un outil essentiel pour analyser des problèmes dans de nombreux domaines scientifiques, il vise à découvrir les subprimes le plus souvent apparaissant dans l'ensemble de données de graphique, qui peut avoir une fonctionnalité spéciale traitant les données en biologie, il peut être utilisé pour identifier les mécanismes communs d'interaction entre les protéines liées à une maladie humaine. et dans le réseau social, en tant que session sur les modèles d'interaction en sciences sociales, les relations entre les personnes sont importantes pour analyser le comportement et l'évolution du réseau. et en chimie, les groupes fonctionnels récurrents sont essentiels pour comprendre ce que je ressens des propriétésun tel corps solide et toxicité.

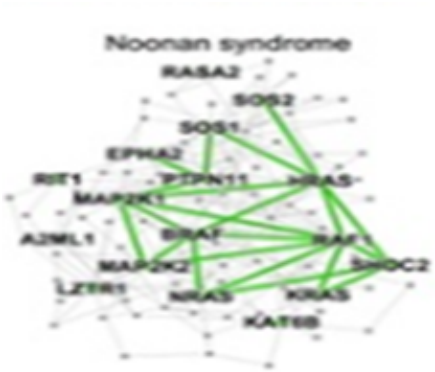


FIGURE 4.1: modèles d'interaction des protéines dans les voies de désaccord

FIGURE 4.2: modèles d'interaction dans les réseaux sociaux

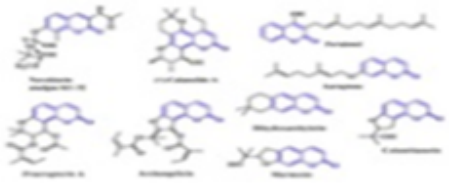


FIGURE 4.3: groupes fonctionnels pour les ensembles de données moléculaires

4.4 Analise du problème : extraction fréquente de sous-graphes :

Dans le problème de l'exploitation fréquente de sous-graphes, une méthode automatisée pour trouver ces modèles communs apparaissant dans ces réseaux réels. Pour formaliser le problème étant donné un graphe cible obtenir un paramètre de taille k puis je mettrai la taille R. D'identifier, parmi tous les graphes possibles de K nœuds, les R motifs de graphes avec la fréquence la plus élevée .

4.4.1 frequence de sous graphe :

Soit g k un motif de sous-graphe et gT les ensembles de données (dataset) du graphe tagret

4.4.2 Définition 1 niveau de graphe

Le nombre de sous-ensembles uniques de nœuds ST de GT pour lequel il existe un isomorphisme de ST aux nœuds de GQ.(cf. ref. [56])

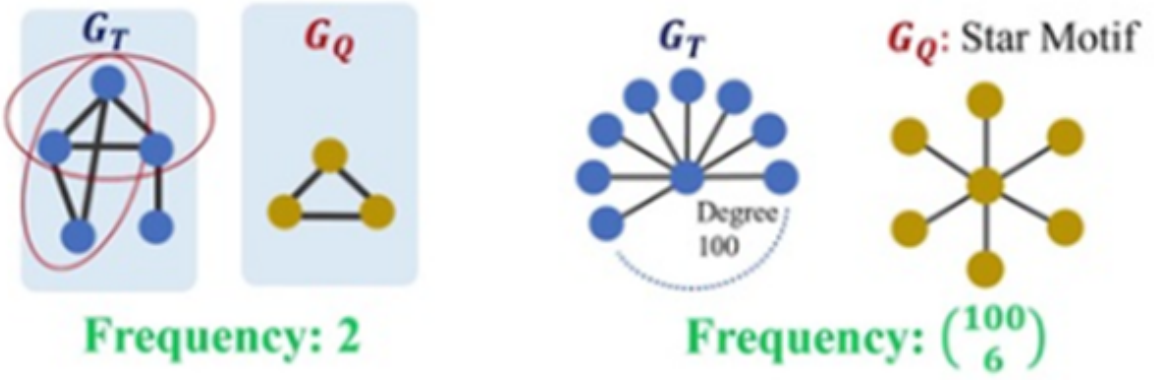


FIGURE 4.4: fréquence de sous graphe au niveau graphe

4.4.2.1 Définition 2 au niveau du nœud

Soit (GQ, V) un motif de sous-graphe ancré au nœud (nœud-anchored)

• Le nombre de nœuds u dans le voisinage GT contient GQ comme sous-graphe

• robuste des valeurs aberrantes

• utiliser la définition 2 pour concevoir le modèle

• en pratique, nous valorisons le modèle sous les deux définitions.(cf. ref. [56])

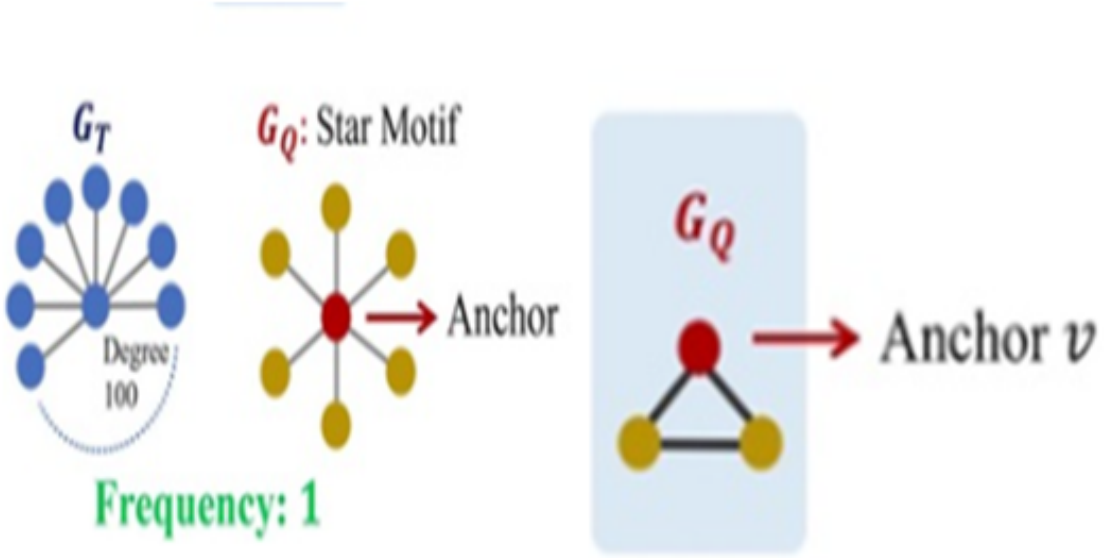


FIGURE 4.5: fréquent de sous graphe au niveau de nœuds

4.4.3 Approches actuelles de l'extraction de sous-graphes :

très difficile de trouver la fréquence de sous-graphe fréquent, et ceci pour deux raisons. Premièrement, il y a une explosion combinatoire du nombre de motifs possibles qui croît avec la taille du motif. Et deuxièmement, le sous-programme de l'isomorphisme du sous-graphe est NP-dur. L'énumération exacte avec l'heuristique ainsi que les méthodes existantes approximativement simples ne sont limitées que pour trouver de très petits modèles. Certaines approches de réseau neuronal ont été proposées pour résoudre des tâches de graphes connexes telles que la distance et le sous-graphe commun maximum. Mais de nouvelles

techniques doivent être développées pour relever ces compétitions ou défis comme expliqué ci-dessus dans notre travail.(cf. ref. [56])

4.5 Notre travail :

Dans notre travail nous proposons une représentation sur l'approche du sous-graphe fréquent. D'abord pour aborder l'explosion combinatoire du nombre de motifs. Nous avons organisé l'espace de recherche en insérant le sous-graphe dans un espace qui préserve l'ordre global, du sous-graphe. Et deuxièmement, résoudre la difficulté du sous-graphe, car un officier convertirait ce problème en tâche de prédiction neuronale, montrera que le test du sous-graphe en tant qu'officier peut être réduit à la comparaison de l'impédance parent à travers une opération efficace à temps constant. Et à notre connaissance. C'est la première approche neuronale de la fréquence de l'exploitation fréquente.

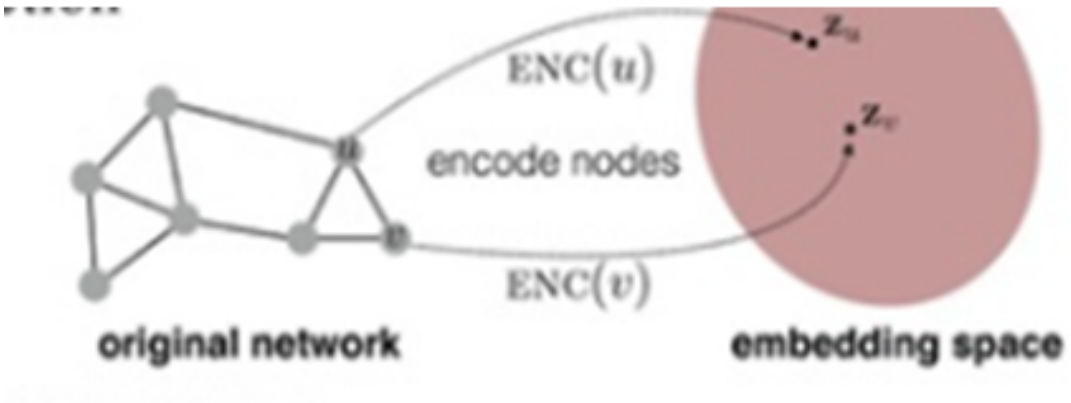


FIGURE 4.6: nœuds encodé

4.5.1 Notre approche SPMiner :

Notre approche, appelée Spmier , selon laquelle l'ensemble de données d'entrée doit être un graphe GT et il pourrait également être déconnecté. Et étant donné que le graphe d'entrée de SPMiner trouve le modèle d'échantillon qui se produit le plus fréquemment dans GT et comprend les étapes clés suivantes : Premièrement, nous décomposons GT en voisinages qui se chevauchent autour de GT supplémentaire avec encored est le centre des voisinage. Nous utilisons ensuite un encodeur GNN coûteux auquel ils peuvent mapper un motif distinct, en utilisant une technique appelée (espace d'inclusion d'ordre)(order embedding space) de sorte que nous pouvons efficacement prédire les relations de sous-graphe entre les parents et les voisinages , Et enfin, nous avons conçu une procédure de recherche en recherchant l'espace d'embarquement. Cela reste à supprimer le plus de voisinages possible.en tirant parti de la propriété de toute l'incorporation par le codeur.et nous présenterons chaque étape ci-dessous.(cf. ref. [56])

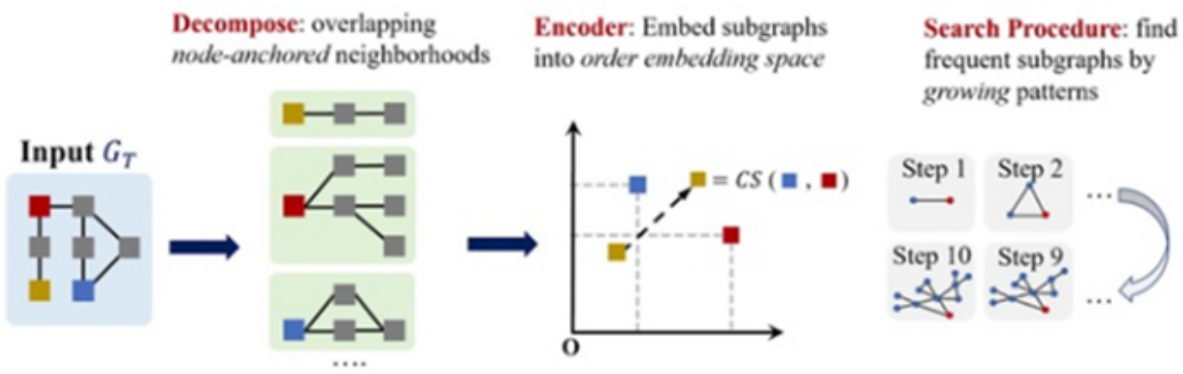


FIGURE 4.7: Approche SPMiner

4.5.1.1 Encodeur SPMiner :

D'abord dans chaque voisinages après la décomposition de GT en de CT voisinages, on utilise ensuite un GNN pour cartographier les voisinages en intégration. Et l'objectif du codeur est de modéliser la relation de sous-graphe entre les graphes encodés. Cela signifie donné sur un nœud encoré les graphes A et B. Le modèle peut prédire si A est un sous-graphe de B en utilisant l'intégration et nous démontrerons ici deux aspects essentiels : L'architecture GNN ainsi que l'espace d'intégration.

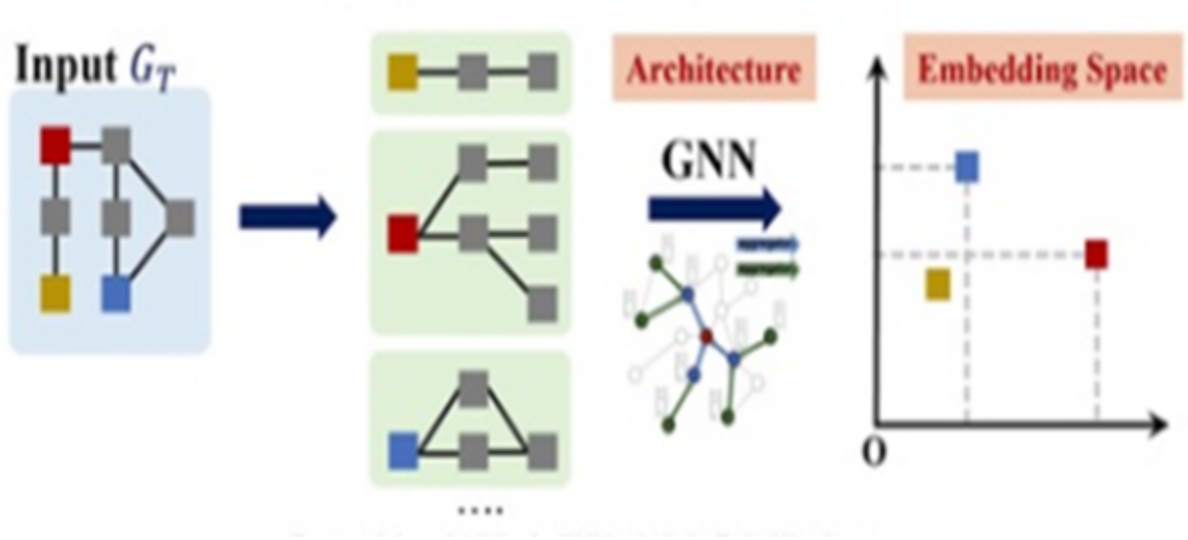


FIGURE 4.8: Encodeur SPMiner

4.5.1.2 Encodeur SPMiner : architecture GNN :

Notre architecture GNN est basée sur une combinaison de graphes C et G, plus complètement, nous concaténons le self de transformation et les messages de voisinage agrégés et chaque couche qui est similaire au graphe sage sur tout en utilisant une certaine agrégation similaire

à G. De plus, nous concevons une architecture de couche de saut apprenable qui, à notre avis, fonctionne particulièrement bien dans notre tâche. et différent de la prémisse GNN et de la couche, nous utilisons un filet de danse analogue de couche de saut entièrement connecté et assigné en plus un mot à chacun sauter la connexion de la couche A à la couche pour G et comme on peut le voir sur la figure avant l'entraînement, toutes les attentes de saut sont les mêmes. Mais après l'entraînement, certains sauts de chemins sont augmentés indiquant des sauts de connexions importantes du modèle, tandis que d'autres chemins de sauts sont diminués indiquant un saut de connexions qui ne sont pas très importants. Et intuitivement. Les intégrations en couche GNN nous décrivent la structure des voisinages de sauts. Venable saute permet à chaque modèle de couche d'accepter facilement des caractéristiques structurelles de voisinage de tailles différentes et cet intérêt est que seuls les réticules de connexion de saut utiles sont conservés.

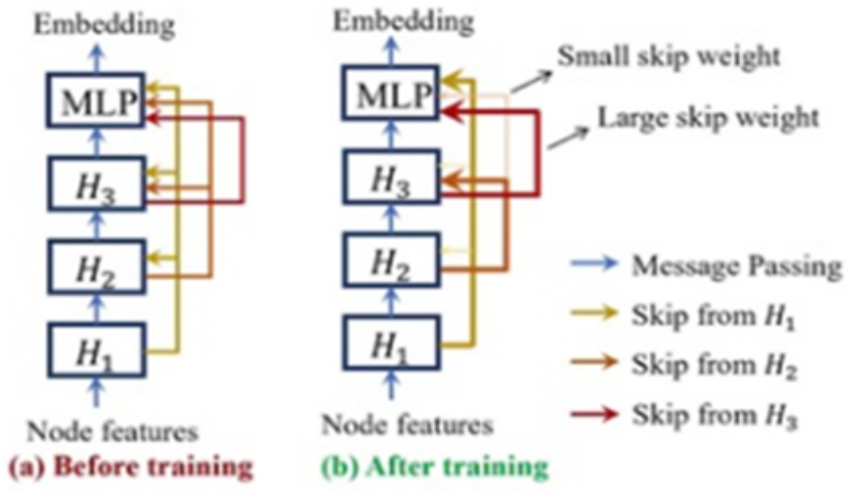


FIGURE 4.9: Encodeur SPMiner : architecture GNN

En utilisant cette tendance GNN. Nous pouvons concevoir une ar-

chitecture à deux tours pour prédire les relations de sous-graphes. Étant donné le modèle de sous-graphe et les voisinages, nous avons utilisé le GNN expliqué ci-dessus pour les inclure dans le graphe dans divers. Et nous pouvons ensuite faire une prédiction binaire finale de ce que les relations de sous-graphes tiennent en utilisant l'intégration à deux graphes. Et la question sexuelle ici est de savoir comment utiliser l'intégration pour faire des prédictions ? Bien sûr, nous pouvons utiliser une approche naïve pour concaténer les deux graphes et utiliser un MLP en plus de cela. Mais ce n'est pas très idéal en pratique car il ne capture pas la propriété de sous-graphe dans l'espace d'intégration. Et ici, nous visons à capturer ces propriétés de relation de sous-graphe en utilisant un espace d'intégration qui a des propriétés d'analogies, ce qui nous amène à l'utilisation de l'espace d'intégration d'ordre.

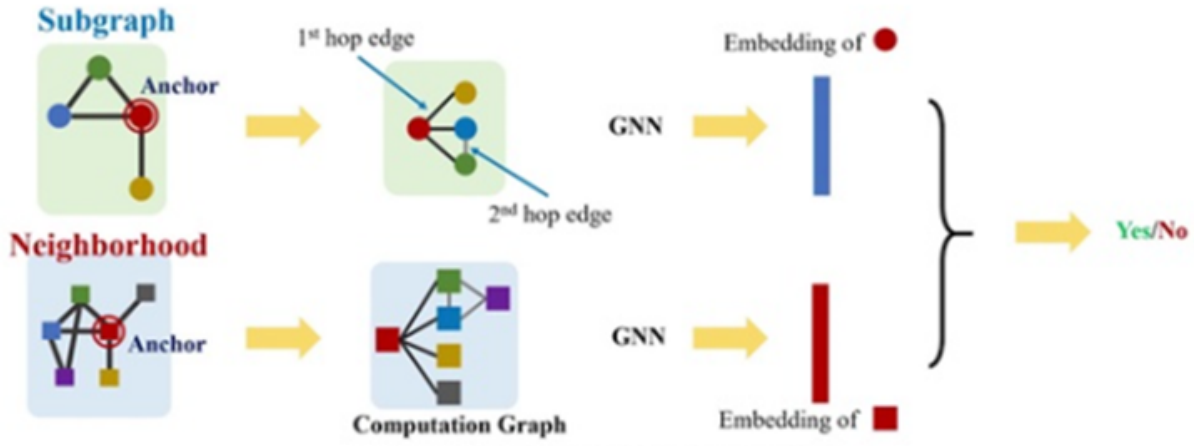


FIGURE 4.10: structure de graph neural networks à deux tours

4.5.1.3 SPMiner : commande d'espace d'intégration :

Nous avons observé trois propriétés clés des relations de sous-graphes. Transitivité et antisymétrie et intersections non vides : Sur la transitivité signifie simplement : A sous-graphe de B, et B sous-graphe de C, alors on sait que A sous-graphe de C. Et l'antisymétrie signifie que si un sous-graphe pour B alors B ne peut pas être un sous-graphe de A, et finalement aucune intersection vide signifie que, étant donné les graphes A et B Nous pouvons en fait trouver au moins un graphique d'hommage. C'est le sous-graphe commun pour les deux graphes. Donc pour capturer cela sur des propriétés séparées dans l'espace infini, nous utilisons cette technique d'intégration comme expliqué à droite.(cf. ref. [56])

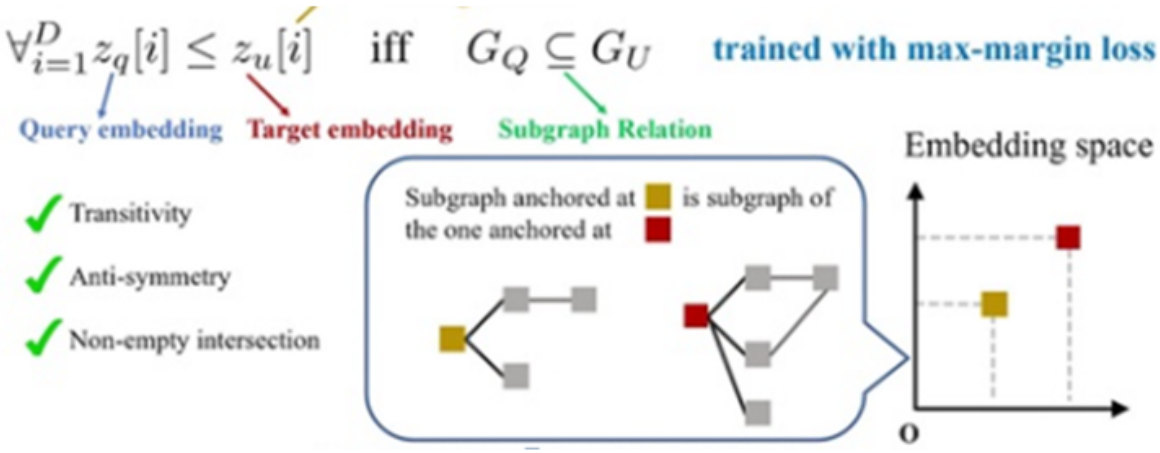


FIGURE 4.11: SPMiner : commande d'espace d'intégration

4.5.1.4 Procédure de recherche SPMiner :

Alors maintenant que nous avons formé l'intégration des commandes. Comment l'utilisons-nous habituellement pour identifier les motifs fré-

quents ? L'idée centrale ici est d'utiliser est de formuler ce problème comme un problème de recherche. Et le but est d'utiliser un algorithme de recherche pour trouver un modèle qui maximise le nombre de voisins et contienne un modèle. Ici, nous visualisons le déséquilibre de l'absence de voisins. Donc en bas à gauche c'est l'ordre espace invariable et chaque inventeur correspond à un voisin ancré ou à un motif de graphe.(cf. ref. [56])

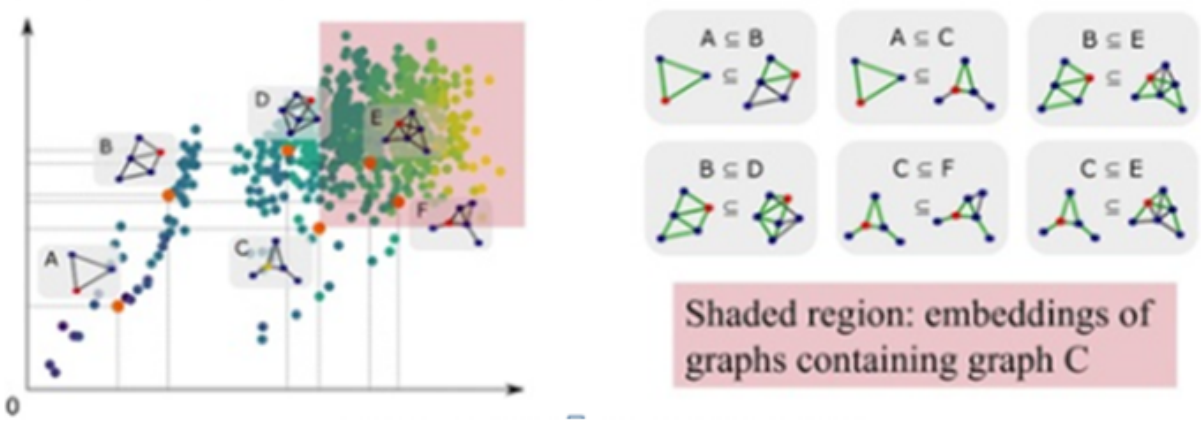


FIGURE 4.12: Procédure de recherche SPMiner(1)

Objectif : maximiser le nombre de points dans la région ombrée en rouge

Définir : violation totale d'un sous-graphe G : intuitivement le nombre de voisinages qui ne contiennent pas G

greedy stratégie

À chaque pas, ajoutez le nœud suivant qui entraîne la plus petite violation totale

Meilleures stratégies alternatives

- Beam Recherche

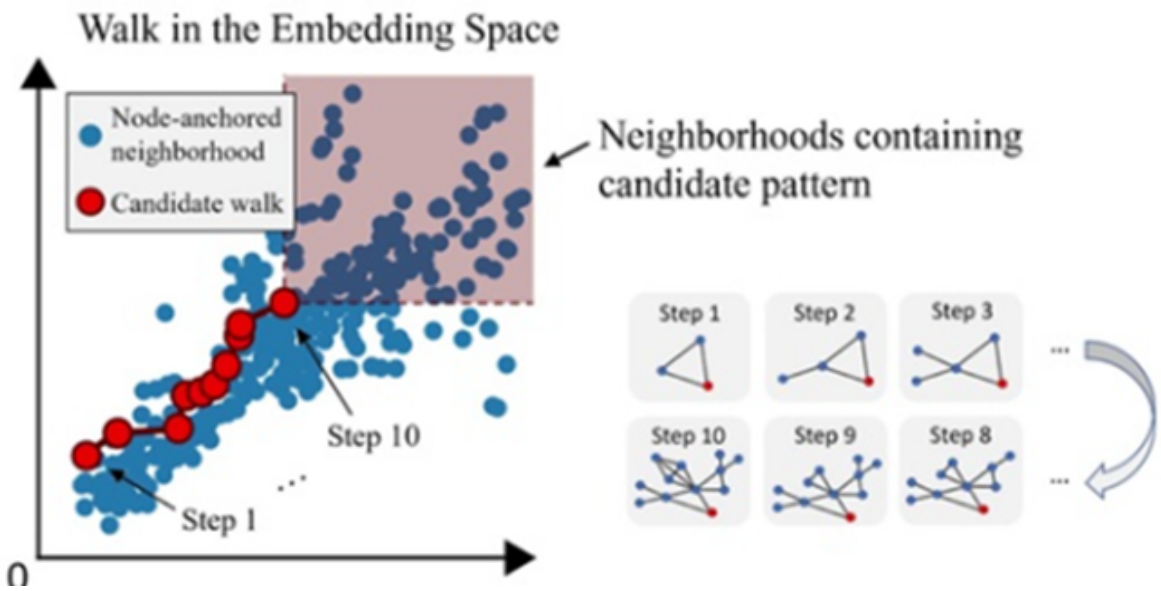


FIGURE 4.13: Procédure de recherche SPMiner(2)

• Recherche d'arbre de monte carlo.(cf. ref. [56])

4.6 Expérience :

Alors maintenant nous allons démontrer les expériences sur une variété d'ensembles de données synthétiques et réels dans plusieurs domaines d'application tels que la biologie, la chimie et les réseaux sociaux. Pour les lignes de base neuronales, nous comparons notre approche ou l'approche en cours à une architecture MLP standard sur plus GNN utilisant la perte d'entropie croisée, et nous avons également comparé avec deux méthodes d'extraction de sous-graphes approximatives populaires basées simplement qui appelle, et trouve, et loue.

* datastes

- datastes synthétiques

-ensembles de données du monde réel

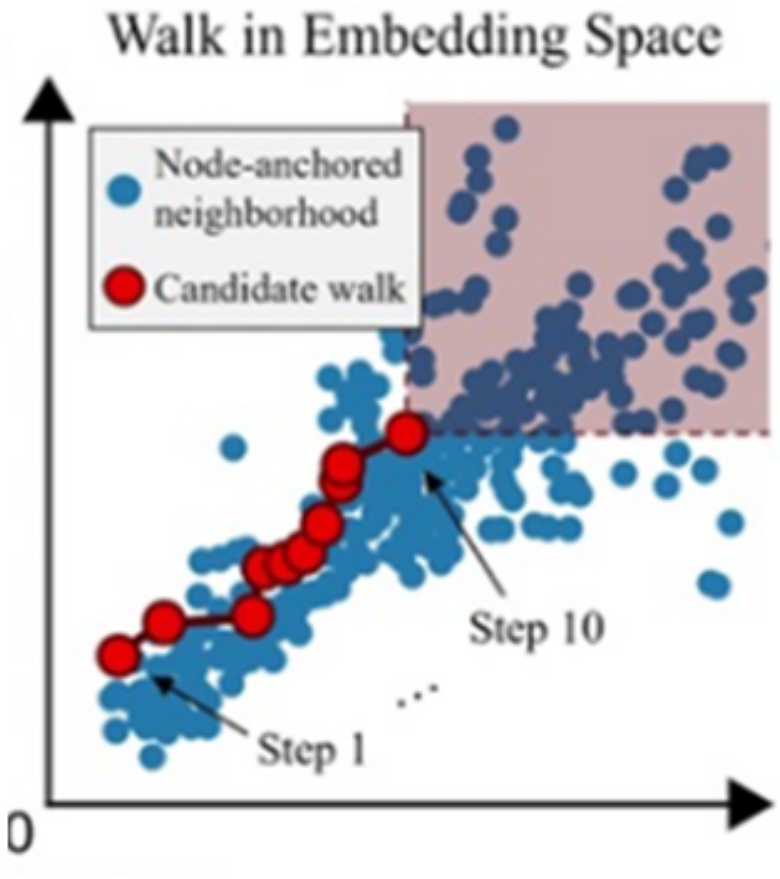


FIGURE 4.14: Procédure de recherche SPMiner dans un espace d'integration

-ENZYMES (biologie), COX2 (chimie), réseau routier

* référence

-neural : perceptron multicouche MLP

-inder MF basé sur l'échantillonnage RAND-ESU

4.6.1 Expérimentations de petits motifs :

Nous commençons donc par des expériences utilisant de petits motifs où nous pouvons réellement obtenir la vérité terrain. Grâce à un calcul exact. étant donné une dissidence immobilière, on retrouve son

top 10 des motifs de taille les plus fréquents. 5 et 6 Et nous évaluons combien de ces 10 principaux motifs peuvent être correctement identifiés par SPMiner. Comme le montre le chiffre, nous constatons que SpMiner identifie actuellement respectivement 9 et 8 des principaux motifs. Tandis que les lignes de base représentent plusieurs des principaux motifs. et en particulier si vous regardez les contre, la courbe bleue, la courbe bleue a représente les cancers de la vérité terrain. Ce sont donc les fréquences du top 10, les motivations les plus fréquentes d'une grande vérité. Et nous voyons que notre matière peut en fait identifier des motifs qui ont presque la même fréquence que les grands motifs de vérité, alors que les lignes de base sont un manque de ces motifs fréquents et que notre méthode fonctionne presque deux fois plus bien dans ce contexte.

*Vérité terrain :

En utilisant la calcule exacte, trouvez les 10 motifs les plus fréquents dans l'ensemble de données.

*Question : les lignes de base de SPMiner peuvent-elles identifier ce motif fréquent.

*Résultat : SPMiner identifie respectivement 9 et 8 des 10 meilleurs motifs.

4.6.2 Expérience : grands motifs :

Enfin, nous avons comparé la capacité de SpMiner et des lignes de base à identifier même les grands motifs plus une taille de 20. Nous n'avons plus une grande vérité dans ce cas, mais nous pouvons compa-

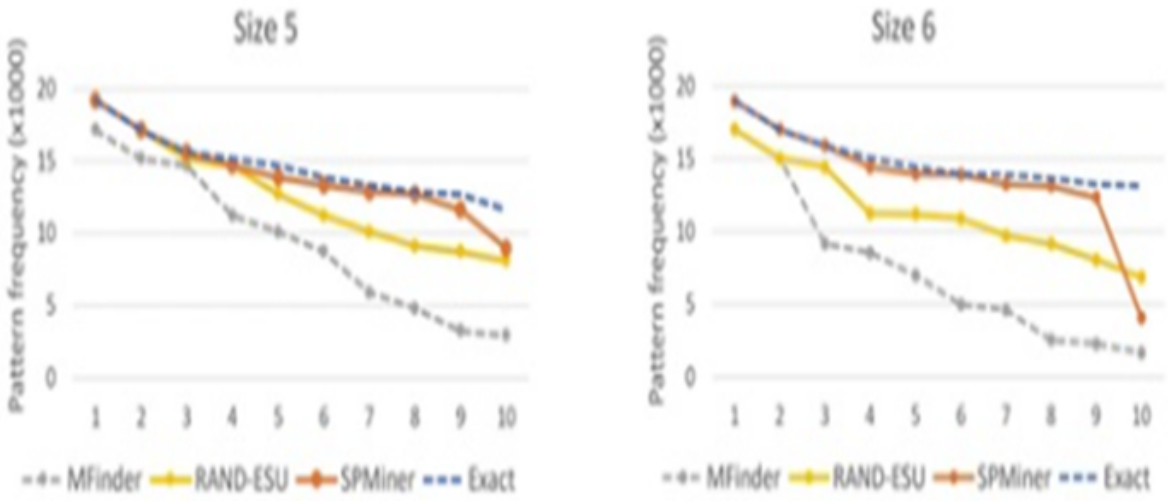


FIGURE 4.15: Expérimentations de petits motifs

rer la fréquence des motifs identifiés par SPMiner ainsi que les motifs identifiés par la ligne de base Et nous constatons que SPMiner identifie les motifs qui ont tendance à apparaître 10 -200 fois plus petite fréquence, puis les lignes de base. En particulier SPMiner et ces variantes TS vides sont capables de trouver de grands modèles récurrents d'enzymes de taille 50 et plus. Alors que les lignes de base se dégradent en raison de l'explosion combinatoire. Et nous voyons en quelque sorte la même transe qualitative, lorsque nous évaluons avec un niveau ou une fréquence grave. Et nous montrons que le SPMiner a les mêmes avantages empiriques sous ces définitions de fréquence

Question : comment se comparent les fréquences du motif identifié ?

Résultat : SPMiner identifie les motifs qui apparaissent 10 à 100 fois plus fréquemment que les lignes de base

Les motifs identifiés par SPMiner ont également une fréquence de niveau de graphe plus élevée que les lignes de base.

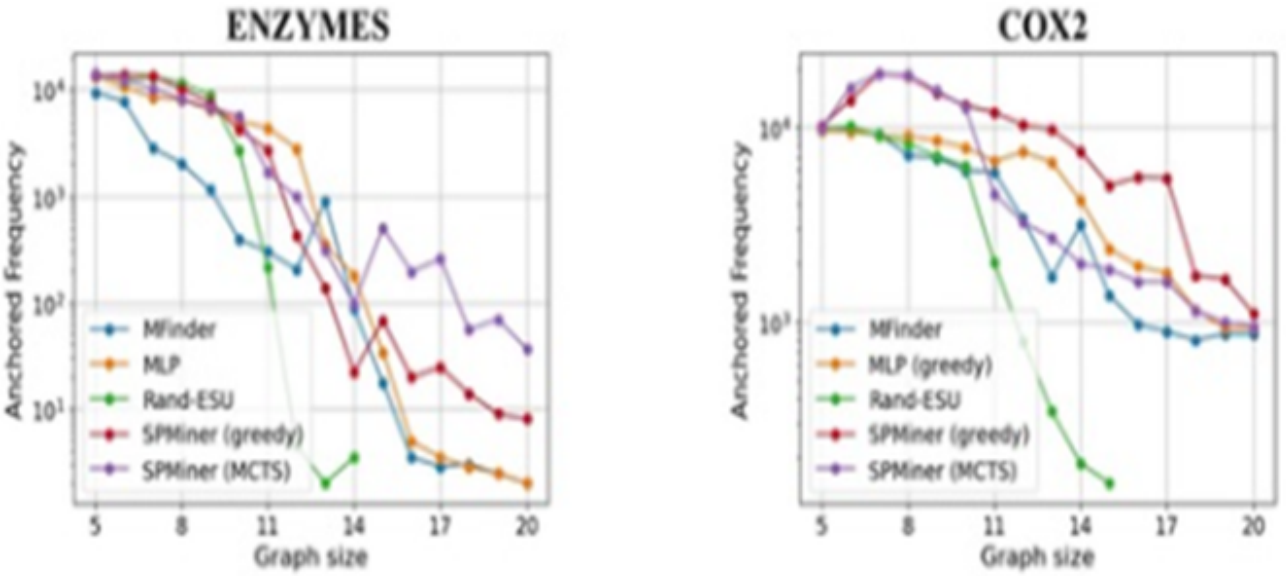


FIGURE 4.16: Expérience : grands motifs(1).

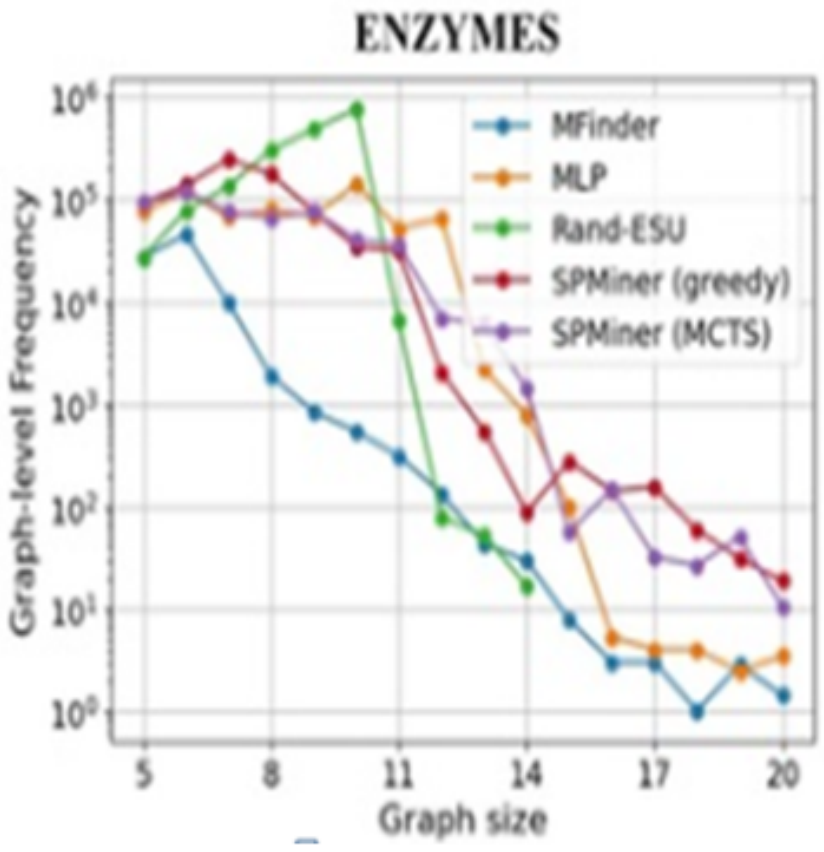


FIGURE 4.17: Expérience : grands motifs(2).

4.7 Conclusion :

Nous avons introduit le chapitre en donnant les diferent domaine d'utilisation de FSM avec GCNN.

Nous avons analisé notre broblème et détaillé l'approche qu'on va utiliser dans notre travail (SPminer)avec l'architecture GCNN.la partie d'experimentale contient deux experiences : de petit motif,et de large motif.en conclu que Les motifs identifiés par SPMiner ont également une fréquence de niveau de graphe plus élevée que les lignes de base. SPMiner est le premier cadre neuronal pour identifier un motif de graphique fréquent à l'aide de l'apprentissage de la représentation graphique. L'inclusion d'ordre et la recherche discrète guidée par les neurones permettent à SPMiner d'identifier les motifs 10 à 100 fois plus fréquemment que les méthodes existantes. Notre framework implémentation de cadre et de référence peut être étendue à un large éventail de tâches impliquant des sous-structures de graphes.

Conclusion générale

L'identification de sous-graphes fréquents (FSM) ou de motifs de réseau a été cruciale pour analyser et prédire les propriétés des réseaux du monde réel. Cependant, trouver de grands motifs courants reste un problème ouvert en raison de son sous-programme NP-dur de comptage de sous-graphes et de la croissance combinatoire du nombre de sous-graphes possibles avec leur taille. Nous présentons ici Subgraph Pattern Miner(SPMiner) une nouvelle approche neuronale pour trouver des sous-graphes fréquents dans un grand graphe cible. On intègre des réseaux neuronaux de graphes (GCNN), un espace d'incorporation d'ordres et une stratégie de recherche efficace pour identifier les modèles de sous-graphes de réseau qui apparaissent le plus fréquemment dans un jeu de données de graphes cible. On décompose d'abord le graphe cible en plusieurs sous-graphes qui se chevauchent, puis encode les sous-graphes en imbrications d'ordre. On utilise ensuite une marche monotone dans l'espace d'insertion de l'ordre pour identifier les motifs fréquents. en comparéson avec les approches existantes et les alternatives neuronales possibles,la méthode SPMiner est plus précise, plus rapide et plus évolutif. En outre, cette méthode peut également identifier de manière fiable les petits motifs fréquents, ce qui est bien

au-delà de la limite de taille des approches de dénombrement exact. Enfin, nous montrons que cette méthode peut trouver de grands motifs de qui apparaissent plus fréquemment que ceux trouvés par les méthodes approximatives couramment utilisées.

Bibliographie

[1] Jiawei Han. gSpan . Xifeng Yan. graph-based substructure pattern mining. eee int conf data min [internet] 1(d) :721–4, 2021.

[2] others Agrawal R, Srikant R. Fast algorithms for mining association rules. in : Proc 20th int conf very large data bases, vldb p. 487–991994, 1994.

[3] others Agrawal R, Srikant R. Fast algorithms for mining association rules. in : Proc 20th int conf very large data bases, vldb,p. 487–99., 1994.

[4] Laurent Alexandre. Posséder un picasso protégerait du cancer, 2018.

[5] Al Hasan M Bhuiyan MA. An iterative mapreduce based frequent subgraph mining algorithm. ieee trans knowl data eng ;27(3) :608–20., 2015.

[6] Berthold MR Borgelt C. Mining molecular fragments : Finding relevant substructures of molecules. in : Data mining, 2002 icdm 2003 proceedings 2002 ieee international conference on. p. 51–8., 2002.

[7] Abdelkader Ralem Ahmed Abdellatif Boubekeur. .implementation and evaluation of a frequent subgraph mining algorithm with hadoop an iterative mapreduced method,june, 2018.

[8] Xavier Bresson. Convolutional neural networks on graphs, 2017.

[9] Xavier Bresson. ***convolutional neural networks on graphs, 2017.

[10] H. Bunke. On a relation between graph edit distance and maximum common subgraph. pattern recognition letters, 1997.

[11] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph.pattern recognition letters, 1988.

[12] Thomas Bärecke. Isomorphisme inexact de graphes par optimisation évolutionnaire université pierre et marie curie - paris france, 2009.

[13] Stéphane Canu. Svm and kernel machine -linear and non-linear classification, 2014.

[14] Balachandran R. DB-Subdue Chakravarthy S, Beera R. Database approach to graph mining. in : Pacific-asia conference on knowledge discovery and data mining p. 341–50, 2004.

[15] Pradhan S. Db-fsg Chakravarthy S. An sql-based approach for frequent subgraph mining. in : International conference on database and expert systems applications p. 684–92, 2008.

[16] Salem S-Zaki MJ Chaoji V, Al Hasan M. An integrated, generic approach to pattern mining : data mining template library. data min knowl discov 17(3) :457–95, 2008.

[17] Galal G Maglothin R Cook DJ, Holder LB. . approaches to parallel graph-based knowledge discovery. j parallel distrib comput. 2001 ;61(3) :427–46.

[18] Hidehumi Sazoait et Alexander Waibel de Masahide Sugiyamat. Review of tdnn (time delay neural network) –architecture for speech recognition, 1991.

[19] Adit Deshpande. A beginner's guide to understanding convolutional neural networks, a beginner's guide to understanding convolutional neural networks part 2,, 20016.

[20] Adit Deshpande. A beginner's guide to understanding convolutional neural networks partie 1, partie 2 et partie 3, 2016.

[21] Vignesh Ramanathan Manohar Paluri et Laurens Van Der Maaten Dhruv Mahajan, Ross Girshick. Advancing state of the art image recognition with deep learning on hashtags, 2018.

[22] Vignesh Ramanathan Manohar Paluri et Laurens Van Der Maaten Dhruv Mahajan, Ross Girshick. Advancing state-of-the-art image recognition with deep learning on hashtags, 2018.

[23] Léon Bottou et al. Experiments with time delay networks and dynamic time warping for speaker independent isolated digits recognition, 1989.

[24] Emmanuel Viennet et Françoise Fogelman. Scene segmentation using multiresolution analysis and mlp, 1992.

[25] Sepp Hochreiter et Jürgen Schmidhuber. Long short-term memory, 1997.

[26] J.A. Bondy et U.S.R. Murty. Théorie des graphes traduit de l'anglais par f. hayet, 2008.

[27] Hopcroft J. E. et Wong J. K. Linear time algorithm for isomorphism of planar graphs, in sixth acm symposium on theory of computing, 1974.

[28] Olivier Ezratty. Les usages de l'intelligence artificielle, 2018.

[29] Chuntao Jiang FC and MZ. A survey of frequent subgraph mining algorithms. knowl eng rev., 2004.

[30] Justin Johnson et Serena Yeung Fei-Fei Li. Recurrent neural networks, 2018.

[31] M.-L. Fernandez and G. Valiente. A graph distance metric combining maximum common subgraph and minimum common supergraph. pattern recognition letters, 2001.

[32] Nicolas Gastineau. Partitionnement, recouvrement et colorabilité dans les graphes université de bourgogne – france, 2004.

[33] Antoine Gournay. Théorie des graphes institut de mathématiques, notes de cours- université de neuchâtel suisse septembre, 2013.

[34] Yin Y. Han J, Pei J. Mining frequent patterns without candidate generation. in : Acm sigmod record, 2000.

[35] Djoko S others Holder LB, Cook DJ. . substucture discovery in the subdue system. in : Kdd workshop. 1994. p. 169–80., 1994.

[36] Motoda H Inokuchi A, Washio T. An apriori-based algorithm for mining frequent substructures from graph data, 2000.

[37] Salim Jouili. Indexation de masses de documents graphiques : approches structurelles - université nancy 2, mars, 2011.

[38] KyungHyun Cho et Yoshua Bengio Empirical Junyoung Chung Chung, Caglar Gulcehre. evaluation of gated recurrent neural networks on sequence modeling, 2014.

[39] Nour El Islem KARABADJI. Fouille de graphes : Algorithme combinatoire et application, 2015.

[40] Magdalena Konkiewicz. Deep learning in a cloud. how to get started with google colab and why ?, 2020.

[41] Karypis G Kuramochi M. Frequent subgraph discovery. in : Data mining, 2001 icdm 2001, proceedings ieee international conference on. p. 313–20, 2001.

[42] Karypis G Kuramochi M. Frequent subgraph discovery. in : Data mining, 2001 icdm 2001, proceedings ieee international conference on. p. 313–20, 2001.

[43] Karypis G Kuramochi M. Grew-a scalable frequent subgraph discovery algorithm. in : Data mining, 2004 icdm'04 fourth ieee international conference on. p. 439–42., 2004.

[44] Laplace. La fiche wikipedia de la classification naïve bayésienne . voir également la présentation naïve bayes classifier.

[45] Justine Lebrun. Appariement inexact de graphes appliquée à la recherche d'image et d'objet 3 - université de cergy pontoise, 2011.

[46] Yann LeCun. la conférence de rob fergus au collège de france dans le cadre de la chaire, 2016.

[47] Yann LeCun. Le slide de droite est issu de la conférence à l'usi à paris inaugurale au collège de france en 2016, 2018.

[48] Chen H Ma J Zhang X Liu Y, Jiang X. Mapreduce-based pattern finding algorithm applied in motif detection for prescription compatibility network. in : International workshop on advanced parallel processing technologies. p. 341–55., 2009.

[49] Abdulkader A. M. Parallel algorithms for labelled graph matching, phd thesis, colorado school of mines, 1998.

[50] Urzova O Fischer I Philippsen M Meinl T, Wörlein M. The parmol package for frequent subgraph mining. electron commun easst, 2007.

[51] Li X Nguyen SN, Orlowska ME. Graph mining based on a data partitioning approach. in : Proceedings of the nineteenth conference on australasian database-volume 75. p. 31–7., 2008.

[52] Kok JN Nijssen S. A quickstart in frequent structure mining can make a difference. in : Proceedings of the tenth acm sigkdd international conference on knowledge discovery and data mining p. 647–52, 2004.

[53] Coatney MEfficient Parthasarathy S. discovery of common substructures in macromolecules. in : Data mining, 2002 icdm 2003 proceedings 2002 ieee international conference on. p. 362–9., 2002.

[54] Dreweke A Werth T. Parsemis Philippsen M, Worlein M. the parallel and sequential mining suite. available www2 inform uni-erlangen de/en/research/parsemis, 2011.

[55] Piyush Rai. Kernel methods and nonlinear classification, un cours de stanford, 2011.

[56] Jiaxuan You Jure Leskovec Rex Ying, Andrew Wang. Frequent subgraph mining by walking inorder embedding space, 2020.

[57] Michel Rigo. Théorie des graphes, université de liège faculté des sciences département de mathématiques - année académique, 2010.

[58] Vadim Smolyakov. Ensemble learning to improve machine learning results, 2017.

[59] James Somers. Is ai riding a one trick pony ?, 2017.

[60] Sunderraman R. Oo-FSG Srichandan B. An object-oriented approach to mine frequent subgraphs. in : Proceedings of the ninth australasian data mining conference-volume 121. p. 221–8., 2011.

[61] E.D. Taillard. Informatique orientation logiciels - Éléments de la théorie des graphes ecole d'ingénieur du canton de vaud, 2003.

[62] Jakob Uszkoreit. A novel neural network architecture for language understanding, 2017.

[63] M. Kraetzl W. D.Wallis, P. Shoubridge and D. Ray. Graph distances using graph union. pattern recognition letters, 2001.

[64] Parthasarathy S Wang C. . parallel algorithms for mining frequent structural motifs in scientific data. in : Proceedings of the 18th annual international conference on supercomputing. p. 31–40., 2004.

[65] Lee ML Sheng C. A Wang J, Hsu W. partition-based approach to graph mining. in : Data engineering, 2006 icde'06 proceedings of the 22nd international conference p. 74., 2006.

[66] Bai Y Wu B. An efficient distributed subgraph mining algorithm in extreme large graphs. in : International conference on artificial intelligence and computational intelligence.. p. 107–15., 2010.