Notes de cours : Analyse statistique de graphes

M2 Statististique / M2 Apprentissage et Algorithmes
Sorbonne Université

Catherine Matias

Avertissement : ce document contient certainement des erreurs et des imprécisions. N'hésitez pas à me les signaler.

Dernière mise à jour : 30 mars 2023

Table des matières

Préambule

Ce cours est axé sur l'analyse statistique d'observations organisées sous forme de graphes. L'accent est mis sur la pratique et la manipulation de fonctions déjà disponibles dans des librairies `R`. Gardez cependant en tête que vous aurez besoin de maîtriser les concepts sous-jacents pour bien comprendre et réussir ce cours.

Ce cours a bénéficié d'ajouts substantiels et améliorations par Tabea Rebafka. Les TPs ont été enrichis grâce à Fanny Villers. Merci à elles deux.

Voici quelques références bibliographiques (ces notes en font un usage immodéré) :

- Générales : Albert and Barabási (2002); Kolaczyk (2009); Kolaczyk and Csárdi (2014) ;
- Chapitre 5 sur le spectral clustering : von Luxburg (2007).

Chapitre 1

Introduction aux graphes

1.1 Les réseaux / les graphes

Ce cours porte sur l'analyse de données d'interactions, c'est-à-dire des données structurées sous la forme d'un graphe. Dans de nombreuses applications, un graphe permet de décrire les interactions paires à paires entre des individus, ou plus généralement, entre des entités. Les entités qui interagissent ou communiquent entre elles ainsi que le type d'interactions considérées varient beaucoup et dépendent du contexte d'application. Par exemple, sur un réseau social en ligne (e.g. Facebook, LinkedIn, ...) les entités sont les membres du site et on peut considérer comme une interaction le fait que deux membres déclarent un lien entre eux (e.g. amitié, relation professionnelle, ...). Dans le cas du trafic aérien, les entités sont les aéroports et une interaction est un vol direct d'un aéroport vers l'autre. L'ensemble des entités en interaction est appelé un **réseau**. Sa représentation mathématique est appelée un **graphe**, lequel est composé de nœuds (ou sommets, *i.e.* les entités considérées) et d'arêtes (ou liens, à savoir les interactions entre paires d'entités).

L'importance grandissante de l'analyse des graphes vient également du fait qu'il y a de plus en plus de contextes où les données de départ ne sont pas naturellement structurées en réseau, mais où il existe des représentations de ces données sous une telle forme. Le fait d'interpréter ces données comme des interactions permet l'application d'outils d'analyse des graphes pour étudier et comprendre ces données (de façon complémentaire à l'étude et l'analyse de leur structure plus naturelle). Par exemple, les réactions biochimiques du métabolisme d'un organisme ou d'une cellule peuvent être décrites comme des interactions entre des enzymes et des métabolites (connu sous le nom de réseau métabolique). Un exemple encore plus éloigné a priori du contexte des interactions consiste à considérer un tableau de données classiques,

i.e. des observations $x_i \in \mathbb{R}^p, 1 \leq i \leq n$, et à définir une notion de similarité entre chaque paire de vecteurs dans $\mathbb{R}^p$. On peut alors construire un graphe dans lequel chaque observation est un nœud du graphe et le poids d'une arête est la similarité entre les deux observations. De cette façon, on peut par exemple utiliser des méthodes de détection de communautés dans un graphe pour faire du clustering non supervisé des données d'origine $(x_i, 1 \leq i \leq n)$.

Pour finir cette introduction, **listons quelques concepts ou méthodes autour du mot-clé « graphe » et qui n'ont rien à voir avec le contenu de ce cours.** Cette liste n'est bien sûre pas exhaustive. 1) De nombreux algorithmes d'apprentissage statistique utilisent des structures de graphes pour analyser des données qui ne sont pas structurées sous forme de graphe (ni naturellement, ni après construction). C'est par exemple le cas des Graph Convolutional Networks. 2) On peut s'intéresser à des processus de diffusion d'information sur des graphes, par exemple quand on étudie des épidémies. Dans ce cadre on a une information additionnelle sur les nœuds du graphe, qui se propage à partir des arêtes. **Vous ne trouverez rien de tout ça dans ce cours.**

Dans ce premier chapitre, nous abordons différents exemples de graphes dans des champs d'application divers, et nous introduisons les notions élémentaires autour des graphes.

1.2 Exemples de graphes

Un **graphe** est composé d'un ensemble de **sommets** ou de **nœuds** qui sont reliés par des **arêtes** comme dans l'exemple de la Figure 1.1.

Voyons quelques exemples issus de domaines d'applications divers.

Historiquement, l'analyse des graphes s'est d'abord beaucoup développée en sciences sociales pour l'étude de réseaux sociaux, afin de mieux comprendre les interactions et dynamiques entre des personnes. Selon le contexte, un réseau social peut représenter des interactions physiques entre personnes, une base de données téléphoniques ou d'emails, ou un réseau social virtuel dont le plus grand est certainement Facebook (voir Figure 1.2).

Internet est un réseau physique avec des routeurs connectés par des câbles ethernet et/ou des liaisons wifi. Le world wide web est également un réseau, dans lequel les pages web sont les nœuds et les arêtes sont les hyperliens présents sur ces pages et qui pointent sur d'autres (Figure 1.3).

Dans les transports, il y a des réseaux de métro (Figure 1.4), bus ou train, des réseaux aériens ou routiers entre des villes, des réseaux avec des vélos en libre service

FIGURE 1.1 – Exemple de graphe composé de nœuds et d'arêtes.

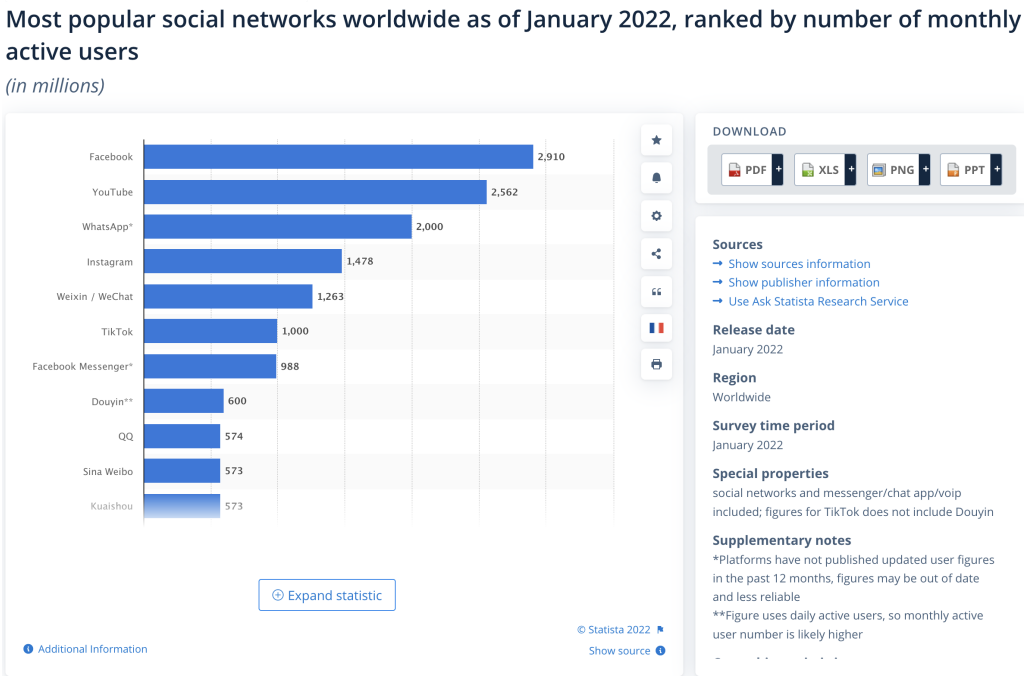

FIGURE 1.2 – Source : www.statista.com.

...

En écologie, on utilise des réseaux trophiques (ou *foodweb*) ou des réseaux plantes-pollinisateurs pour décrire des écosystèmes par les interactions entre espèces (Figure

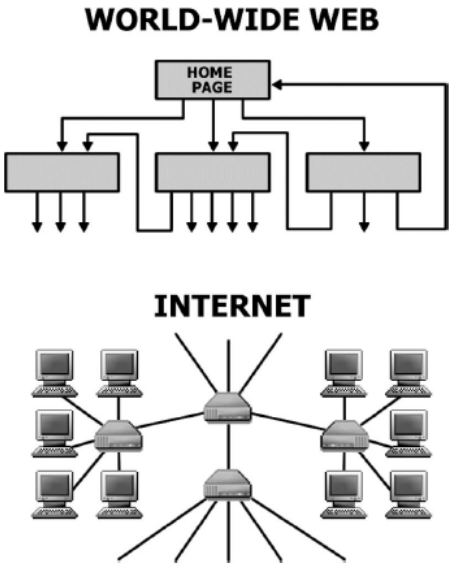

FIGURE 1.3 – Internet et WWW. Source : Albert and Barabási (2002).

1.5).

En médecine, on utilise des réseaux pour étudier les interactions entre gènes au sein d'une cellule, même si l'interaction entre gènes se fait par l'intermédiaire physique de protéines ou ARN (Figure 1.6). Cela permet de mieux comprendre la dynamique cellulaire au niveau du génome, et mieux comprendre le *système cellule* dans sa complexité. On utilise également des réseaux pour décrire en détail les activités du cerveau, comme par exemple l'influence des gènes sur différentes parties du cerveau (Figure 1.7).

Soyez toujours vigilants et sceptiques à toute visualisation de graphe, car elle peut être trompeuse. À titre d'exemple, la Figure 1.8 représente le même réseau de deux manières différentes et le résultat ne semble pas du tout le même. Ce problème est autant plus important que le graphe est grand. Nous voyons alors qu'en aucun cas l'analyse de graphe ne peut se contenter de la visualisation du graphe, même si celle-ci peut s'avérer intéressante. Nous en reparlerons dans la section 1.5

1.3 Vocabulaire des graphes

Un réseau est un ensemble d'entités en interaction tandis qu'un graphe est l'objet mathématique qui représente le réseau.

Un graphe $G = (V, E)$ est composé d'un ensemble $V = \{1, \ldots, n\}$ de nœuds

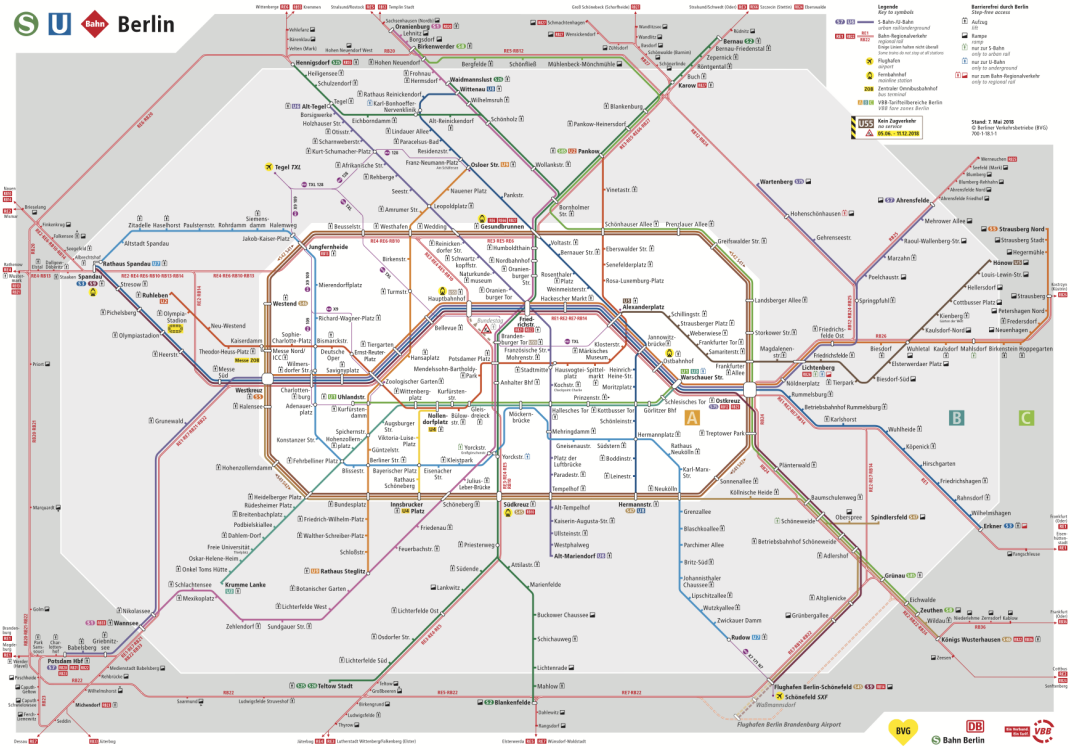

FIGURE 1.4 – Réseau de métro à Berlin.

(vertices en anglais) et d'un ensemble E d'arêtes (edges en anglais) avec $\{i, j\} \in E$ s'il y a une interaction entre les nœuds i et j dans le graphe. On dit (indifféremment) que l'arête $e = \{i, j\} \in E$ est issue de i (et de j).

Le nombre de nœuds n est l'**ordre** du graphe tandis que son nombre d'arêtes $|E|$ est appelé **taille** du graphe.

Un graphe est **dirigé** (ou orienté) lorsque ses arêtes le sont *i.e.* lorsque l'arête (i, j) est différente de l'arête (j, i). Il est **non dirigé** sinon. Une **boucle** (self-loop en anglais), est une arête du type $\{i, i\}$.

Les graphes peuvent être **binaires** : une arête est alors soit présente (1), soit absente (0) ; ou bien **valués** : les arêtes présentes sont alors munies d'une valeur (poids) tandis que les arêtes absentes sont encore codées par un 0. Noter qu'un graphe binaire est un cas particulier de graphe valué où toutes les arêtes présentes ont le poids 1.

Les graphes **simples** sont des graphes non dirigés, binaires, sans boucles, ni arêtes multiples (une arête (i, j) n'apparaît qu'au plus une fois. Dans la suite on ne considère jamais des graphes avec des arêtes multiples).

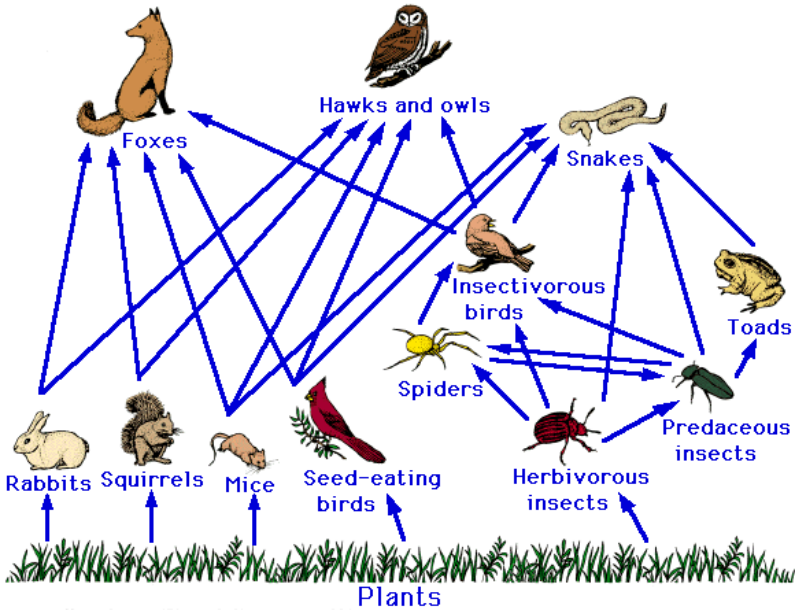

FIGURE 1.5 – Réseau trophique simplifié. Une arête indique de qui une espèce est la proie.

Un graphe simple sur n nœuds possède au plus $\binom{n}{2} = n(n-1)/2$ arêtes. Un graphe dirigé (sans arêtes multiples) en possède au plus $2\binom{n}{2} = n(n-1)$.

Les graphes **biparties** sont tels que $V = V_1 \cup V_2$ avec $V_1 \cap V_2 = \emptyset$ et les arêtes $e = \{u, v\} \in E$ sont telles que $u \in V_1, v \in V_2$. Tout ce qui suit se généralise facilement à ce cas.

Définition (Matrice d'adjacence)**.** Un graphe $G = (V, E)$ binaire sur un ensemble $V = \{1, \ldots, n\}$ de nœuds peut-être représenté par sa **matrice d'adjacence** (binaire) $A = (A_{ij})_{1 \leq i,j \leq n}$ où

$$A_{ij} = \begin{cases} 1 & \text{si } \{i, j\} \in E, \\ 0 & \text{sinon.} \end{cases}$$

Lorsque le graphe est non dirigé, la matrice A est symétrique. Si le graphe n'a pas de boucles on pose $A_{ii} = 0$ pour tout $i \in \{1, \ldots, n\}$. Si le graphe est valué, on peut considérer une matrice d'adjacence valuée

$$A_{ij} = \begin{cases} w_{ij} & \text{si l'arête est présente entre } i \text{ et } j \text{ et de poids } w_{ij}, \\ 0 & \text{sinon.} \end{cases}$$

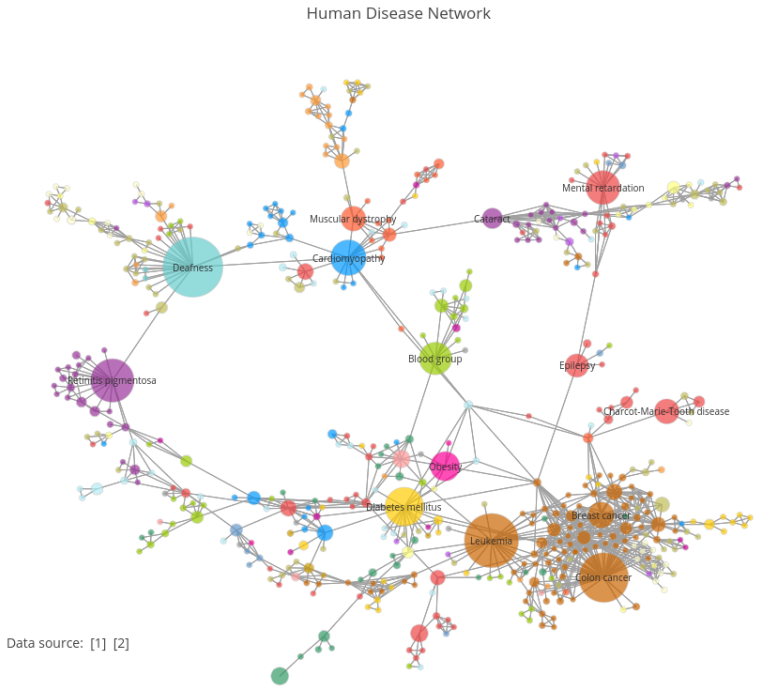

FIGURE 1.6 – Réseau de régulation des gènes.

Exercice. Considérons la matrice :

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Est-ce la matrice d'adjacence d'un graphe ? Si oui, quelles sont ses caractéristiques ?

1.4 Stockage informatique

Comment stocker efficacement un graphe en mémoire ? On peut bien sûr utiliser sa matrice d'adjacence, comme vu ci-dessus. On remarquera cependant que :

- Dans le cas d'un graphe non dirigé, cette matrice est symétrique donc de l'information redondante est stockée.
- Par ailleurs, si n est grand, stocker et manipuler une matrice de taille $n \times n$ peut s'avérer compliqué.

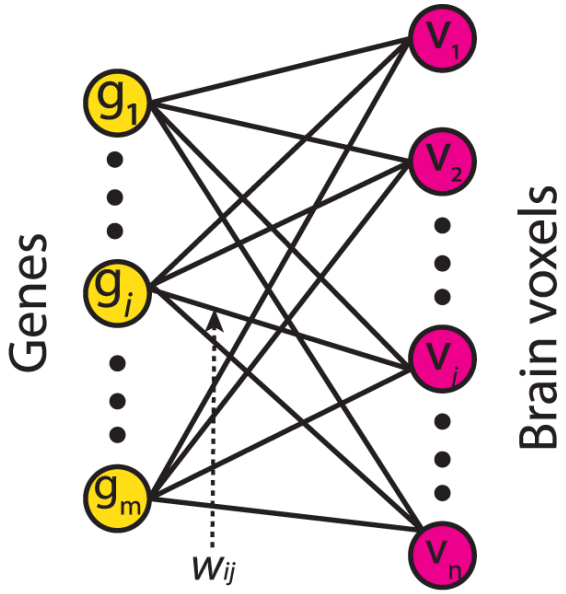

FIGURE 1.7 – Source : Ji et al. (2014).

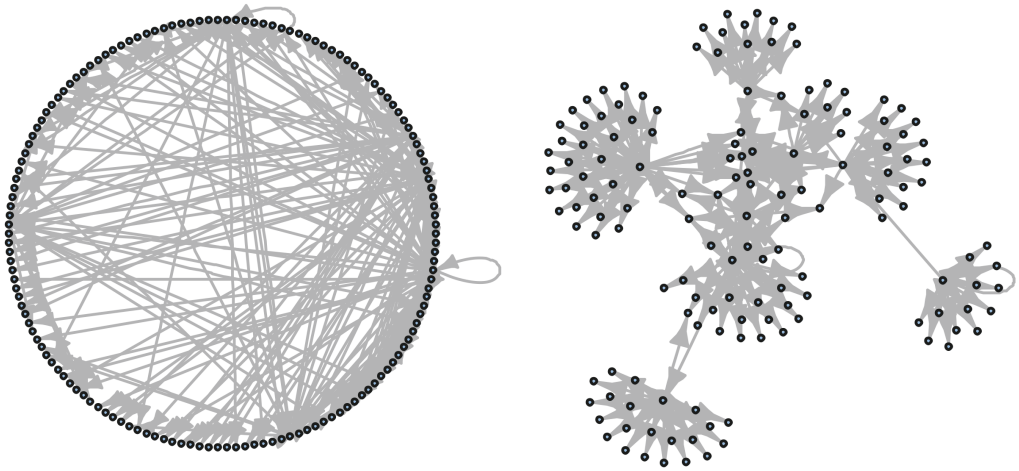

FIGURE 1.8 – Deux représentations différentes du même réseau de blogs (Kolaczyk and Csárdi, 2014).

- Enfin, si le graphe est très creux (*sparse* en anglais) c'est-à-dire s'il contient peu d'arêtes, on préfèrera utiliser des outils spécifiques pour manipuler les matrices creuses (matrices avec beaucoup de 0).

On peut également se contenter de stocker **la liste des arêtes du graphe**. C'est le codage de loin le plus efficace.

Il faut noter que dans ce cas, soit la liste des nœuds est déjà connue (*i.e.* la valeur de n est stockée par ailleurs) ; soit on considère que n est le nombre d'entités différentes apparaissant dans la liste des arêtes. Dans ce second cas, le graphe ne peut pas avoir de **nœuds isolés** (*i.e.* des nœuds qui n'ont aucun lien dans le graphe).

Exercice. Reconstruire le graphe avec $n = 5$ nœuds est dont la liste d'arêtes est : $\{1, 2\}, \{1, 3\}, \{2, 3\}, \{2, 4\}, \{3, 4\}$.

Pourtant, pour faire des calculs sur un graphe, il est souvent beaucoup plus simple d'utiliser la matrice d'adjacence (comme on le verra juste après). Dans la pratique, il faut parfois jongler entre ces deux représentations de graphe (matrice d'adjacence et liste d'arêtes).

1.5 Visualisation des graphes

En statistique, en général, la représentation graphique est un outil puissant d'exploration et d'analyse des données.

> *"L'excellence graphique est une présentation bien conçue des données d'intérêt [...], [elle] vise à communiquer des idées complexes avec clarté, précision et efficacité ; [elle] est ce qui donne au lecteur le plus grand nombre d'idées dans le temps le plus court, tout en utilisant le moins d'encre sur un minimum d'espace."* (Tufte, 2001).

La visualisation permet d'illustrer graphiquement les résultats, mais également de susciter des questionnements ou de valider des hypothèses. Concernant les graphes, la représentation graphique peut aider à comprendre la structure du graphe : y a-t-il des nœuds importants ? des nœuds très connectés ? comment se distribuent les arêtes ? ... Il est clair que plus le graphe est grand, plus la lisibilité du graphique est difficile.

A priori on a le droit d'arranger les nœuds dans l'espace comme on veut. En revanche, le résultat visuel peut changer énormément d'une représentation à l'autre comme nous l'avons déjà observé dans la Figure 1.8. On constate que **toute représentation est subjective** et peut être **trompeuse**. La visualisation de graphe est un problème compliqué, dû en particulier à une grande complexité intrinsèque des données relationnelles ou d'interaction.

Bahoken et al. (2013) proposent quelques règles générales pour la visualisation de graphes qui sont largement acceptées.

Trois principes de représentation de graphe :

- Les sommets les plus connectés sont placés au centre de la figure.
- Les sommets les moins connectés sont placés en périphérie de la figure.
- Il faut limiter, autant que possible, le chevauchement des liens.

(Remarque : Un graphe est dit **planaire** lorsque l'on peut le représenter dans le plan sans qu'aucune arête n'en croise une autre. Cette propriété n'est pas très intéressante pour ce qui nous concerne).

Dans la littérature on distingue différents positionnements des nœuds (Bahoken et al., 2013) : aléatoirement, en cercle, en étoile, en arbre (pour des graphes acycliques), ou encore selon différents algorithmes :

- fondé sur l'analyse des données (p. ex. analyse spectrale de la matrice d'adjacence) ;
- fondé sur la minimisation de l'énergie du système : système de forces entre des nœuds qui se repoussent et les arêtes assimilées à des ressorts qui tendent à rapprocher les nœuds voisins. Les résultats sont esthétiques, mais le temps de calcul peut être long, car l'algorithme est itératif (Fruchterman and Reingold, 1991; Kamada and Kawai, 1989).

La Figure 1.9 montre différentes représentations du réseau trophique de la Figure 1.5. Sous R, dans le package `igraph` l'option `layout` de la fonction `plot` permet de définir le type de représentation du graphe. Sous python, le package `NetworkX` propose différentes approches de visualisations.

Représentations visuelles avancées. Souvent, les données ne sont pas uniquement de type relationnelles, et on peut disposer de covariables sur les individus qui composent le graphe. Ces covariables peuvent être incluses dans la représentation, par exemple en jouant sur la couleur (covariable catégorielle) ou la taille (covariable quantitative) des nœuds .

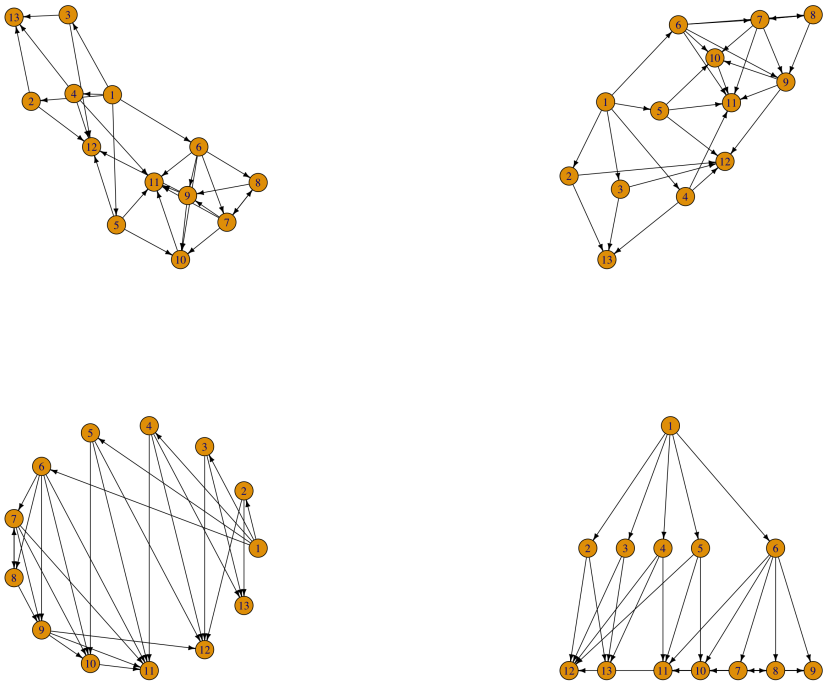

FIGURE 1.9 – Différentes visualisations du graphe trophique de la Figure 1.5. La dernière représentation est partielle et produit un *warning*.

Chapitre 2

Statistiques descriptives sur les graphes

Toute analyse statistique d'un nouveau graphe commence par l'exploration des données. Par des outils simples on tente de :

- comprendre et décrire la topologie du graphe ou la structure des interactions,
- identifier les sommets qui jouent un rôle important,
- comparer plusieurs graphes entre eux.

Les outils pour y parvenir peuvent être

- des indicateurs statistiques, pour capter les propriétés et caractéristiques du graphe,
- la réduction de dimension,
- des *graph embeddings*.

Dans ce chapitre, on présente des caractéristiques et indicateurs statistiques usuels pour des graphes. Pour aller plus loin, on pourra consulter des références bibliographiques (Kolaczyk, 2009; Kolaczyk and Csárdi, 2014; Luke, 2015).

Dans toute la suite et sauf mention contraire, $G = (V, E)$ est un graphe simple (donc en particulier binaire et non dirigé).

2.1 Densité

Nous avons déjà introduit l'ordre et la taille d'un graphe $G = (V, E)$ à la section 1.3. Ce sont clairement les caractéristiques les plus simples d'un graphe. En combinant les notions d'ordre et de taille d'un graphe, on définit sa densité.

Définition. La **densité** d'un graphe $G = (V, E)$ est définie par

$$\mathrm{den}(G) = \frac{|E|}{|V|(|V|-1)/2}.$$

La densité, comprise entre 0 et 1, traduit à quel point les nœuds sont connectés ou pas. Plus la densité est proche de 1, plus le graphe ressemble à un graphe complet où chaque nœud est connecté à tous les autres nœuds.

Définition. Le graphe **complet** K_n est le graphe non dirigé sur n sommets qui contient toutes les arêtes possibles entre ces sommets.

Ainsi, K_2 est une simple arête entre deux nœuds, K_3 est un triangle. Les sous-graphes complets d'un graphe sont plus communément appelés **cliques**. La densité du graphe K_n vaut 1.

Étant une quantité normalisée, la densité permet de comparer des graphes d'ordres différents entre eux.

On peut également définir une densité *locale* en considérant la densité d'un sous-graphe $H \subset G$. (Note : un sous-graphe $H = (V', E')$ de $G = (V, E)$ est un graphe tel que $V' \subset V$ et $E' \subset E$.)

2.2 Voisinages et degrés

Définition. Dans un graphe $G = (V, E)$, les **voisins** de $i \in V$ sont les nœuds $j \in V$ tels que $\{i, j\} \in E$. Ces nœuds forment le **voisinage** de i dans G et on note cet ensemble

$$\mathcal{V}(i) = \{j \in V ; \{i, j\} \in E\}.$$

Le **degré** d_i d'un nœud i dans le graphe G est le nombre de voisins de i dans G. C'est donc le cardinal $|\mathcal{V}(i)|$ du voisinage de i dans G. Le **degré moyen** d'un graphe est défini par

$$\bar{d} = \frac{1}{|V|} \sum_{i \in V} d_i.$$

Si G est un graphe dirigé, on peut définir le **degré sortant** d_i^{out} et le **degré entrant** d_i^{in} du nœud i. Dans un graphe dirigé, les **degrés moyens sortants et entrants sont nécessairement égaux** :

$$\bar{d}^{\mathrm{in}} := \frac{1}{|V|} \sum_{i \in V} d_i^{\mathrm{in}} = \bar{d}^{\mathrm{out}} := \frac{1}{|V|} \sum_{i \in V} d_i^{\mathrm{out}},$$

(parce que le nombre de flèches qui sortent de l'ensemble des nœuds est nécessairement égal au nombre de flèches qui entrent).

Un **hub** est (de façon informelle) un nœud du graphe de degré particulièrement élevé.

Proposition 2.1. *Dans un graphe $G = (V, E)$ on a $\sum_{i \in V} d_i = 2|E|$. En particulier, la somme des degrés d'un graphe est toujours paire.*

Proposition 2.2. *(Dans un graphe sans boucles), les degrés s'obtiennent à partir de la matrice d'adjacence via des sommes en ligne ou en colonne*

$$d_i = \sum_{j;j \neq i} A_{ij} \text{ (cas non dirigé)} ; \quad d_i^{out} = \sum_{j;j \neq i} A_{ij}; \quad d_i^{in} = \sum_{j;j \neq i} A_{ji}.$$

Remarque. La suite des degrés $(d_1, \ldots, d_n)$ d'un graphe est très contrainte. En fait Erdős and Gallai (1961) ont montré la propriété suivante (voir aussi Berge, 1976, Chapitre 6, Théorème 4). Quitte à ré-ordonner $(d_1, \ldots, d_n)$ de sorte que $d_1 \geq d_2 \geq \cdots \geq d_n$, une condition nécessaire et suffisante pour que $(d_1, \ldots, d_n)$ soit la réalisation de la séquence des degrés d'un graphe est que pour tout $1 \leq k \leq n-1$ on ait

$$\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \min(k, d_i).$$

Distribution des degrés. Notons tout d'abord que le degré moyen $\bar{d}$ n'est pas une variable très informative car dans les réseaux observés, les degrés des nœuds varient beaucoup. La distribution des degrés contient plus d'information que le degré moyen (voir par exemple la Figure 2.1). Il faut cependant garder en tête que des graphes très différents peuvent avoir la même distribution des degrés des nœuds (voire la même suite des degrés observés !). De la même façon, si deux graphes ont les mêmes degrés moyens entrants et sortants $\bar{d}^{\text{in}} = \bar{d}^{\text{out}}$, leurs suites de degrés entrants $(d_i^{\text{in}})_{1 \leq i \leq n}$ et sortants $(d_i^{\text{out}})_{1 \leq i \leq n}$ peuvent être très différentes, comme le montre la Figure 2.2.

Définition. Un graphe dont tous les nœuds ont le même degré d est un **graphe régulier** ou encore d-régulier.

Exemple de graphes réguliers. La grille $\mathbb{Z}^2$ est un graphe 4-régulier, le graphe complet K_n est un graphe $(n-1)$-régulier.

Les graphes réels sont rarement réguliers.

2.3 Connexité, distance et diamètre

Pour étudier le flux d'information ou le transport d'un objet sur un graphe, le voisinage direct d'un nœud n'est pas suffisant. On s'intéresse à l'existence de chemins

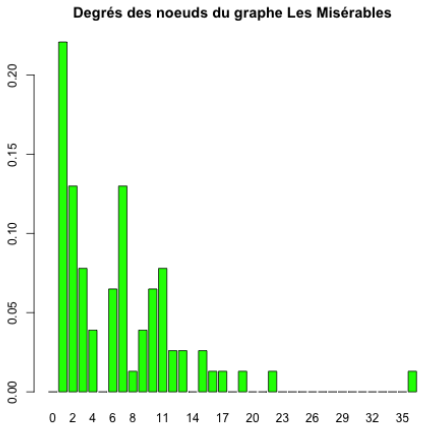

FIGURE 2.1 – Distribution des degrés du graphe Les Misérables.

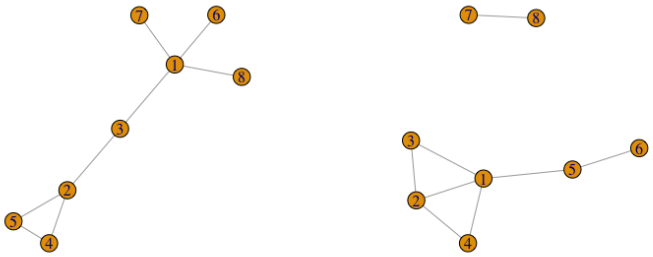

FIGURE 2.2 – Exemple de deux graphes qui ont la même suite de degrés.

qui relient deux nœuds quelconques dans un graphe, c'est la notion de connexité. Lorsqu'ils existent, la longueur de ces chemins indique le coût ou la durée pour aller d'un nœud vers l'autre, c'est la notion de distance. Enfin, le diamètre d'un graphe va caractériser les longueurs des chemins les plus longs.

Définition (Chemins, cycles, connexité)**.** Dans le graphe $G = (V, E)$ non orienté, un **chemin** entre $i, j \in V$ est une suite d'arêtes $e_1, \ldots, e_k \in E$ telle que :

- pour tout $1 \leq t \leq k-1$, les arêtes e_t et e_{t+1} partagent un nœud dans V ;
- e_1 est issue de i ;
- e_k est issue de j.

Un **cycle** est un chemin d'un nœud i à lui même (dans G). En particulier, une boucle est un cycle de longueur 1.

Une **composante connexe** (cc) de $G = (V, E)$ est un sous-ensemble $C = \{v_1, \ldots, v_k\} \subset V$ tel que pour tous $v_i, v_j \in C$, il existe un chemin dans G de v_i à v_j.

Une cc C est dite **maximale** si

- soit $C = V$,
- soit pour tout $v \in V \setminus C$, l'ensemble $V' = C \cup \{v\}$ n'est pas une cc.

Un graphe $G = (V, E)$ est dit **connexe** si V est une cc (*i.e.* pour tous $i, j \in V$, il existe un chemin de i à j dans G).

Un nœud qui n'a pas de voisins est dit **isolé** et il forme une composante connexe maximale.

Proposition 2.3. *Tout graphe peut être décomposé en un unique ensemble de composantes connexes maximales. Le nombre de composantes connexes maximales d'un graphe est supérieur ou égal à* $n - |E|$.

Preuve. On vérifie facilement que si $|E| = 0$ alors il y a n composantes connexes. De même si $|E| = 1$ on a exactement $n-1$ composantes connexes. Par induction sur le nombre d'arêtes : supposons que $G = (V, E)$ est un graphe avec c composantes connexes tel que $c \geq n - |E|$. On ajoute une arête à G pour fabriquer $G' = (V, E')$ et on note c' le nombre de composantes connexes de G'. Alors soit $c' = c$ (l'arête ajoutée relie deux nœuds qui sont déjà dans la même composante connexe), soit $c' = c - 1$ (l'arête ajoutée relie deux composantes connexes entre elles pour n'en créer plus qu'une). Dans le premier cas on a $c' = c \geq n - |E| \geq n - |E| - 1 = n - |E'|$. Dans le second cas, on a $c' = c - 1 \geq n - |E| - 1 = n - (|E| + 1) = n - |E'|$. La relation est toujours vérifiée. □

Si un graphe a plusieurs composantes connexes, on a tendance à commencer par les identifier, puis à analyser **séparément** chacune de ces composantes (mais ce n'est pas une règle absolue).

Dans un graphe orienté, on peut définir la notion de **chemin orienté** (ou dirigé) entre i et j. Il se peut alors qu'il existe un chemin orienté de i vers j sans chemin orienté de j vers i. De même il peut exister un chemin de i vers j sans chemin orienté de i vers j. La connexité d'un graphe dirigé est définie à partir des chemins non dirigés.

Composante connexe géante. Soit $G = (V, E)$ un graphe et C une composante connexe de ce graphe. La taille relative de C est définie par $|C|/|V|$ (nombre de nœuds de la composante sur nombre de nœuds total). Soit $(G_n)_{n\geq 1}$ une suite de graphes telle que G_n est un graphe à n nœuds et $(C_n)_{n\geq 1}$ suite croissante ($C_n \subset C_{n+1}$) telle que C_n est une composante connexe de G_n. Alors C_n est dite **géante** si sa taille relative $|C_n|/n$ ne tend pas vers 0 lorsque n devient grand.

Définition (Longueurs de chemins, diamètre)**.** La **longueur** d'un chemin $e_1, \ldots, e_k \in E$ dans $G = (V, E)$ est le nombre d'arêtes qui le composent (ici k).
Si deux nœuds i, j sont connectés dans G, alors la **distance** ℓ_{ij} entre i et j est la longueur d'un plus court chemin qui les relie dans le graphe. Si les deux nœuds ne sont pas connectés dans G alors $\ell_{ij} = +\infty$.
La **longueur moyenne** des chemins est définie par

$$\bar{\ell} = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j=1}^{n} \ell_{ij} = \frac{2}{n(n-1)} \sum_{i,j;i<j} \ell_{ij}.$$

Le **diamètre** d'un graphe G est la plus grande distance entre deux nœuds du graphe

$$\mathrm{diam}(G) = \max\{\ell_{ij}; i, j \in V\}.$$

Cette quantité n'est finie que pour les graphes connexes.

Le diamètre décrit le pire des cas pour envoyer de l'information ou une autre ressource dans le réseau. Un petit diamètre indique que de l'information circule rapidement dans le graphe. Le réseau est alors compact ou efficace.

Propriété petit monde (small-world property). Un réseau a la propriété **petit monde** si la distance moyenne $\bar{\ell}$ est proportionnelle à $\log(|V|)$.

Cette propriété traduit le fait que dans certains réseaux même très grands, la distance entre deux nœuds pris au hasard reste relativement petite. L'origine du terme est due à Stanley Milgram (Milgram, 1967) qui en 1967 étudie un réseau social de connaissances entre personnes aux USA et conclu (de façon empirique) au phénomène des « six degrees of separation » à savoir que dans ce réseau on a $\bar{\ell} \approx 6$.

Plus récemment, à l'ère de Facebook, Bhagat et al. (2016) estiment que parmi les membres du réseau social Facebook du monde entier, on est passés à $\bar{\ell} \approx 3,5$ (*three and a half degrees of separation*).

D'autres réseaux exhibent cette propriété de petit monde : le réseau des acteurs Holywoodiens reliés par leur co-apparition dans un film est caractérisé par $\bar{\ell} = 3$.

2.4 Clustering, transitivité

Le clustering est la propriété qui traduit que le graphe est composé de groupes de sommets très connectés entre eux. Le réseau est alors organisé en cliques ou quasi-cliques. Pour rappel, une clique est un sous-graphe complet, c'est-à-dire que chaque nœud est connecté à tous les autres.

Dans les réseaux sociaux avec une grande cohésion sociale, on observe le principe : *Les amis de mes amis sont mes amis.* Pour quantifier le clustering, on peut alors mesurer à quel point les voisins d'un nœud i sont connectés entre eux.

Comme pour les degrés, il s'agit de regarder l'environnement local, en incluant cette fois les voisins à distance 2.

Rappelons que pour chaque sous-graphe de G, on peut définir la densité (dite locale) de ce sous-graphe. Un cas intéressant est obtenu pour H_i : **le sous-graphe induit construit à partir des voisins d'un nœud** $i \in V$, *i.e.* le sous-graphe $H_i = (\mathcal{V}(i), E_i)$ où $\mathcal{V}(i)$ est l'ensemble des voisins de i et E_i l'ensemble des arêtes $\{j, k\} \in E$ telles que $j, k \in \mathcal{V}(i)$. C'est à partir de cette notion qu'est défini le coefficient de clustering.

Définition (Coefficient de clustering local, moyen et global)**.** On note d_i le degré du nœud i et $|E_i|$ le nombre d'arêtes qui connectent les voisins de i entre eux. On définit le coefficient C_i de clustering du nœud i par

$$C_i = \begin{cases} \frac{2|E_i|}{d_i(d_i-1)} & \text{si } d_i \geq 2, \\ 0 & \text{sinon.} \end{cases}$$

C'est la densité locale du sous-graphe H_i et donc à valeurs dans $[0, 1]$. De même, on définit le coefficient de clustering moyen $\bar{C}$ par

$$\bar{C} = \frac{1}{|V|} \sum_{i \in V} C_i.$$

Ainsi, on a toujours

$$0 \leq \bar{C} \leq 1,$$

avec $\bar{C} = 0$ ssi chaque nœud est tel qu'aucun de ses voisins ne sont reliés par une arête (le graphe ne contient *aucun* triangle, mais peut contenir des cycles de longueur supérieure ou égale à 4) et $\bar{C} = 1$ ssi deux arêtes adjacentes dans le graphe forment toujours un triangle.

Le coefficient de clustering moyen est relié à la densité des triangles parmi les paires de relations dans le graphe. Les triangles traduisent les relations de *transitivité* : importante en particulier dans les réseaux sociaux. On peut aussi mesurer directement cette densité des triangles par le coefficient de transitivité.

Définition (Transitivité)**.** Pour chaque nœud $i \in V$, on note

- λ_i le nombre de triangles (*i.e.* cycle de longueur 3) dont i fait partie dans G, *i.e.* le nombre de paires $\{j, k\} \in \mathcal{V}(i)^2$ telles que l'arête $e = \{j, k\} \in E$
- τ_i le nombre de chemins de longueur 2 qui contiennent i.

On définit le coefficient de transitivité par

$$T = \frac{\sum_{i \in V} \lambda_i}{\sum_{i \in V} \tau_i} = \frac{\sharp \text{ triangles}}{\sharp \text{ triplets de nœuds connectés}}.$$

Remarques.

- Il arrive que le coefficient de transitivité soit appelé coefficient de clustering.
- Formellement, T n'est pas défini si le graphe n'est composé que d'arêtes isolées (cas pas très intéressant).
- Dans la définition précédente, au numérateur, chaque triangle du graphe compte 3 fois. Mais au dénominateur, chaque chemin de longueur 2 compte aussi 3 fois. Donc au final, T est simplement la fréquence des triangles.
- Pour un arbre (*i.e.* un graphe acyclique), on a $\bar{C} = 0 = T$, mais ce n'est pas une condition suffisante. En effet, un graphe peut avoir des valeurs nulles de $\bar{C}$ ou T et contenir des cycles (donc ne pas être un arbre).

Exercice.

1. Déterminer le coefficient de clustering C_i de chaque nœud dans la Figure 2.3 à gauche, ainsi que le coefficient de clustering moyen et le coefficient de transitivité.
2. Pour le graphe de la Figure 2.3, à droite, étudier la limite du coefficient de clustering moyen et du coefficient de transitivité lorsque n tend vers l'infini.
3. Écrire un script `R` qui permet de vérifier vos résultats (vous pouvez utiliser des fonctions de `igraph`).

Remarque. Dans le graphe de l'exercice 2 ci-dessus, le coefficient de clustering moyen tend vers 1 car tous les nœuds sauf deux ont un coefficient de clustering égale à 1. Pourtant, le coefficient de transitivité tend vers 0 car la proportion de triangles sur les triplets connectés tend vers 0. On voit donc que $\bar{C}$, qui est la moyenne des coefficients locaux C_i ne mesure pas du tout la même chose que T.

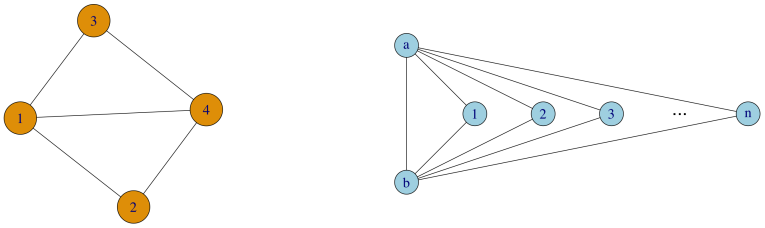

FIGURE 2.3 – À gauche : Graphe pour la question 1 de l'exercice. À droite : Graphe pour la question 2 de l'exercice.

2.5 Centralité des nœuds

Les nœuds d'un graphe ne jouent pas tous le même rôle ou n'ont pas tous la même importance dans le graphe. Différents indicateurs définis dans cette section permettent d'identifier des nœuds 'centraux' d'un réseau, qui sont les plus influents ou des éléments clés de ce réseau.

Définition. L'indicateur de centralité le plus simple est la **centralité de degré** C_D ou **degree centrality**, aussi appelée centralité de prestige, qui coïncide avec le degré du nœud i

$$C_D(i) = d_i.$$

Intuitivement, un nœud très connecté aux autres est très important dans un graphe et on l'appelle un **hub**.

Au lieu d'analyser les connexions directes, on peut analyser comment le nœud est relié à tous les autres sommets.

Définition. Dans un graphe connexe, la **centralité de proximité** C_P ou **closeness centrality** du nœud i est définie par

$$C_P(i) = \left(\sum_{j \in V} \ell_{ij} \right)^{-1},$$

où ℓ_{ij} est la distance (*i.e.* le plus court chemin) de i à j.

La centralité $C_P(i)$ d'un nœud i est élevée quand sa distance à tous les autres nœuds est faible. Le nœud le plus central est « proche » de toute l'information dans le graphe.

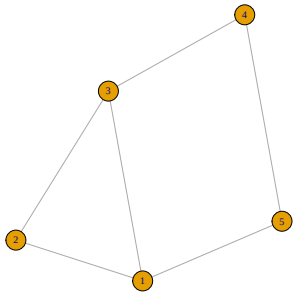

FIGURE 2.4 – Calculez la betweenness centrality du nœud 2.

On peut également mesurer à quel point un nœud est important pour connecter deux autres nœuds dans le graphe.

Définition. Dans un graphe connexe, la **centralité d'intermédiarité** C_B ou **betweenness centrality** du nœud i est définie par

$$C_B(i) = \sum_{j,k:j\neq k\neq i} \frac{g_{jk}(i)}{g_{jk}},$$

où g_{jk} est le nombre de plus courts chemins de j à k, et $g_{jk}(i)$ le nombre de plus courts chemins de j à k qui en plus passent par i.

La centralité $C_B(i)$ d'un nœud est élevée s'il est un point de passage sur un grand nombre de chemins les plus courts entre deux autres nœuds. Autrement dit, la plupart de la communication dans le graphe passe par lui.

Exercice.
- Regardez le graphe de la Figure 2.4. Que vaut la betweenness centrality du nœud 2 ?
- Prenez une étoile, que vaut la betweenness centrality du nœud au centre de l'étoile ?

Une autre façon de déterminer l'importance d'un nœud est de regarder à quel point sa suppression modifie les caractéristiques du graphe.

Définition. Un **point d'articulation** ou **cutpoint** est un sommet qui, si on le supprime du graphe, augmente le nombre de composantes connexes.

Retirer un point d'articulation d'un graphe implique que certains nœuds ne peuvent plus communiquer entre eux. Aux points d'articulation avec une centralité d'intermédiarité élevée, le graphe est très vulnérable en termes de communication.

De la même façon on peut analyser l'importance des arêtes d'un graphe.

Définition. Un **pont** ou **bridge est une arête** qui, si on la retire du graphe, augmente le nombre de composantes connexes.

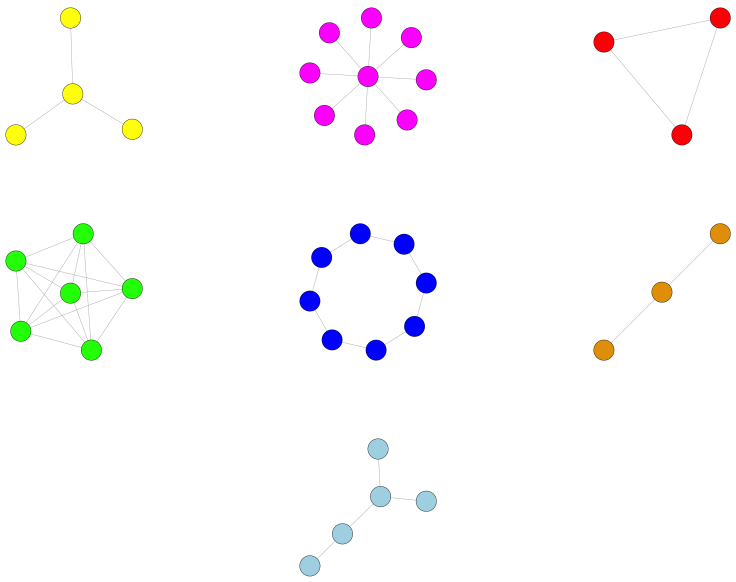

FIGURE 2.5 – Quelques exemples de motifs : étoiles (k-stars avec $k = 3$ et $k = 8$), cliques (K_3 c'est à dire triangle et K_6), cycle de longueur 8, ...

2.6 Motifs

La topologie d'un graphe peut être captée par les occurrences d'un motif donné. Un motif est un sous-graphe de structure particulière, comme des triangles, cliques, étoiles, cycles, arbres,.... La Figure 2.5 montre quelques exemple.

Définition. Soient $G = (V, E)$ et $G' = (V', E')$ deux graphes. On dit que

- G' est un sous-graphe de G (et on note $G' \subset G$) si $V' \subset V$ et $E' \subset E$.
- G' est un **sous-graphe induit** de G lorsque $G' \subset G$ et E' contient toutes les arêtes $\{i, j\} \in E$ telles que $i, j \in V'$.
- G et G' sont **isomorphes** si il existe une bijection $\phi : V \to V'$ telle que $\{i, j\} \in E$ ssi $\{\phi(i), \phi(j)\} \in E'$.

Un **motif** m d'un graphe G est un sous-graphe induit de G. Chercher les occurrences de m dans G c'est chercher tous les sous-graphes induits de G qui sont isomorphes à m. Souvent, on ne s'intéresse pas au nombre absolu d'un motif dans un graphe, mais à sa fréquence, *i.e.* à la proportion relative de ce motif dans le graphe, comparé au nombre de motifs dans un graphe complet.

On peut ensuite chercher à caractériser le nombre d'occurrences d'un motif observé par rapport au nombre attendu sous une hypothèse nulle H_0 (le modèle nul

est obtenu par simulations ou par un modèle défini analytiquement). On va ainsi construire des tests statistiques basés sur la fréquence d'un motif, par rapport à un modèle de référence. Et répondre ainsi à la question : la fréquence d'occurrences de ce motif est-elle trop faible ou trop grande dans ce graphe ? (par rapport au modèle attendu).

Chapitre 3

Modèles simples de graphes aléatoires

Le chapitre précédent a présenté des caractéristiques et indicateurs empiriques pour des graphes. Ils sont très utiles pour une première exploration des données, mais une analyse statistique ne peut pas s'arrêter là. Ce chapitre présente quelques premiers modèles de graphes aléatoires : d'une part, le modèle célèbre d'Erdős-Rényi, d'autre part, des modèles définis à partir d'une suite de degrés des nœuds.

Un modèle de graphes aléatoires est une collection (finie ou dénombrable) $\mathcal{G}$ de graphes et une loi de probabilité $\mathbb{P}$ sur cette collection $\mathcal{G}$.

3.1 Le modèle d'Erdős-Rényi

3.1.1 Définition du modèle

Le modèle de graphe aléatoire le plus simple est le modèle introduit à la fin des années 50 par Erdős et Rényi (Erdős and Rényi, 1959). Il s'exprime sous deux variantes différentes : les modèles $\mathcal{G}(n, M)$ en $\mathcal{G}(n, p)$.

Le modèle $\mathcal{G}(n, M)$ est la collection de tous les graphes simples (binaires, non dirigés, sans boucles ni arêtes multiples) d'ordre n et de taille M, munie de la loi uniforme $\mathbb{P}$ sur cette collection. Ainsi, la collection $\mathcal{G}(n, M)$ contient $\binom{N}{M}$ graphes différents, où $N = n(n-1)/2$ et la probabilité de chacun d'eux est $1/\binom{N}{M}$.

Une variante plus commune consiste à considérer la collection $\mathcal{G}(n, p)$ de graphes simples générés selon un modèle à deux paramètres : n le nombre de nœuds du graphe et $p \in (0, 1)$ la probabilité de connection de deux nœuds pris au hasard. C'est un graphe dont toutes les arêtes $A_{ij}, 1 \leq i < j \leq n$ sont des variables i.i.d. de loi de Bernoulli $\mathcal{B}(p)$.

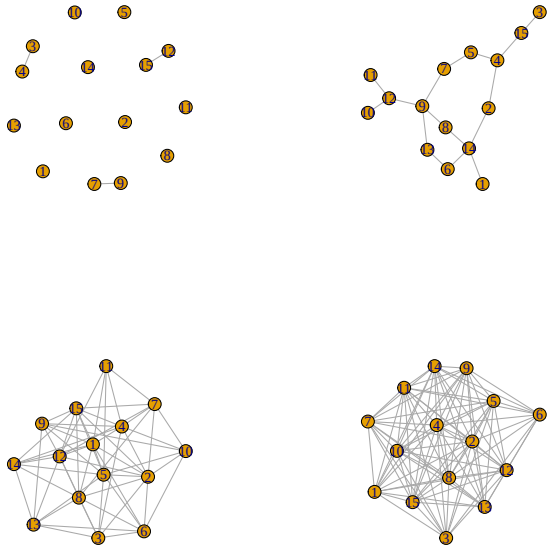

FIGURE 3.1 – Exemples de graphes du type Erdős-Rényi $\mathcal{G}(n,p)$ avec $n = 15$ et $p \in \{0.05, 0.2, 0.6, 0.9\}$.

Un réseau observé peut être considéré comme une réalisation de la variable aléatoire notée $G(n,p)$ de loi $\mathcal{G}(n,p)$ (ou de la même façon, une variable aléatoire notée $G(n,M)$ de loi $\mathcal{G}(n,M)$).

La figure 3.1 montre des exemples d'un graphe généré sous le modèle d'Erdős-Rényi $\mathcal{G}(n,p)$ pour différentes valeurs du paramètre $p \in [0,1]$.

Exercice. Soit n et p fixés. Considérons un graphe généré sous le modèle $\mathcal{G}(n,p)$.

1. Donner le nombre moyen d'arêtes dans ce graphe.
2. Soit $M \geq 0$. Quelle est la probabilité que le graphe est de taille M ?
3. Donner la loi de la variable aléatoire D_i qui désigne le degré du nœud i dans le modèle $\mathcal{G}(n,p)$. En déduire l'espérance du degré D_i.
4. Etudier la convergence de $\bar{D}_n/(n-1)$ lorsque $n \to \infty$, où $\bar{D}_n$ désigne le degré moyen du graphe sous $\mathcal{G}(n,p)$.
5. Donner une approximation de la loi de D_i lorsque n est grand.

Lorsqu'on considère une suite de graphes G_n générés sous la suite de modèles $\mathcal{G}(n,p_n)$, on suppose typiquement que $(p_n)_n$ est décroissante, sinon les graphes associés sont trop denses pour modéliser des grands graphes réels. Pour ces suites de graphes, on peut étudier le phénomène de transition de phase , *i.e.* on veut savoir à partir de quelle suite $(p_n)_n$ certains motifs apparaissent dans le graphe lorsque n tend vers l'infini.

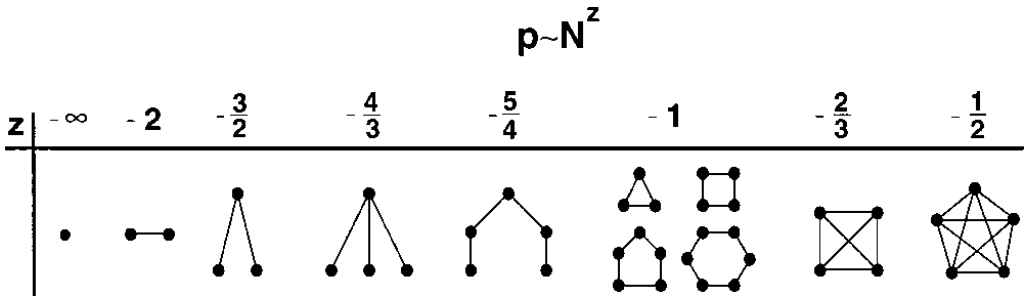

FIGURE 3.2 – Transition de phase dans $\mathcal{G}(N, p_N)$. Figure tirée de Albert and Barabási (2002).

Exercice. Soit m un motif avec k sommets et ℓ arêtes. Notons X_n le nombre d'occurrences du motif m (non induit) dans un graphe sous le modèle $\mathcal{G}(n, p_n)$.

1. Montrer que
$$\mathbb{E}[X_n] = \binom{n}{k} \frac{k!}{a} p_n^l \approx \frac{n^k p_n^l}{a},$$
où a est le nombre de permutations de k nœuds du motif m tel que le graphe obtenu par permutation est isomorphe à lui même (*i.e.* c'est le même motif).
2. Montrer que si $p_n = o(n^{-k/l})$, alors
$$\lim_{n\to\infty} \mathbb{P}(X_n = 0) = 1.$$

D'après Bollobás (1985) on a aussi : si $p_n = c_n n^{-k/l}$ avec $c_n \to \infty$, alors

$$\lim_{n\to\infty} \mathbb{P}(X_n > 0) = 1.$$

La figure 3.2 illustre ce résultat. En particulier, si $p_n = n^z$ pour $z < 0$, alors avec $z < -2$, un graphe généré sous $\mathcal{G}(n, p)$ est essentiellement composé de nœuds isolés. Pour $-2 < z < -\frac{3}{2}$ des arêtes apparaissent, ensuite des arbres. À partir de $z > -1$ on observe des cycles, et il faut $z > -\frac{2}{3}$ pour voir apparaître les premières cliques d'ordre 4.

Simulation de grands graphes sous le modèle $\mathcal{G}(n, p)$. En principe, il suffit de générer $n(n-1)/2$ variables aléatoires de Bernoulli de paramètre p. Lorsque n est grand et $p = p_n$ de l'ordre de $1/n$, cette procédure est très inefficace : l'espérance du degré d'un nœud est finie et donc la plupart des variables valent 0.

Dans ce cas, il est plus efficace de simuler d'abord le nombre M d'arêtes présentes dans le graphe selon une loi binomiale $Bin(n(n-1)/2, p)$, et ensuite tirer les positions

des arêtes, i.e. tirer sans remise M positions parmi les $n(n-1)/2$ arêtes possibles. La complexité de calcul est alors en $O(n + |E|)$ au lieu de $O(n^2)$ (Voir Kolaczyk (2009), section 6.2.3 pour plus de détails).

3.1.2 Propriétés des graphes $G(n, p)$

Dans le cas de $\mathcal{G}(n, p)$, les variables A_{ij} sont i.i.d. de loi $\mathcal{B}(p)$. L'analyse des propriétés du graphe est facilitée par cette constatation.

Distribution des degrés. Rappelons que pour un graphe simple, le degré D_i du nœud i pour $i = 1, \ldots, n$ et le degré moyen $\bar{D}$ du graphe vérifient la relation suivante

$$D_i = \sum_{j \neq i} A_{ij}, \quad \text{et } \bar{D} = \frac{1}{n} \sum_{i=1}^{n} D_i = \frac{1}{n} \sum_{i=1}^{n} \sum_{j \neq i} A_{ij} = \frac{2|E|}{n}.$$

Les degrés $\{D_i\}_{i=1,\ldots,n}$ des nœuds d'un graphe sont des variables aléatoires, non indépendantes en général. On s'intéresse ici à la distribution marginale de ces variables.

Proposition 3.1. *Le degré D_i du nœud i du graphe aléatoire $G(n, p)$ vérifie*

$$D_i \sim \mathcal{B}(n-1, p).$$

Par la loi des grands nombres, la variable $\bar{D}/(n-1)$ converge vers $\mathbb{E}(A_{ij}) = p$. En particulier, l'espérance du degré d'un nœud vérifie $\mathbb{E}(D_i) = (n-1)p = pn(1 + o(1))$ (lorsque n grand, p petit).

Lorsque $n \to +\infty$ et $p \to 0$ avec $np \to \lambda > 0$ alors la loi $\mathcal{B}(n-1, p)$ est approchée par une loi de Poisson $\mathcal{P}(\lambda)$.

La loi binomiale comme la loi de Poisson sont des lois à queues légères impliquant qu'il y a très peu de valeurs extrêmes. Or, les réseaux réels sont typiquement très hétérogènes ; ils contiennent un petit nombre de nœuds avec des degrés très forts, les *hubs*. Ceci est une première indication que le modèle d'Erdős-Rényi s'ajuste mal sur des graphes issus d'applications pratiques.

Coefficient de clustering. Dans $\mathcal{G}(n, p)$, puisque toutes les arêtes sont indépendantes, la probabilité que deux voisins d'un nœud i soient connectés vaut p. Donc en moyenne, $\mathbb{E}(2|E_i| \,|D_i) = pD_i(D_i - 1)$ ce qui donne une valeur moyenne du coefficient local de clustering $\mathbb{E}(C_i) = p$ et $\mathbb{E}(\bar{C}) = p \simeq \mathbb{E}(D_i)/n \simeq \bar{d}/n$. En conséquence, dans $G(n, p)$, le rapport $\bar{c}/\bar{d}$ doit être de l'ordre de $1/n$. Dans les graphes réels, on observe plutôt un rapport constant.

Distances moyennes. Pour un graphe $G(n,p)$, on peut montrer que la valeur typique de la distance ℓ_{ij} est de l'ordre de $\log(n)$, donc les graphes $G(n,p)$ sont des graphes petit monde.

Mauvaise adéquation. Le modèle d'Erdős-Rényi est un modèle mathématique très étudié, avec de nombreuses propriétés très intéressantes, mais il s'ajuste mal aux réseaux observés. Son principal défaut est qu'il produit des graphes homogènes, alors que la vaste majorité des graphes réels sont relativement hétérogènes.

En revanche, l'idée de modéliser les arêtes par des variables aléatoires permet de définir facilement d'autres modèles de graphes plus appropriés pour l'analyse de graphes réels, cf. Chapitre 6.

3.2 Modèles sur la loi marginale des degrés ou modèles de configuration

Beaucoup de graphes réels ont une distribution des degrés des nœuds qui s'ajuste correctement sur une loi de puissance, *i.e.*

$$f_{D_i}(k) := \mathbb{P}(D_i = k) = \frac{c}{k^\gamma},$$

où c est une constante de normalisation et $\gamma > 0$ est l'exposant de la loi puissance.

Dans les années 2000, beaucoup de publications se sont concentrées sur ce phénomène de loi de puissance de la distribution des degrés, caractérisant par exemple l'exposant de la loi de puissance de réseaux observés. Le mauvais ajustement du modèle $G(n,p)$ sur la loi des degrés a donné lieu à de nouveaux modèles, fondés uniquement sur cette distribution des degrés.

On peut donc vouloir définir des modèles de graphes aléatoires en utilisant uniquement la distribution des degrés des nœuds. Ainsi, on peut considérer les modèles suivants

1. Loi de puissance des degrés : On considère des graphes aléatoires sur n nœuds tels que les variables aléatoires $D_1, \ldots, D_n$ sont i.i.d selon une loi de puissance (pour un certain $\gamma > 0$). On dit aussi que ces réseaux sont **invariants d'échelle** ou *scale free* (Lhomme, 2012).
2. Modèle à degrés fixés : Soient $\underline{d} = (d_1, \ldots, d_n)$ une suite (possible) de degrés de nœuds et $FD(\underline{d})$ la collection de tous les graphes sur n nœuds qui possèdent exactement la suite des degrés $\underline{d}$, munis de la probabilité uniforme.

3. Modèle à degrés variables : Soient $\underline{d} = (d_1, \ldots, d_n)$ une suite (possible) de degrés de nœuds et $RD(\underline{d})$ le modèle de graphe aléatoire sur n nœuds tel que toutes les arêtes A_{ij} sont indépendantes, de loi $\mathcal{B}(p_{ij})$ avec $p_{ij} = d_i d_j / C$ où C constante positive telle que $0 \leq p_{ij} \leq 1$ (par exemple $C = \max_{i \neq j} d_i d_j$).

Dans le modèle $FD(\underline{d})$, tous les graphes ont exactement la suite de degrés $\underline{d}$. Dans le modèle de loi de puissance, on commence par tirer une suite $\underline{d}$ de degrés selon cette loi de puissance, puis on considère un graphe qui a cette suite de degrés fixés. Enfin dans le modèle $RD(\underline{d})$, les degrés sont seulement approchés par la suite $\underline{d}$. En effet dans ce cas

$$\mathbb{E}(D_i) = \sum_{j \neq i} \mathbb{E}(A_{ij}) = \sum_{j \neq i} p_{ij} = \frac{d_i}{C} \sum_{j \neq i} d_j = \frac{d_i(2|E| - d_i)}{C}.$$

En prenant d_i pas trop grand et $C \simeq 2|E|$ on obtient $\mathbb{E}(D_i) \simeq d_i$.

Remarques.
- Le modèle de loi de puissance des degrés n'est pas constructif ni simplement simulable : si on tire une suite de D_i comme indiqué, on a peu de chances que la réalisation satisfasse les conditions du théorème d'Erdős-Gallai et donc soit réalisable en tant que suite de degrés d'un graphe.
- Par contre, lorsque l'on trace un histogramme des d_i observés et qu'on ajuste une loi de puissance sur cette distribution empirique, on est bien en train de travailler sous ce modèle !
- La simulation de graphes dans le modèle à degrés variables est directe puisqu'il suffit de tirer les A_{ij} de façon indépendante (non identiquement distribués). Notez que dans ce cas, la suite $\underline{d}$ n'a même pas besoin d'être une suite de degrés et on pourrait généraliser la définition du modèle à une suite quelconque. (Ce n'est pas le cas pour $FD(\underline{d})$ comme on le verra par la suite).
- Pour générer des graphes dans $FD(\underline{d})$, on utilise soit un algorithme de *matching* (voir Algorithme 3.1) soit un algorithme re-branchement (rewiring ou switching algorithm, voir Algorithme 3.2).

L'algorithme de matching ne crée pas nécessairement un graphe simple (avant le test final, car possibilité de boucles et d'arcs multiples). Si le graphe produit n'est pas simple, il doit être jeté et on en tire un nouveau. Algorithme très peu efficace !

Attention, une correction naïve de cet algorithme, qui vérifie que $i \neq j$ ou que l'arête $\{i, j\}$ n'existe pas encore peut soit ne pas converger, soit donner des tirages biaisés de l'ensemble des graphes possibles.

L'algorithme de re-branchement est plus efficace mais il fonctionne uniquement à partir d'un graphe déjà existant qui possède la suite de degrés qu'on s'est fixée.

Algorithm 3.1: Algorithme de matching

`//Entrée` : $\underline{d} = (d_1, \ldots, d_n)$
`//Sortie` : liste d'arêtes

`//Initialisation` : Edge.List $\leftarrow$ () ; Node.List $\leftarrow$ ()
`// Créer fake Node.List :`
for $i \in \{1, \ldots, n\}$ **do**
  while $d_i \geq 1$ **do**
    Node.list $\leftarrow$ concatenate(Node.List,i)
    $d_i \leftarrow d_i - 1$

`// Créer Edge.List :`
while *Node.List is not empty* **do**
  Tirer i, j uniformément dans Node.List et sans remise
  Edge.List $\leftarrow$ concatenate(Edge.List, $\{i, j\}$)

`// Test graphe simple :`
Si Edge.List contient des boucles ou des arêtes multiples, la sortie est non valide. On recommence.

De plus, il nécessite un paramètre : le nombre d'itérations. Empiriquement, on fixe ce nombre à environ 100 fois le nombre d'arêtes du graphe.

Le comportement du modèle $FD(\underline{d})$ peut être étudié à l'aide de simulations numériques (coûteuses).

Remarque. Il faut garder à l'esprit que les degrés des nœuds sont une caractérisation très partielle du réseau et que des réseaux très différents peuvent avoir des suites de degrés de nœuds très similaires.

Application : Tests

Il est courant de vouloir tester la significativité d'une statistique T (par exemple l'une de celles proposées au chapitre 2) mesurée sur le graphe. Pour cela, on peut tenter de se placer sous un modèle statistique simple (par exemple Erdős-Rényi) et caractériser la distribution de la statistique sous ce modèle. On pourra ainsi facilement décider si la valeur observée de la statistique est dans la queue de sa distribution sous le modèle sélectionné ou pas.

En pratique, le modèle d'Erdős-Rényi est trop simple et s'ajuste très mal à des données observées. Il est plus intéressant de travailler par exemple avec les modèles

Algorithm 3.2: Algorithme de re-branchement

```
//Entrée : Edge.List ; Nb.iter
//Sortie : Edge.List
```

while *Nb.iter* ≥ 1 **do**

Choisir $e_1 = \{u_1, v_1\}$ et $e_2 = \{u_2, v_2\}$ uniformément dans Edge.List
Proposer la création de $e'_1 = \{u_1, v_2\}, e'_2 = \{u_2, v_1\}$
Si aucune boucle ni arête multiple créée : remplacer e_1, e_2 par e'_1, e'_2 dans Edge.List
Nb.iter $\leftarrow$ Nb.iter -1

de configuration (qui contrôlent la distribution des degrés des nœuds du graphe). Plus particulièrement, le modèle $FD(\underline{d})$ que nous venons de voir est défini justement par un principe de ré-échantillonnage.

Pour tester si une statistique observée sur le graphe T_{obs} est significative, le plus souvent on simule un grand nombre de graphes sous le modèle $FD(\underline{d})$ où $\underline{d}$ est la suite des degrés du graphe observé. Pour chaque graphe simulé, on calcule la valeur de la statistique sur ce graphe. On obtient ainsi une distribution empirique de cette statistique, sous l'hypothèse H_0 que le graphe initial suit la loi $FD(\underline{d})$. En comparant T_{obs} avec la distribution empirique, on décide alors si la valeur T_{obs} est inattendue ou pas sous l'hypothèse H_0.

3.3 Quelques autres modèles (non probabilistes)

Dans cette section, on rassemble des modèles de graphes aléatoires souvent issus de la physique. Cette liste n'est pas du tout exhasutive, et il existe beaucoup d'autres modèles. Pour ceux qui sont cités ici, la plupart du temps, il est difficile voire impossible d'avoir une approche probabiliste de ces modèles, *i.e.* de définir la probabilité d'un graphe donné ou d'une classe de graphes avec certaines propriétés sous ces modèles. Par ailleurs, ces modèles dépendent de. paramètres qui en général ne peuvent pas être inférés sur des données observées.

3.3.1 Attachement préférentiel

Il s'agit d'un modèle de formation dynamique des graphes, qui illustre le concept *Rich get richer*.

Paramètres : Graphe initial $G_0 = (V_0, E_0)$, entier $m \geq 1$, nombre d'itérations T.

Principe : on commence avec un petit graphe initial $G_0 = (V_0, E_0)$ et la suite de degrés associés $(d_{1,0}, \ldots, d_{|V_0|,0})$; on fabrique une suite croissante de graphes $G_t = (V_t, E_t)$. Pour cela, on itère les étapes suivantes à chaque temps $t \geq 1$,

- un nouveau nœud i_t de degré $m \geq 1$ est ajouté au réseau et $V_t = V_{t-1} \cup \{i_t\} = V_0 \cup \{i_1, \ldots, i_t\}$.
- Ce nouveau nœud se connecte avec m nœuds existants qui sont choisis chacun avec probabilité $d_{j,t-1}/(2|E_{t-1}|)$ où $d_{j,t}$ est le degré du nœud j au temps t et $2|E_t|$ la somme totale des degrés au temps t (attachement préférentiel aux nœuds de degrés les plus élevés),
- On met à jour les degrés $d_{j,t}$ pour $j \in V_t$.

A l'itération finale T, le graphe possède donc $|V_0| + T$ nœuds et $|E_0| + Tm$ arêtes.

Avantages et inconvénients :

- C'est un model génératif dynamique.
- Il permet d'expliquer la loi de puissance des degrés : à la limite ($T \to \infty$) et sous certaines conditions, la distribution des degrés du graphe suit une loi de puissance.
- Problème du choix des paramètres G_0, m, T_{final}. Impact de ce choix sur le graphe obtenu ?
- D'un point de vue statistique, ce n'est pas un modèle qu'on peut ajuster sur les données.

3.3.2 Modèle de Watts-Strogatz

Ce modèle est parfois improprement appelé « small-world » model car il a été introduit dans Watts and Strogatz (1998) dans le but d'exhiber cette propriété, combinée avec celle d'un fort coefficient de clustering. Plus exactement, Watts & Strogatz ont proposé une méthode constructive de graphe aléatoire, et montré que la classe de graphes ainsi produits a cette propriété de petit monde.

Paramètres : Nombre de nœuds N, entier $r \geq 2$, probabilité $p \in (0, 1)$.

Principe : On démarre avec un ensemble N de nœuds , disposés en cercle, et on relie chaque nœud à ses r plus proches voisins de gauche et ses r plus proches voisins de droite (cf Figure 3.3, configuration bleue). On démarre donc avec un graphe $2r$-régulier. Puis, pour chaque arête de ce graphe, avec probabilité p on change une de

ces extrêmités. Pour cela, on sélectionne uniformément un nouveau nœud mais on doit vérifier qu'on n'introduit ni boucles ni arêtes multiples (cf Figure 3.3, arêtes rouges).

La configuration initiale est celle d'un graphe avec un coefficient de clustering élevé. Le fait de perturber quelques arêtes permet de rajouter des « raccourcis » entre chaque paire de nœuds , ce qui fait baisser la distance moyenne entre deux nœuds (propriété petit monde).

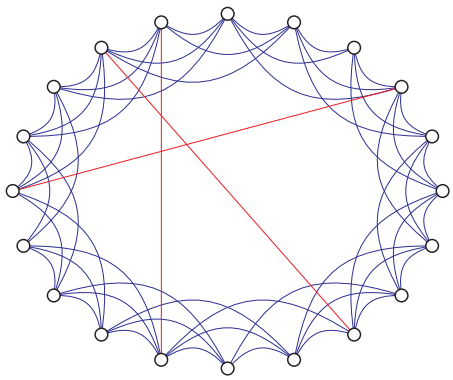

FIGURE 3.3 – Le modèle de Watts & Strogatz (source Kolaczyk (2009)).

Avantages et inconvénients :

- C'est un model génératif dynamique.
- Il permet d'obtenir des graphes petit monde avec un fort coefficient de clustering.
- Problème du choix des paramètres r, p. Impact de ce choix sur le graphe obtenu ?
- D'un point de vue statistique, ce n'est pas un modèle qu'on peut ajuster sur les données.

3.3.3 Random geometric graphs

Ce modèle est motivé par la modélisation d'interactions entre objets qui ont une **position aléatoire** dans l'espace $\mathbb{R}^d$ (on peut penser $d = 2$ ou 3 pour les applications) muni de la norme euclidienne $\|\cdot\|$: l'interaction a lieu si et seulement si les objets sont suffisamment proches dans cet espace (distance inférieure à $r > 0$). On pourra consulter Penrose (2003) pour plus de détails sur les graphes aléatoires géométriques.

Paramètres : Nombre de nœuds n, une densité f sur $\mathbb{R}^d$ et $r > 0$.

Principe : Le modèle $RGG(n, f, r)$ est la collection de graphes aléatoires sur n nœuds obtenue de la façon suivante :

- On tire n variables aléatoires $X_1, \ldots, X_n \in \mathbb{R}^d$ iid de densité f ;
- Les nœuds i, j du graphe sont reliés si et seulement si $\|X_i - X_j\| \leq r$.

Avantages et inconvénients.

- C'est un modèle de graphe dit *spatial*, dans lequel les nœuds sont associés à des positions dans l'espace. Physiquement, c'est un modèle naturel (exemple communication radio entre stations).
- La question de savoir si un graphe observé peut être la réalisation d'un RGG avec des paramètres bien choisis est un problème NP-dur.
- Une question plus abordable (mais statistiquement difficile) consiste à se demander si un graphe observé contient de l'information géométrique, *i.e.* est-ce qu'il existe un test de l'hypothèse $H_0 : G \sim G(n, p)$ contre $H_1 : G \sim RGG(n, f, r)$ (avec une dimension sous-jacente d potentiellement très grande). C'est le problème dit de geometry detection.

Chapitre 4

Échantillonnage dans les graphes aléatoires

Fréquemment, le graphe observé G est une partie ou un échantillon d'un graphe plus grand, non observé, noté $G^\star$. Plusieurs types de questions peuvent apparaître :

1. Dans quelle mesure les caractéristiques de G sont-elles une bonne approximation des caractéristiques de $G^\star$ lorsque la taille de G grandit ?
2. Lorsque l'on peut choisir le mode d'échantillonnage, quel type d'échantillonnage est à privilégier ?
3. Lorsque le type d'échantillonnage exact est inconnu, quels sont de bons modèles d'échantillonnages pour relier G et $G^\star$?

La question 1 est difficile et peu de réponses existent. Quant à la question 2, cela dépend du problème final que l'on se pose. Il convient de sélectionner un mode d'échantillonnage qui soit en adéquation avec la question de recherche sous-jacente. Pour le problème 3, nous allons voir quelques modèles d'échantillonnage parmi les plus utilisés.

Il existe différents types d'échantillonnage, on décrit ici uniquement les plus utilisés et leurs principales caractéristiques. Dans la suite, on note $G = (V, E)$ un graphe observé qui est un échantillon d'un graphe plus grand et inconnu noté $G^\star = (V^\star, E^\star)$, possédant $|V^\star| = n^\star$ nœuds.

Échantillonnages par sous-graphe induit et sous-graphe incident. L'échantillonnage par sous-graphe *induit* est obtenu comme suit : on tire n individus au hasard et sans remise parmi les $n^\star$ nœuds existants et on observe les liens entre ces nœuds.

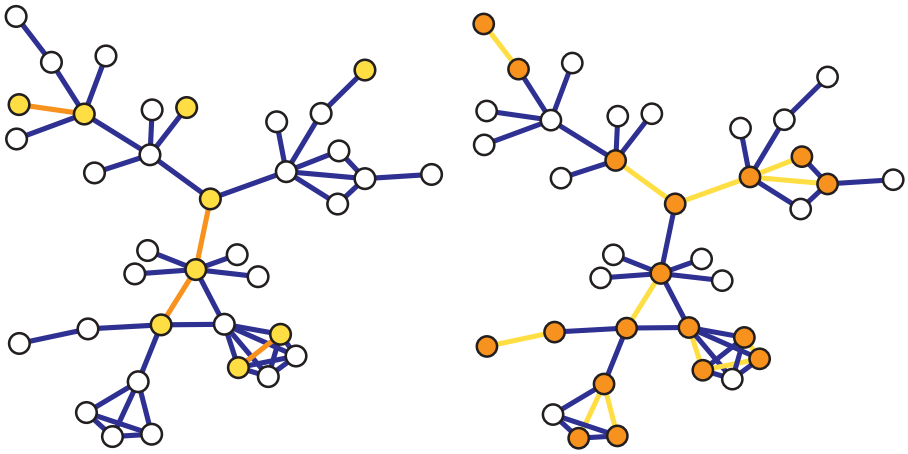

FIGURE 4.1 – À gauche, illustration de l'échantillonnage induit : les nœuds sélectionnés apparaissent en jaune et les arêtes observées en orange. À droite, illustration de l'échantillonnage incident : les arêtes sélectionnées apparaissent en jaune et les nœuds observés en orange. Source Kolaczyk (2009).

Exemples. Réseaux sociaux 'classiques' où on sélectionne des individus (au sein d'un groupe) et on les interroge sur leurs relations (amitiés, ...).

Avantages et inconvénients de l'échantillonnage induit :

- Si l'on échantillonne ainsi dans un grand graphe (ex Facebook) on obtient un graphe essentiellement vide !
- Ce type d'échantillonnage n'est pas adapté si la densité est variable au sein du graphe (même probabilité de sélectionner chaque nœud).

Les échantillonnages induit et incident sont illustrés dans la Figure 4.1.

L'échantillonnage par sous-graphe *incident* consiste en : on tire m arêtes au hasard et sans remise parmi les $m^\star$ arêtes existantes, chaque nœud incident à une arête est inclus dans le graphe.

Exemples. On a une base de données d'échanges d'email ou d'appels téléphoniques entre individus dont on extrait des entrées.

Avantages et inconvénients de l'échantillonnage incident :

- aucun nœud isolé dans ce graphe ;
- potentiellement les degrés obtenus sont très faibles (plus que dans le graphe original) car on tire peu d'arêtes incidentes aux mêmes nœuds.

Échantillonnages 'link tracing'. Le principe général des échantillonnages link tracing est le suivant : on tire n individus au hasard et sans remise parmi les $n^\star$ nœuds existants, puis on suit un sous-ensemble d'arêtes à partir de ces nœuds.

Dans l'échantillonnage *égocentrique* (également appelé star sampling) : on observe tous les liens incidents aux nœuds initiaux. Puis 2 variantes sont possibles : inclusion ou pas des nœuds supplémentaires incidents. (En général, on ne les inclue pas).

Exemples. On réalise un sondage dans une population où on demande aux individus de dire avec combien de personnes ils sont amis. On note ou pas le nom des amis (version avec ou sans inclusion des nœuds supplémentaires incidents).

Avantages et inconvénients de l'échantillonnage égocentrique :

- Les degrés des nœuds observés sont les mêmes dans G et dans $G^\star$ (ce n'est pas le cas pour les échantillonnages induits et incident).
- Quand on n'inclut pas les nœuds supplémentaires, en fait on échantillonne seulement une suite de degrés.

L'échantillonnage *Boule de neige (Snowball sampling)* (voir Figure 4.2) est un échantillonnage égocentrique itéré. Initialement, on a un ensemble V_0 de nœuds dont on observe les arêtes incidentes. Les nouveaux nœuds incidents à ces arêtes sont notés V_1, puis on observe toutes les arêtes incidentes à $V_1 \cup V_0$. Les nouveaux nœuds incidents sont notés V_2, etc .... On arrête soit lorsque le nouvel ensemble V_k est vide, soit après un nombre K d'itérations. Le graphe final est tel que $V = V_0 \cup V_1 \cup \cdots \cup V_K$ et les arêtes sont toutes les arêtes de $G^\star$ incidentes à des nœuds de V.

Exemples. Certains sondages en sciences sociales ; Web crawling ; ...

Avantages et inconvénients de l'échantillonnage boule de neige :

- À la première étape, chaque nœud a la même probabilité d'être sélectionné. Mais dès la première itération et pour toutes les suivantes, on sélectionne avec une plus grande probabilité des nœuds qui ont un degré élevé (car ils sont les voisins des nœuds sélectionnés aux étapes précedentes). On biaise donc l'échantillonnage en faveur des nœuds de degré plus élevé.

Échantillonnages 'Traceroute'. On tire un ensemble de nœuds 'sources' S et un ensemble de nœuds 'cibles' T dans $V^\star \setminus S$. Pour chaque paire (s_i, t_j), on sélectionne un chemin dans $G^\star$ de s_i à t_j : tous les nœuds et toutes les arêtes sur ces chemins sont inclus. Voir Figure 4.3.

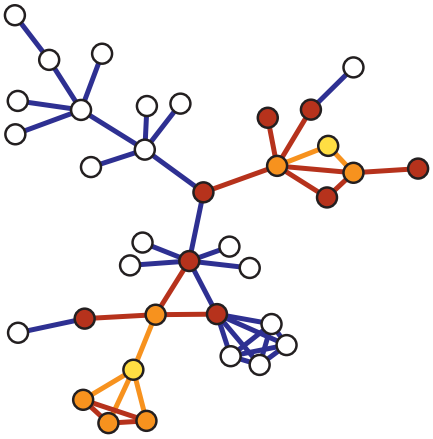

FIGURE 4.2 – Illustration de l'échantillonnage boule de neige, pour 2 itérations. Les nœuds sélectionnés à la première itération apparaissent en jaune. Puis les arêtes et les nœuds obtenus à la première itération apparaissent en orange. Enfin les arêtes et les nœuds issus de la seconde itération sont en rouge. Source Kolaczyk (2009).

Exemples. Sondages de la topologie d'internet.

Avantages et inconvénients de l'échantillonnage traceroute :

- Ce type d'échantillonnage nécessite d'être capable de sélectionner (efficacement) les chemins entre 2 nœuds.
- Cet échantillonnage est très biaisé, en particulier il produit systématiquement des distributions de degrés observés qui suivent une loi de puissance (Achlioptas et al., 2009).

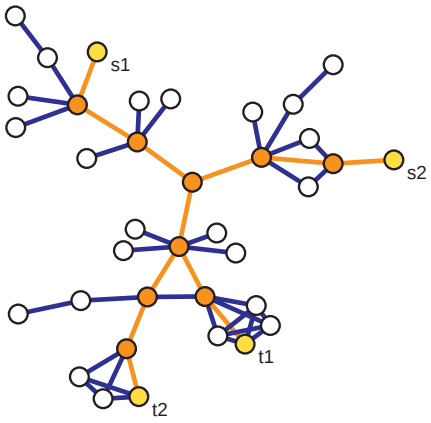

FIGURE 4.3 – Illustration de l'échantillonnage traceroute. Les nœuds source sont $\{s_1, s_2\}$ et les nœuds cibles $\{t_1, t_2\}$. En orange apparaissent les nœuds et les arêtes des chemins sélectionnés entre les sources et les cibles. Source Kolaczyk (2009).

Chapitre 5

Spectral Clustering : détection de communautés

Ce chapitre utilise en grande partie l'article de von Luxburg (2007).

L'analyse de grands graphes passe souvent par un résumé de l'information, par exemple à travers un partitionnement (clustering) des nœuds du graphe : on va alors chercher à grouper les individus en classes « homogènes », i.e. les individus dans la même classe se comportent de façon similaire au sein du graphe.

Nous verrons plusieurs façons de faire du partitionnement des nœuds d'un graphe dans ce cours. Ce chapitre s'intéresse au clustering spectral, qui est une technique de partitionnement qui détecte des nœuds très connectés entre eux. En ce sens, le clustering spectral est adapté à la *recherche de communautés* dans des graphes (une communauté est un groupe de nœuds qui forme une quasi-clique, *i.e.* qui sont très connectés entre eux).

L'heuristique du spectral clustering est simple : si le graphe est composé de communautés, alors il existe une permutation des lignes et des colonnes de la matrice d'adjacence pour laquelle cette matrice est presque diagonale par blocs. Il suffit donc de chercher à diagonaliser la matrice d'adjacence pour trouver cette permutation.

Le spectral clustering est une technique de partitionnement utilisée de façon plus large que dans la simple analyse de graphes : on va voir qu'il peut-être utilisé sur un tableau de données classique, par exemple comme une alternative à l'algorithme de k-means, à partir du graphe de similarité des données.

Les caractéristiques du spectral clustering sont :

- classification adaptée à la recherche de communautés (exclusivement) ;
- qui n'est pas fondée sur un modèle probabiliste ;
- mais qui a l'avantage de fonctionner sur de très grands graphes.

Dans tout ce chapitre, on ne considère que des graphes non dirigés (les arêtes

représentent des similarités ou des distances et sont donc symétriques).

5.1 Graphes de similarité

5.1.1 Introduction

On dispose d'un tableau de données classique de taille $n \times p$, *i.e.* n observations $x_1, \ldots, x_n$ avec $x_i \in \mathbb{R}^p$ de dimension p. On va faire de la classification (non supervisée) de cet ensemble de n points. Les techniques les plus classiquement utilisées sont les k-means ou la classification hiérarchique. Elles sont souvent fondées sur une notion de similarité $s_{ij} \geq 0$ (inversement proportionnelle à la distance) entre chaque paire d'observations x_i, x_j. À partir d'une notion de similarité entre les vecteurs $\{x_i\}_{i \leq n}$, on peut définir un graphe $G = (V, E)$ avec $V = \{v_1, \ldots, v_n\}$ ensemble des nœuds du graphe et $e = \{v_i, v_j\}$ est une arête du graphe si la similarité s_{ij} entre x_i, x_j est plus grande qu'un certain seuil. Le graphe G peut être binaire ($s_{ij} \geq s \implies \{v_i, v_j\} \in E$ et $s_{ij} < s \implies \{v_i, v_j\} \notin E$) ou valué ($s_{ij} \geq s \implies \{v_i, v_j\} \in E$ et l'arête porte la valeur s_{ij}, sinon l'arête n'est pas présente).

Le problème de clustering des points $x_1, \ldots, x_n$ peut être reformulé comme un problème de partitionnement du graphe de similarité où l'on cherche des groupes de nœuds tels que les connections intra-groupes sont importantes (les vecteurs qui correspondent aux nœuds du groupe sont très similaires entre eux) et tels que les connections inter-groupes sont faibles (peu de similarité entre les vecteurs qui correspondent à des nœuds de groupes différents).

Il y a différentes façons de définir un graphe de similarité comme on le verra dans la prochaine section.

5.1.2 Différents graphes de similarité

On considère un ensemble de n observations $x_1, \ldots, x_n$ avec $x_i \in \mathbb{R}^p$ et on dispose d'une mesure de similarité $s_{ij} \geq 0$ (l'inverse d'une distance d_{ij}) entre chaque paire d'observations x_i, x_j. On va construire différents graphes de similarité $G = (V, E)$ avec $V = \{v_1, \ldots, v_n\}$.

Définition (Graphe de similarité dense.)**.** On peut définir la similarité entre les vecteurs $\{x_j\}$ à travers les voisinages dans $\mathbb{R}^p$ (et donc la distance entre les points), par exemple $\forall i \neq j$ on pose $s_{ij} = \exp(-\|x_i - x_j\|^2/(2\sigma^2))$ pour un certain $\sigma^2 > 0$ qui contrôle la taille des voisinages dans $\mathbb{R}^p$ et $s_{ii} = 0$. Dans le cas de similarités strictement positives, on peut construire un graphe valué dense (toutes les arêtes

sont présentes) à partir des s_{ij}. Ainsi, tous les nœuds v_i, v_j avec $i \neq j$ sont connectés et le poids de l'arête $\{v_i, v_j\}$ est $s_{ij} > 0$. Le graphe ainsi construit est dense.

Définition (Graphe de ϵ-voisinage.)**.** On fixe un seuil $\epsilon > 0$ et on connecte tous les nœuds v_i, v_j tels que $s_{ij} \geq \epsilon$ (distance entre les vecteurs x_i, x_j en-dessous du seuil). Le graphe ainsi construit est binaire.

Définition (Graphe des k plus proches voisins.)**.** On commence par définir un graphe orienté $\tilde{G} = (V, \tilde{E})$. Si x_j est l'un des k plus proches voisins de x_i (*i.e.* d_{ij} est parmi les k plus petits éléments de $\{d_{il}; l \neq i\}$ ou s_{ij} est parmi les k plus grands éléments de $\{s_{il}; l \neq i\}$) alors on crée une arête (orientée) de v_i vers v_j, *i.e.* $(v_i, v_j) \in \tilde{E}$.

À partir de ce graphe orienté $\tilde{G}$, on peut définir $G = (V, E)$ non orienté de deux façons différentes :

- Soit $\{v_i, v_j\} \in E$ dès que $(v_i, v_j) \in \tilde{E}$ ou $(v_j, v_i) \in \tilde{E}$ (graphe des k plus proches voisins) ;
- Soit $\{v_i, v_j\} \in E$ dès que $(v_i, v_j) \in \tilde{E}$ et $(v_j, v_i) \in \tilde{E}$ (graphe des k plus proches voisins mutuels).

Les arêtes sont ensuite munies de leur poids s_{ij} pour former un graphe valué.

Remarques.

- Le graphe des k plus proches voisins est une sorte de compromis entre le graphe dense et le graphe de ϵ-voisinage : étape de seuillage qui réduit le bruit (comme pour le ϵ-voisinage) mais on garde les valeurs des arêtes s_{ij} les plus grandes (contrairement au ϵ-voisinage).
- Le choix du graphe de similarité entre les vecteurs x_i influe sur le résultat du partitionnement que l'on obtient sur les points. Mais on ne sait pas quel choix est meilleur a priori.
- Dans le cadre de ce cours, on dispose d'un graphe (binaire ou valué), qui est déjà construit et qui définit les relations entre nos entités. On appliquera le spectral clustering sur ce graphe.

À partir de mesures d'interactions $\{C_{ij}\}_{1 \leq i,j \leq n}$ entre individus, on peut définir une valeur symétrisée et normalisée des interactions, à partir du coefficient de Jaccard.

Définition. (Jaccard coefficient ou index de Jaccard). Il s'agit d'une mesure de similarité symétrique et normalisée entre éléments, définie à partir de valeurs d'interactions C_{ij} entre les individus, par

$$JAC_{ij} = JAC_{ji} = \frac{C_{ij} + C_{ji}}{\sum_{k \neq j} C_{ik} + \sum_{k \neq i} C_{jk}}.$$

Cet index sert parfois à construire des graphes valués et non dirigés entre un ensemble d'entités.

5.2 Matrices laplaciennes de graphe

Pour des raisons de robustesse, le clustering spectral ne diagonalise pas la matrice d'adajcence du graphe mais plutôt une version normalisée de celui-ci : une matrice *laplacienne* du graphe (de similarité) G. Il y a plusieurs définitions de matrices laplaciennes d'un graphe, ici nous n'en considérerons que certaines.

Dans la suite, G est un graphe valué et non dirigé, de matrice d'adjacence valuée A (taille $n \times n$) dont les entrées sont positives $A_{ij} \geq 0$ (il faut penser que les A_{ij} sont des similarités). On note D la matrice diagonale (de taille n) dont la diagonale vaut $(d_1, \ldots, d_n)$, avec d_i est le degré *valué* du nœud i dans G, *i.e.* $d_i = \sum_j A_{ij} = \sum_j A_{ji}$ est la somme des poids des arêtes issues de i. (Le cas d'un graphe binaire est un cas particulier du cas général que l'on décrit ici).

Pour tout vecteur (colonne) $u \in \mathbb{R}^n$, on note $u^\intercal$ le vecteur (ligne) transposé de u. On note $\mathbf{1}$ le vecteur dont toutes les coordonnées valent 1 et I la matrice identité. Les valeurs propres d'une matrice seront ordonnées de façon croissante (en respectant les multiplicités). Ainsi, les 'k premiers vecteurs propres' désignent les k vecteurs propres associés aux k plus petites valeurs propres.

Rappels : une matrice L symétrique réelle est diagonalisable dans une base orthogonale de vecteurs propres et possède n valeurs propres réelles. Si la matrice est en plus positive (*i.e.* pour tout $u \in \mathbb{R}^n$, on a $u^\intercal L u \geq 0$) alors les valeurs propres sont positives.

Nous commençons par définir la matrice laplacienne de graphe la plus simple et par donner ses propriétés spectrales (*i.e.* valeurs propres et vecteurs propres associés).

5.2.1 Laplacien non normalisé

Définition. On définit la matrice laplacienne non normalisée L d'un graphe par

$$L = D - A.$$

Proposition 5.1 (Spectre de L)**.** *La matrice L vérifie les propriétés suivantes*

1. *Pour tout vecteur $u \in \mathbb{R}^n$ on a*

$$u^\intercal L u = \frac{1}{2} \sum_{i,j=1}^{n} A_{ij}(u_i - u_j)^2. \tag{5.1}$$

2. *L est une matrice symétrique et positive.*
3. *La plus petite valeur propre de L est 0 de vecteur propre associé $\mathbf{1}$.*

4. L possède n valeurs propres réelles positives, notées $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.

Démonstration. Par définition de $L = D - A$ et de la matrice D on a $\forall u \in \mathbb{R}^n$,

$$\begin{aligned} u^\intercal L u = u^\intercal D u - u^\intercal A u &= \sum_{i=1}^{n} u_i^2 d_i - \sum_{i,j} u_i A_{ij} u_j \\ &= \frac{1}{2}\Big(\sum_{i=1}^{n} d_i u_i^2 - 2 \sum_{i,j} u_i u_j A_{ij} + \sum_{j=1}^{n} d_j u_j^2 \Big) \\ &= \frac{1}{2} \sum_{i,j} A_{ij} (u_i - u_j)^2, \end{aligned}$$

(car $\sum_j A_{ij} = d_i$ et $\sum_i A_{ij} = d_j$). Ceci prouve le point 1.

Comme D et A sont symétriques, la matrice $L = D - A$ l'est aussi. D'après (5.1), on a pour tout $u \in \mathbb{R}^n, u^\intercal L u \geq 0$, donc L est positive. En conséquence, ses valeurs propres sont réelles et positives, on les note $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Enfin, par définition de L, on voit que $\mathbf{1}$ est un vecteur propre, associé à la valeur propre 0 (puisque $\sum_i A_{ij} = d_i$). □

Remarques.

1. La définition de L est inchangée si on modifie la diagonale de A (tant que D est toujours défini comme la matrice diagonale dont les entrées sont la somme des lignes de A). On peut le voir à partir de la définition $L = D - A$ ou à partir de l'équation (5.1). En particulier si on a « oublié » de mettre une diagonale nulle sur A (par exemple à partir de la fonction de similarité exp vue plus haut), cela n'aura pas d'impact sur L (ni sur son spectre).
2. Attention : cette remarque n'est pas du tout valable pour les laplaciens qui vont suivre ! Donc il est préférable de bien faire attention à la diagonale de A.

Proposition 5.2 (Nombre de composantes connexes de G et spectre de L)**.** *Soit G un graphe valué dont les poids des arêtes sont positifs et L la matrice laplacienne non normalisée associée. Alors la multiplicité de 0 en tant que valeur propre de L est exactement le nombre de composantes connexes du graphe G. Si on note $C_1, \ldots, C_k \subset \{v_1, \ldots, v_n\}$ ces composantes connexes et $\mathbf{1}_{C_1}, \ldots, \mathbf{1}_{C_k}$ les vecteurs indicatrices des composantes (définis par $\mathbf{1}_{C_l}(i) = 1$ si $v_i \in C_l$ et $\mathbf{1}_{C_l}(i) = 0$ sinon), alors l'espace propre associé à la valeur propre 0 est engendré par $\mathbf{1}_{C_1}, \ldots, \mathbf{1}_{C_k}$.*

Démonstration. On commence par considérer le cas $k = 1$ d'une seule composante connexe dans le graphe. Si $u \in \mathbb{R}^n$ est un vecteur propre de L associé à la valeur

propre 0, on a $Lu = 0$ et d'après (5.1),

$$u^\intercal Lu = 0 = \sum_{i,j} A_{ij}(u_i - u_j)^2.$$

Puisque $A_{i,j} \geq 0$, la somme est nulle seulement si tous ses termes sont nuls, ie seulement si pour tout $1 \leq i, j \leq n$ on a $A_{ij}(u_i - u_j)^2 = 0$. Si l'arête $\{v_i, v_j\} \in E$ alors le poids A_{ij} est non nul et nécessairement $u_i = u_j$. Donc le vecteur propre $u \in \mathbb{R}^n$ est constant sur les coordonnées correspondant à des nœuds connectés dans le graphe. Par définition d'une composante connexe (tous les nœuds dans la composante sont connectés), et puisqu'on a une seule composante connexe (cas $k = 1$), on a $u_i = cte$ pour tout $i \in \{1, \ldots, n\}$. Donc u est proportionnel à $\mathbf{1}$, *i.e.* le vecteur $\mathbf{1}$ engendre l'espace propre associé à la valeur propre 0.

Si $k \geq 2$. Soient $C_1, \ldots, C_k \subset \{v_1, \ldots, v_n\}$ les composantes connexes du graphe G. Sans perte de généralité, on peut supposer que les nœuds de V sont ordonnés selon la composante à laquelle ils appartiennent. Alors, la matrice d'adjacence A a une forme diagonale par blocs (puisque si v_i, v_j ne sont pas dans la même composante connexe, alors $A_{ij} = 0$). En conséquence, $L = D - A$ a aussi une forme diagonale par blocs

$$L = \begin{pmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_k \end{pmatrix}.$$

Chacun des blocs L_i (de taille $n_i \times n_i$) est une matrice laplacienne, associé au sous-graphe $G_i = (C_i, E_i) \subset G$ induit par la i-ème composante connexe C_i (de cardinal n_i) de G. L'ensemble des valeurs propres de L (= spectre de L) est la réunion des spectres de chaque L_i et les vecteurs propres correspondants sont formés par les vecteurs propres de L_i, augmentés de coordonnées nulles aux positions des autres blocs. Comme pour chaque sous-graphe G_i, on a une seule composante connexe, le résultat précédent nous dit que l'espace propre associé à la valeur propre 0 de L_i est engendré par $\mathbf{1}_{C_i}$ (dans $\mathbb{R}^{n_i}$). On obtient donc que l'espace propre associé à la valeur propre 0 de L est engendré par les vecteurs $\mathbf{1}_{C_1}, \ldots, \mathbf{1}_{C_k}$ (en tant que vecteurs de $\mathbb{R}^n$ cette fois). □

Remarque. L'étude du spectre du laplacien d'un graphe permet donc de déterminer simplement le nombre de composantes connexes de ce graphe.

Le laplacien L est intéressant car on peut faire facilement des calculs de spectre et comprendre ce qui se passe. Cependant pour le clustering il ne donne pas les meilleurs résultats numériques possibles et on lui préfère des versions normalisées.

5.2.2 Laplaciens normalisés

Définition. On considère une matrice laplacienne normalisée définie par

$$L_N = I - D^{-1/2} A D^{-1/2}.$$

NB : dans la littérature, il existe d'autres définitions de laplacien normalisé.

Rappels.
- Comme D est une matrice diagonale, la matrice $D^{-1/2}$ est une matrice dont les éléments diagonaux valent $1/\sqrt{d_i}$ (ce n'est pas vrai si D n'est pas diagonale !).
- La multiplication à gauche par une matrice diagonale revient à multiplier les vecteurs lignes de la matrice, tandis qu'une multiplication à droite multiplie les vecteurs colonnes. Ainsi, $D^{-1/2}AD^{-1/2}$ est la matrice dont chaque entrée i, j vaut $A_{ij}/\sqrt{d_i d_j}$. Ainsi,

$$L_N = \begin{pmatrix} 1 & & & \\ & 1 & & -\frac{A_{ij}}{\sqrt{d_i d_j}} \\ & & \star & \ddots \\ & & & 1 \end{pmatrix}$$

C'est une matrice symétrique (puisque A l'est et D est diagonale).

Proposition 5.3 (Spectre de L_N)**.** *La matrice laplacienne normalisée vérifie les propriétés suivantes :*

1. *Pour tout $u \in \mathbb{R}^n$,*
$$u^\intercal L_N u = \frac{1}{2} \sum_{1 \le i,j \le n} A_{ij} \Big(\frac{u_i}{\sqrt{d_i}} - \frac{u_j}{\sqrt{d_j}} \Big)^2.$$
2. *0 est valeur propre de L_N, de vecteur propre associé $D^{1/2}\mathbf{1}$.*
3. *La matrice L_N est une matrice positive qui possède n valeurs propres réelles positives.*

Démonstration. Soit $u \in \mathbb{R}^n$, on a

$$\begin{aligned} u^\intercal L_N u = u^\intercal (I - D^{-1/2} A D^{-1/2}) u &= \sum_{i=1}^n u_i^2 - \sum_{1 \le i,j \le n} u_i u_j \frac{A_{ij}}{\sqrt{d_i d_j}} \\ &= \frac{1}{2} \Big(\sum_{i=1}^n u_i^2 - 2 u_i u_j \frac{A_{ij}}{\sqrt{d_i d_j}} + \sum_{j=1}^n u_j^2 \Big) \\ &= \frac{1}{2} \sum_{1 \le i,j \le n} A_{ij} \Big(\frac{u_i}{\sqrt{d_i}} - \frac{u_j}{\sqrt{d_j}} \Big)^2, \end{aligned}$$

car $\sum_j A_{ij} = d_i$ et $\sum_i A_{ij} = d_i$. On a donc prouvé le point 1.

On a $L_N D^{1/2}\mathbf{1} = D^{1/2}\mathbf{1} - D^{-1/2}A\mathbf{1} = D^{1/2}\mathbf{1} - D^{-1/2}(d_1, \ldots, d_n)^\intercal = D^{1/2}(\mathbf{1} - \mathbf{1}) = 0$, donc 0 est valeur propre de L_N associé au vecteur propre $D^{1/2}\mathbf{1}$.

D'après le point 1, L_N est positive donc ses valeurs propres sont réelles et positives. □

La multiplicité de la valeur propre 0 du laplacien normalisé est reliée au nombre de composantes connexes du graphe.

Proposition 5.4 (Nombre de composantes connexes de G et spectre de L_N)**.** *Soit G un graphe valué dont les poids des arêtes sont positifs et L_N la matrice laplacienne normalisée définie ci-dessus. Alors la multiplicité de la valeur propre 0 de L_N est égale au nombre de composantes connexes du graphe G. Si on note $C_1, \ldots, C_k \subset \{v_1, \ldots, v_n\}$ ces composantes connexes et $\mathbf{1}_{C_1}, \ldots, \mathbf{1}_{C_k}$ les vecteurs indicatrices des composantes (définis par $\mathbf{1}_{C_l}(i) = 1$ si $v_i \in C_l$ et $\mathbf{1}_{C_l}(i) = 0$ sinon), alors l'espace propre associé à la valeur propre 0 est engendré par $D^{1/2}\mathbf{1}_{C_1}, \ldots, D^{1/2}\mathbf{1}_{C_k}$.*

Démonstration. En exercice.

□

Remarques.

- En pratique, on s'intéresse couramment à des graphes qui n'ont qu'une seule composante connexe (s'il y en a plusieurs, autant les étudier séparément). Dans ce cas, on sait que 0 est valeur propre de multiplicité 1 et que l'espace propre associé est engendré par le vecteur $D^{1/2}\mathbf{1}$: pas très intéressant. L'étude du spectre n'apporte rien de plus sur cette question.
- On utilise le spectre du laplacien de la façon suivante : on s'intéresse aux k premiers vecteurs propres de L_N : c'est similaire à une ACP (analyse en composantes principales) ou du MDS (multi-dimensional scaling). Dans ce nouvel espace, les points initiaux (nœuds du graphe) sont mieux séparés et un simple clustering (type k-means) donne de bons résultats.

Enfin, on définit également

$$L_{\text{abs}} = D^{-1/2}AD^{-1/2} = I - L_N.$$

Cette matrice L_{abs} a exactement les mêmes vecteurs propres que L_N. Si on note $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ les valeurs propres de L_N alors les valeurs propres de L_{abs} sont $1 - \lambda_n \leq \ldots \leq 1 - \lambda_2 \leq 1 - \lambda_1 = 1$. Attention : dans L, L_N ce sont les petites valeurs propres qui contiennent l'information intéressante alors que pour L_{abs} on va voir que ce sont les *grandes valeurs propres, en valeur absolue* !

5.3 Algorithmes de clustering spectral

Tout comme il existe de nombreuses définitions de la matrice laplacienne d'un graphe, il existe de nombreux algorithmes de clustering spectral. Nous en verrons uniquement 2 : l'algorithme de spectral clustering normalisé qui utilise L_N (Algorithme 5.1) et l'absolute spectral clustering fondé sur L_{abs} (Algorithme 5.2).

Algorithm 5.1: Spectral clustering normalisé de Ng et al. (2001)

`//Entrée` : A de taille $n \times n$ d'entrées positives, nombre k de clusters
`//Sortie` : Clusters $C_1, \ldots, C_k$ qui partitionnent $\{1, \ldots, n\}$

Calculer la matrice laplacienne normalisée L_N
Calculer les k vecteurs propres $u_1, \ldots, u_k$ associés aux plus petites valeurs propres de L_N
Former la matrice U de taille $n \times k$ dont les **colonnes** sont $u_1, \ldots, u_k$
Former la matrice T de taille $n \times k$ en normalisant les lignes de U pour avoir une norme euclidienne 1 (*i.e.* $t_{ij} = u_{ij}/\sqrt{\sum_k u_{ik}^2}$)
Créer des clusters $C_1, \ldots, C_k$ sur les n **lignes** de T par k-means

Algorithm 5.2: Absolute Spectral clustering de Rohe et al. (2011).

`//Entrée` : A de taille $n \times n$ d'entrées positives, nombre k de clusters
`//Sortie` : Clusters $C_1, \ldots, C_k$ qui partitionnent $\{1, \ldots, n\}$

Calculer la matrice laplacienne L_{abs}
Calculer les k vecteurs propres $u_1, \ldots, u_k$ de L_{abs} associés aux k **plus grandes valeurs propres en valeur absolue**
Former la matrice U de taille $n \times k$ dont les **colonnes** sont $u_1, \ldots, u_k$
Créer des clusters $C_1, \ldots, C_k$ sur les n **lignes** de U par k-means

Le principe du spectral clustering est donc de transformer les observations de départ $x_i \in \mathbb{R}^p, 1 \leq i \leq n$ en un nouvel ensemble de points $y_i \in \mathbb{R}^k, 1 \leq i \leq n$ (=les lignes de la matrice U), via un graphe de similarité, la matrice laplacienne associée et ses k premiers vecteurs propres. Les propriétés de ces matrices laplaciennes font que ce nouvel ensemble de points y_i est facilement classifiable en k groupes (un simple algorithme k-means suffit à bien séparer ces nouveaux points).

Si le graphe de départ a k composantes connexes, les k premiers vecteurs propres $u_1, \ldots, u_k$ engendrent l'espace propre associé à la valeur propre 0, et on a vu que cet

espace est engendré par les vecteurs indicatrices $\mathbf{1}_{C_1}, \ldots, \mathbf{1}_{C_k}$. Si on applique l'algorithme des k-means sur les lignes de U avec k groupes, on retrouve exactement les composantes connexes $C_1, \ldots, C_k$. Par analogie, lorsqu'on a une seule composante connexe, les algorithmes de spectral clustering vont donner un partitionnement des nœuds du graphe en un ensemble de 'presque composantes connexes' ou plus exactement, de communautés.

Le spectral clustering en valeur absolue a une propriété supplémentaire : il va chercher des structures de type biparties dans le graphe. Il tend à mettre dans le même groupe des nœuds qui partagent beaucoup de voisins.

5.4 Exemples jouets

On considère ici le cas de quelques graphes remarquables : on étudie leur spectre (avec L au lieu de L_N ou de L_{abs} car les calculs sont plus simples) et on essaye de voir l'impact sur le principe du clustering.

Proposition 5.5. *1. Soit K_n le graphe complet sur n nœuds , alors les valeurs propres du laplacien L associé sont : 0 de multiplicité 1 et n de multiplicité $n-1$.*

2. Soient i, j deux nœuds de degré 1 qui partagent le même voisin k dans le graphe G. Alors le vecteur $u \in \mathbb{R}^n$ défini par $u_i = 1, u_j = -1$ et $u_l = 0$ pour tout $l \in \{1, \ldots, n\} \setminus \{i, j\}$ est un vecteur propre du laplacien L associé à la valeur propre 1.

3. Soit S_n le graphe en étoile sur n nœuds , alors les valeurs propres du laplacien L associé sont : 0 de multiplicité 1, 1 de multiplicité $n-2$ et n de multiplicité 1.

Démonstration. 1. K_n est connexe donc 0 est valeur propre de multiplicité 1, associée au vecteur propre $\mathbf{1}$. Soit $u \in \mathbb{R}^n$ un vecteur propre associé à une valeur propre $\lambda > 0$, alors u est orthogonal à $\mathbf{1}$, i.e. $\sum_{i=1}^n u_i = 0$. Sans perte de généralité, on peut supposer $u_1 \neq 0$ et on a $u_1 = -\sum_{i=2}^n u_i \neq 0$. De plus, le laplacien L de K_n vérifie $L_{ij} = -1$ si $i \neq j$ et $L_{ii} = n-1$. On obtient alors

$$(Lu)_1 = \sum_{i=1}^n L_{1i} u_i = (n-1)u_1 - \sum_{i=2}^n u_i = n u_1.$$

Donc si u est un vecteur propre pour la valeur propre λ, on a $\lambda u_1 = (Lu)_1 = n u_1$. Donc n est la seule autre valeur propre (elle a la multiplicité $n-1$ et est associé à

n'importe quel vecteur orthogonal à $\mathbf{1}$).
2. Quitte à réordonner les nœuds du graphe, on peut écrire le laplacien sous la forme

$$L = \begin{pmatrix} 1 & 0 & -1 & 0\dots 0 \\ 0 & 1 & -1 & 0\dots 0 \\ -1 & -1 & d_k & \star\star\star \\ 0 & 0 & \star & \\ \vdots & \vdots & \star & \star\star \\ 0 & 0 & \star & \end{pmatrix}.$$

Alors, le vecteur $u^\intercal = (1, -1, 0, \dots, 0)$ est un vecteur propre associé à la valeur propre 1.
3. On considère à présent le graphe en étoile S_n. Il est connexe donc 0 est valeur propre de multiplicité 1, associée au vecteur propre $\mathbf{1}$. On numérote 1 le nœud au centre de l'étoile et de 2 à n les nœuds au bout des branches. En appliquant le résultat du point 2 pour les nœuds $(i, i+1)$ pour i allant de 2 à $n-1$ (ce sont des nœuds de degré 1 qui partagent le nœud 1 en commun), on obtient (n-2) vecteurs $u^{(i)}$ associés à la valeur propre 1. On vérifie qu'ils sont linéairement indépendants (écrire $\sum_{i=2}^{n-1} \alpha_i u^{(i)} = 0$ et voir que nécessairement $\alpha_i = 0$).

Enfin pour trouver la dernière valeur propre λ, on utilise $Tr(L) = \sum_{i=1}^{n} \lambda_i = \lambda + 0 + (n-2)$. Or $Tr(L) = (n-1) + 1 \times (n-1) = 2n-2$, donc $\lambda = Tr(L) - n + 2 = n$. (Le vecteur propre correspondant est nécessairement constant sur les indices i allant de 2 à n et orthogonal à $\mathbf{1}$, on peut déduire facilement sa forme). □

Conséquences.

- Le résultat pour K_n indique que si on fait un clustering des lignes de U avec $k > 1$ groupes, on n'obtient rien qui fasse du sens. C'est normal puisqu'il n'y a qu'une seule communauté dans K_n.
- Pour S_n, le clustering spectral trouve soit une seule communauté, soit $n-1$ communautés : le nœud central associé à un nœud au hasard, puis chaque autre nœud tout seul.

Proposition 5.6. *1. Un graphe est bipartie si et seulement si le spectre de L_{abs} est symétrique.*

2. Un graphe connexe est bipartie si et seulement si $\lambda_{\min}(L_{abs}) = -\lambda_{\max}(L_{abs})$.

Démonstration. Admis. □

Conséquences. On comprend que l'absolute spectral clustering qui regarde les plus grandes valeurs propres en valeur absolue va capturer les structures de type bipartie.

5.5 Commentaires pratiques

- Le choix de la fonction de similarité (quand on part de données non graphe) doit dépendre du type de données.
- Une différence importante entre le graphe d'ϵ-voisinage et les graphes des k plus proches voisins (simple ou mutuel) est l'adaptation locale du voisinage des seconds : les tailles de voisinage sont différentes en fonction des régions de l'espace (plus grandes dans les régions peu denses, plus petites dans les régions plus denses).
- Le graphe des k plus proches voisins *mutuel* tend à connecter entre eux des points dans des régions de densité constante (comme la version *simple*) mais ne connecte pas entre elles des régions proches mais de densité différente. En ce sens, c'est un compromis entre ϵ-voisinage et k plus proches voisins simple.
- Les graphes des k plus proches voisins sont plus faciles à manipuler que le graphe construit avec une similarité gaussienne (qui lui est dense). Il peuvent donc être préférables ; mais attention à la perte d'information : on peut par exemple avoir plus de composantes connexes dans ces graphes que de clusters désirés !
- Recommandations empiriques pour les choix des paramètres :
  — prendre k de l'ordre de $\log(n)$ pour le graphe des k-plus proches voisins simple et plus grand (sans règle explicite) pour le graphe des k-plus proches voisins mutuel. Il faut de toute façon regarder le nombre de composantes connexes obtenues, le comparer au nombre de clusters voulus et ajuster en conséquence.
  — prendre ϵ tel que le graphe résultant soit connecté.
  — pas de bonne règle pour le choix de σ dans la similarité gaussienne.
- On a vu que si le graphe a p composantes connexes, alors l'espace propre associé à la valeur propre 0 a pour dimension p et est engendré par les indicatrices des clusters. Cependant, la sortie d'un algorithme de décomposition spectrale est n'importe quelle base orthogonale de vecteurs propres de cet espace (*i.e.* pas forcément la base des vecteurs d'indicatrices mais une base issue d'une combinaison linéaire de celle-ci). Par contre, le k-means sur ces vecteurs permet d'obtenir simplement les clusters. (En fait, la matrice U n'a que k lignes différentes, on peut faire le clustering visuellement).
- Le choix du nombre de clusters k est un problème récurrent du clustering. Ici, pas de modèle probabiliste donc pas de critère type BIC ou reposant sur une vraisemblance mais on peut utiliser d'autres critères ad-hoc type 'similarité intra-groupes et inter-groupes'. Une technique courante consiste à utiliser

l'heuristique du 'trou des valeurs propres' (eigengap) : on choisit le nombre de clusters k par

$$\hat{k} = \underset{1 \leq j \leq n-1}{\text{Argmax}} \, \lambda_{j+1}(L_N) - \lambda_j(L_N).$$

Rem : il n'y a pas d'équivalent pour L_{abs}.

Chapitre 6

Modèles de graphes aléatoires et classification des nœuds

Nous avons déjà vu le modèle $\mathcal{G}(n,p)$ et constaté qu'il s'ajustait mal sur les réseaux réels observés (hypothèses d'indépendance entre les arêtes et uniformité de la probabilité de connection dans le graphe trop restrictives).

6.1 Généralités sur les modèles à variables latentes

6.1.1 Définitions

Les modèles à variables latentes (ie non observées) supposent l'existence d'une variable aléatoire (latente) associée à chaque observation et qui caractérise la distribution de cette observation. Cette variable latente peut être soit à valeurs continues, soit à valeurs discrètes (finies). Dans ce dernier cas, on obtient naturellement une classification des observations en fonction de la valeur latente. Ainsi, dans un modèle à variables latentes, on dispose d'une suite d'observations $(X_i)_{1\leq i\leq n}$ et on suppose qu'il existe des variables latentes (non observées) $(Z_i)_{1\leq i\leq n}$ telles que la loi de X_i conditionnelle aux $(Z_j)_{1\leq j\leq n}$ ne dépend que de Z_i. Pour des raisons de commodité, on suppose même le plus souvent que la loi des $(X_i)_{1\leq i\leq n}$ sachant les $(Z_i)_{1\leq i\leq n}$ est le produit des lois de chaque X_i conditionnelle à Z_i uniquement. On fait ainsi une hypothèse *d'indépendance conditionnelle* des observations.

$$\mathbb{P}((X_i)_{1\leq i\leq n}|(Z_i)_{1\leq i\leq n}) = \prod_{i=1}^{n} \mathbb{P}(X_i|Z_i).$$

Lorsque les $(Z_i)_{1\leq i\leq n}$ sont indépendantes, on obtient alors que les $(X_i)_{1\leq i\leq n}$ sont aussi des variables indépendantes (mais non identiquement distribuées). Il s'agit des

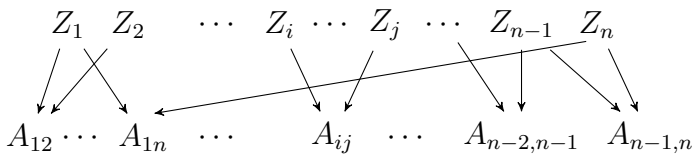

FIGURE 6.1 – Dépendances entre les variables d'un modèle à variables latentes pour graphes.

modèles de mélange (finis lorsque les Z_i sont à valeurs finies). Lorsque les $(Z_i)_{1\leq i\leq n}$ forment une chaîne de Markov, alors les $(X_i)_{1\leq i\leq n}$ ne sont plus indépendantes (seulement conditionnellement indépendantes) et on obtient les chaînes de Markov cachées (hidden Markov models).

Lorsqu'on observe un graphe aléatoire, on a vu que l'on dispose en fait d'un ensemble de variables $(A_{ij})_{1\leq i,j\leq n}$ (binaires ou valuées). La modélisation par variables latentes *naïve* consisterait à supposer l'existence de variables non observées $(Z_{ij})_{1\leq i,j\leq n}$ qui caractérisent la distribution des $(A_{ij})_{1\leq i,j\leq n}$. Cette approche est naïve car elle ne tient pas compte du fait que la donnée A_{ij} est une caractérisation du lien entre les individus i et j. Il est en fait plus naturel d'envisager qu'il existe des variables latentes $(Z_i)_{1\leq i\leq n}$ qui caractérisent les individus et que la variable A_{ij} de relation entre i et j a une distribution qui est caractérisée par la valeur de Z_i et de Z_j.

Dans toute la suite, on va donc supposer qu'il existe des variables $(Z_i)_{1\leq i\leq n}$ indépendantes et identiquement distribuées (iid), à valeurs continues ou discrètes et finies, telles que la loi conditionnelle des $(A_{ij})_{1\leq i,j\leq n}$ sachant les $(Z_i)_{1\leq i\leq n}$ vérifie

$$\mathbb{P}((A_{ij})_{1\leq i,j\leq n}|(Z_i)_{1\leq i\leq n}) = \prod_{1\leq i,j\leq n} \mathbb{P}(A_{ij}|Z_i, Z_j).$$

Il faut remarquer que même si les Z_i sont indépendantes, les A_{ij} ne le sont plus du tout : la structure de dépendance entre les variables aléatoires est compliquée par le fait que par exemple A_{ij} et A_{ik} dépendent tout les deux de la même variable latente Z_i (voir Figure 6.1).

6.1.2 Estimation des paramètres

Considérons la vraisemblance d'un modèle à variables latentes : la distribution des $(A_{ij})_{1\leq i,j\leq n}$ n'est donnée que conditionnellement aux variables latentes $(Z_i)_{1\leq i\leq n}$,

on écrit donc

$$\begin{aligned} L(\theta) =& \mathbb{P}_\theta((A_{ij})_{1\le i,j\le n}) = \int_{z_1}\cdots\int_{z_n} \mathbb{P}_\theta((A_{ij})_{1\le i,j\le n}, Z_1 = z_1, \dots, Z_n = z_n) dz_1 \dots dz_n \\ =& \int_{z_1}\cdots\int_{z_n} \prod_{i=1}^n \mathbb{P}_\theta(Z_i = z_i) \times \prod_{i,j} \mathbb{P}_\theta(A_{ij}|Z_i = z_i, Z_j = z_j) dz_1 \dots dz_n. \end{aligned}$$

En pratique, si les Z_i sont à valeurs dans $\{1, \dots, Q\}$, les intégrales ci-dessus sont des sommes et on a Q^n termes à sommer. Lorsque n n'est pas très petit ($n \ge 10$), cette somme n'est pas accessible numériquement en un temps raisonnable. Si les Z_i sont à valeurs continues, on peut approcher les intégrales en les discrétisant (par exemple sur Q points) et le problème reste exactement le même.

Dans un modèle à variables latentes, il n'est pas possible (en général) de faire un calcul efficace de la vraisemblance. L'estimation des paramètres se fait généralement en utilisant l'algorithme `EM` (expectation-maximization) qui approche l'estimateur du maximum de vraisemblance.

L'algorithme `EM`. L'algorithme EM (expectation-maximization) est un algorithme itératif qui permet de maximiser (localement) la vraisemblance dans des modèles à données manquantes (typiquement, les modèles à variables latentes sont des modèles à données manquantes).

Supposons que l'on ait un modèle avec données observées $X_{1:n}$ et données manquantes (ie non observées) $S_{1:m}$ (pas nécessairement de même taille). On appelle données complètes l'ensemble des variables $(S_{1:m}, X_{1:n})$.

Le principe de l'algorithme `EM` est le suivant :

- On part d'une valeur initiale θ^0 du paramètre,
- À l'itération k, on effectue les deux étapes
  - *Expectation* : on calcule $Q(\theta, \theta^k) := \mathbb{E}_{\theta^k}(\log \mathbb{P}_\theta(S_{1:m}, X_{1:n})|X_{1:n})$.
  - *Maximization* : on maximise $\theta^{k+1} := \mathrm{Argmax}_\theta\, Q(\theta, \theta^k)$.
- Arrêt lorsque $\delta := \|\theta^{k+1} - \theta^k\|/\|\theta^k\| \le \epsilon$ ou un nombre maximum d'itérations est atteint.

À chaque itération, la vraisemblance (observée) augmente. En effet, par construction on sait que $Q(\theta^{k+1}, \theta^k) \ge Q(\theta^k, \theta^k)$, *i.e.* :

$$\begin{aligned} 0 \le& \mathbb{E}_{\theta^k}\left[\log \frac{\mathbb{P}_{\theta^{k+1}}(S_{1:m}, X_{1:n})}{\mathbb{P}_{\theta^k}(S_{1:m}, X_{1:n})}\Big| X_{1:n}\right] \underset{\text{Ineg. Jensen}}{\le} \log \mathbb{E}_{\theta^k}\left[\frac{\mathbb{P}_{\theta^{k+1}}(S_{1:m}, X_{1:n})}{\mathbb{P}_{\theta^k}(S_{1:m}, X_{1:n})}\Big| X_{1:n}\right] \\ =& \log \int_{\mathcal{S}^m} \frac{\mathbb{P}_{\theta^{k+1}}(S_{1:m} = s_{1:m}, X_{1:n})}{\mathbb{P}_{\theta^k}(S_{1:m} = s_{1:m}, X_{1:n})} \mathbb{P}_{\theta^k}(S_{1:m} = s_{1:m}|X_{1:n}) ds_1 \dots ds_m \\ =& \log \int_{\mathcal{S}^m} \frac{\mathbb{P}_{\theta^{k+1}}(s_{1:m}, X_{1:n})}{\mathbb{P}_{\theta^k}(X_{1:n})} ds_1 \dots ds_m = \log \frac{\mathbb{P}_{\theta^{k+1}}(X_{1:n})}{\mathbb{P}_{\theta^k}(X_{1:n})}. \end{aligned}$$

Ainsi, $\mathbb{P}_{\theta^{k+1}}(X_{1:n}) \geq \mathbb{P}_{\theta^k}(X_{1:n})$.

Donc l'algorithme `EM` converge (quand le nombre d'itérations augmente) vers un maximum local de la vraisemblance. En lançant l'algorithme avec plusieurs initialisations, on devrait atteindre le maximum global.

L'algorithme `EM` est particulièrement adapté au cas où les variables latentes sont à valeurs finies. Nous reviendrons sur son application dans le cadre du modèle à blocs stochastiques.

6.2 Espaces latents continus (pour graphes binaires)

Les modèles à espaces latents continus n'ont été développés que pour les graphes binaires.

6.2.1 Modèle à positions latentes de Hoff *et al.*

Le modèle à positions latentes (latent position model) de Hoff et al. (2002) a été proposé pour étudier des réseaux sociaux. Dans ce modèle, les variables latentes sont i.i.d. à valeurs dans $\mathbb{R}^q$ qui représente un *espace social*. La proximité des individus dans cet espace induit une plus grande probabilité de connexion dans le graphe. Ainsi, seule la position relative des variables latentes entres elles est importante pour le modèle (et pas leur position absolue).

On considère un graphe binaire non dirigé $(A_{ij})_{1\leq i,j\leq n}$ et (possiblement) des vecteurs de covariables $\mathbf{x}_{ij} \in \mathbb{R}^s$ sur chaque relation (i,j). On utilise un modèle de régression logistique

$$\text{logit}(\mathbb{P}(A_{ij}=1|Z_i,Z_j,\mathbf{x}_{ij})) = \log\frac{\mathbb{P}(A_{ij}=1|Z_i,Z_j,\mathbf{x}_{ij})}{1-\mathbb{P}(A_{ij}=1|Z_i,Z_j,\mathbf{x}_{ij})} = \alpha+\beta^\intercal\mathbf{x}_{ij}-\|Z_i-Z_j\|,$$

où $\|\cdot\|$ est la norme euclidienne dans l'espace latent $\mathbb{R}^q$. Les paramètres du modèle sont $(\alpha,\beta) \in \mathbb{R}\times\mathbb{R}^s$. On peut remplacer norme euclidienne par n'importe quelle distance.

Le paramètre α règle la densité du graphe. Il faut remarquer que les variables $\{Z_i\}_i$ ne peuvent être reconstituées qu'à rotation, symétrie axiale et translation près. En effet, chacune de ces opérations laisse l'ensemble des distances $(\|Z_i-Z_j\|)_{i,j}$ inchangé et donc ne modifie pas le modèle. On appelle *configurations équivalentes* deux ensembles $\{Z_i\}_i$ et $\{Z'_i\}_i$ qui induisent les mêmes valeurs de distances $(\|Z_i-Z_j\|)_{i,j} = (\|Z'_i-Z'_j\|)_{i,j}$.

Ainsi, pour des valeurs des paramètres (α,β) fixées, deux configurations équivalentes $\{Z_i\}_i$ et $\{Z'_i\}_i$ induisent la même distribution sur les observations, et réciproquement,

si α et β sont fixés alors si on a deux ensembles de configuration $\{Z_i\}_i$ et $\{Z'_i\}_i$ qui induisent la même loi alors les configurations sont équivalentes.

Estimation des paramètres et des variables latentes. Le package `latentnet` propose une méthode d'estimation bayésienne des paramètres et des positions latentes. Voir TP pour plus de détails.

6.2.2 Version classifiante du modèle

Dans le modèle précédent, les nœuds du graphe ne sont pas naturellement classifiés en groupes qui permettent de les interpréter. On peut obtenir une telle classification en combinant l'approche avec un modèle de mélange sur les variables latentes (Handcock et al., 2007).

Ainsi, on suppose que les variables latentes $Z_i \in \mathbb{R}^q$ sont en fait générées selon un modèle de mélange de lois gaussiennes multi-dimensionnelles $\mathcal{N}_q(m_k, \sigma_k^2 Id)$ avec $1 \leq k \leq K$, de proportions $\pi_k, 1 \leq k \leq K$, de moyennes différentes $(m_k, 1 \leq k \leq K)$ et des matrices de covariance sphériques $(\sigma_k^2 Id)$.

Le choix du nombre de clusters K se fait automatiquement dans ce cadre bayésien : on place une loi a priori sur K et on estime par le maximum a posteriori. Il faut noter que les groupes obtenus sont nécessairement des communautés : si deux variables Z_i, Z_j sont dans la même composante gaussienne, alors elles sont proches dans $\mathbb{R}^q$ et la probabilité que les nœuds i, j soient connectés est plus grande.

6.2.3 Choix de la dimension de l'espace latent

En pratique, il n'existe aucun méthode permettant de choisir la dimension q de l'espace latent (attention, cette dimension n'est pas le nombre de clusters K de la méthode de Handcock et al. (2007) !).

Les logiciels sont implémentés avec $q = 2$ (ou 3) mais rien ne permet d'affirmer que ce choix est pertinent, ni qu'il n'a pas un impact majeur sur les résultats.

6.3 Espaces latents finis : Modèles à blocs stochastiques (stochastic block model)

6.3.1 Le modèle

Dans cette section, les variables latentes $\mathbf{Z} := \{Z_1, \ldots, Z_n\}$ sont i.i.d. à valeurs finies dans $\{1, \ldots, Q\}$ et de loi $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_Q)$. Il sera parfois pratique de voir

plutôt Z_i comme un vecteur de taille Q de la forme $Z_i = (Z_{i1}, \dots, Z_{iQ})$ dont les coordonnées sont dans $\{0, 1\}$, somment à 1 et tel que Z_i est de loi multinomiale $\mathcal{M}(1, \boldsymbol{\pi})$.

On va décrire le modèle à blocs stochatiques (SBM) dans le cadre d'un graphe non dirigé mais les notations se généralisent facilement au cas dirigé. On considère donc la matrice d'adjacence d'un graphe non dirigé $\mathbf{A} := \{A_{ij}\}_{1 \leq i < j \leq n}$ constitué de variables aléatoires $A_{ij} \in \mathcal{A}$ (cas binaire ou valué), qui caractérisent les relations entre les nœuds i et j.

Comme précédemment, conditionellement aux variables latentes $\mathbf{Z} = \{Z_i\}_{1 \leq i \leq n}$, les variables $\mathbf{A} = \{A_{ij}\}_{i,j}$ sont indépendantes et la distribution de chaque A_{ij} ne dépend que de Z_i et Z_j. On note $F(\cdot; \gamma_{Z_i Z_j})$ cette distribution conditionnelle, où $\boldsymbol{\gamma} = (\gamma_{q\ell})_{1 \leq q, \ell \leq Q}$ est appelé paramètre de connectivité. C'est une matrice symétrique dans le cas d'un graphe non dirigé puisque $\gamma_{q\ell} = \gamma_{\ell q}$. Le paramètre $\gamma_{q\ell}$ décrit la loi des interactions entre des nœuds des groupes q et ℓ.

Ainsi, le modèle à blocs stochastiques est caractérisé par

- $\mathbf{Z} = Z_1, \dots, Z_n$ variables latentes i.i.d. de loi $\boldsymbol{\pi}$ sur $\{1, \dots, Q\}$,
- $\mathbf{A} = \{A_{ij}\}_{i,j}$ ensemble d'observations à valeurs dans $\mathcal{A}$,
- $\mathbb{P}(\mathbf{A}|\mathbf{Z}) = \prod_{i,j} \mathbb{P}(A_{ij}|Z_i, Z_j)$ (indépendance conditionnelle),
- $\forall i, j$ et $\forall 1 \leq q, \ell \leq Q$, on a $A_{ij}|\{Z_i = q, Z_j = \ell\} \sim F(\cdot; \gamma_{q\ell})$.

On va distinguer à présent le SBM binaire (apparu dès le début des années 80 en Sciences Sociales) du cas valué (beaucoup plus récent).

Dans le cas binaire, la loi conditionnelle de A_{ij} sachant Z_i, Z_j est simplement une loi de Bernoulli $\mathcal{B}(\gamma_{Z_i Z_j})$. Ainsi,

$$\forall y \in \{0, 1\}, \quad F(y; \gamma) = \gamma^y (1 - \gamma)^{1-y}.$$

Pour les graphes valués, on peut utiliser pour modéliser la loi conditionnelle de A_{ij} sachant Z_i, Z_j n'importe quelle loi paramétrique qui dépend seulement de Z_i, Z_j (ex : Poisson, Gaussienne, Laplace, ...). Cependant, si cette loi est absolument continue par rapport à la mesure de Lebesgue, on récupère un graphe valué dense, ce qui n'est pas toujours adéquat. Pour pallier ce problème, on introduit un mélange avec une masse de Dirac en 0 (notée $\delta_0(\cdot)$) qui modélise les arêtes absentes. Ainsi,

$$\forall y \in \mathcal{A}, \quad F(y; \gamma) = \alpha G(y, \eta) + (1 - \alpha)\delta_0(y),$$

où le paramètre de connectivité $\gamma = (\alpha, \eta)$ avec $\alpha \in [0, 1]$ et $G(\cdot, \eta)$ est la loi conditionnelle sur les valeurs des arêtes présentes.

Pour des raisons d'identifiabilité, il est préférable de restreindre G à être une loi absolument continue en 0. En effet, dans le cas contraire, on ne peut pas identifier

α. Si G est absolument continue en 0, alors on a $\alpha_{q\ell} = 1 - \mathbb{P}(Y_{ij} = 0 | Z_i = q, Z_j = \ell)$. Si on veut utiliser une loi de Poisson par exemple, on utilise pour G la loi de Poisson tronquée en 0. Les valeurs nulles de Y_{ij} sont ainsi dues uniquement à la masse de Dirac δ_0 et on obtient une loi dite « à inflation » ou « à déflation » de zéros. L'avantage étant que la densité du graphe n'est pas nécessairement liée à la valeur moyenne des arêtes présentes.

Si tous les $\alpha_{q\ell}$ valent 1, le graphe est dense (toutes les arêtes sont présentes). Ainsi les $\alpha_{q\ell}$ sont des paramètres de densité du graphe. Si tous les $\alpha_{q\ell}$ valent 0, on obtient un graphe vide (ie sans arêtes), ce qui n'est pas très intéressant.

Lorsque la loi $G(\cdot, \eta)$ est une masse de Dirac en 1 (indépendante de η), on retrouve le SBM binaire. Le seul paramètre de la loi conditionnelle est alors α. Les cas classiques pour le choix de G sont : une loi de Poisson tronquée en 0, une gaussienne (multivariée), etc.

Dans le cas non binaire, on peut (pour des raisons de parcimonie), supposer que tous les $\alpha_{q\ell}$ sont constants (égaux à un certain α fixé). Alors, la densité des arêtes est homogène dans le graphe, seule leur intensité (ie la valeur de A_{ij}) va varier en fonction des groupes (q, ℓ).

Dans la suite, on note le paramètre global du modèle $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{\gamma}) = (\boldsymbol{\pi}, \boldsymbol{\alpha}, \boldsymbol{\eta}) = ((\pi_1, \dots, \pi_Q); (\alpha_{q\ell})_{q,\ell}; (\eta_{q\ell})_{q,\ell})$. La vraisemblance du modèle s'écrit

$$\begin{aligned}
\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{A}) &= \sum_{z_1=1}^{Q} \cdots \sum_{z_n=1}^{Q} \mathbb{P}_{\boldsymbol{\theta}}(\mathbf{A}, Z_1 = z_1, \dots, Z_n = z_n) \\
&= \sum_{z_1=1}^{Q} \cdots \sum_{z_n=1}^{Q} \Big(\prod_{i=1}^{n} \pi_{z_i} \Big) \times \Big(\prod_{i,j} F(A_{ij}; \gamma_{z_i z_j}) \Big) \\
&= \sum_{z_1=1}^{Q} \cdots \sum_{z_n=1}^{Q} \Big(\prod_{q=1}^{Q} \prod_{i=1}^{n} \pi_q^{z_{iq}} \Big) \times \Big(\prod_{1 \le q,\ell \le Q} \prod_{i,j} F(A_{ij}; \gamma_{q\ell})^{z_{iq} z_{j\ell}} \Big),
\end{aligned}$$

et la log-vraisemblance des données complètes s'écrit simplement

$$\log \mathbb{P}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{Z}) = \sum_{q=1}^{Q} \sum_{i=1}^{n} Z_{iq} \log \pi_q + \sum_{1 \le q,\ell \le Q} \sum_{i,j} Z_{iq} Z_{j\ell} \log F(A_{ij}; \gamma_{q\ell}). \tag{6.1}$$

Cas particulier : affiliation (planted partition model). Liens avec la détection de communauté. Lorsque le paramètre de connectivité $\boldsymbol{\gamma}$ ne prend que deux valeurs différentes : une valeur intra-groupes et une valeur inter-groupes, on parle de modèle d'affiliation (ou parfois, dans le cas binaire, de 'planted partition model'). Il

s'agit d'un sous-modèle où on contraint :

$$\forall 1 \leq q, \ell \leq Q, \quad \gamma_{q\ell} = \begin{cases} \gamma_{\text{in}} & \text{lorsque } q = \ell, \\ \gamma_{\text{out}} & \text{lorsque } q \neq \ell. \end{cases} \tag{6.2}$$

Dans le cas d'un graphe binaire, sous un modèle d'affiliation, si on suppose en plus que $\gamma_{\text{in}} \gg \gamma_{\text{out}}$, la classification des nœuds induite par le modèle correspond exactement à une détection de communautés : on cherche des groupes de nœuds fortement connectés entre eux. Dans un modèle d'affiliation avec $\gamma_{\text{out}} \gg \gamma_{\text{in}}$, on va au contraire chercher des structures de type 'multi-parties'.

Dans le cas général (pas affiliation), on récupère avec SBM une classification des nœuds en groupes de nœuds qui 'se connectent de la même façon' aux autres groupes. C'est un type de classification beaucoup moins contraint que la simple détection de communautés. Ces différences sont illustrées sur l'exemple jouet de la Figure 6.2.

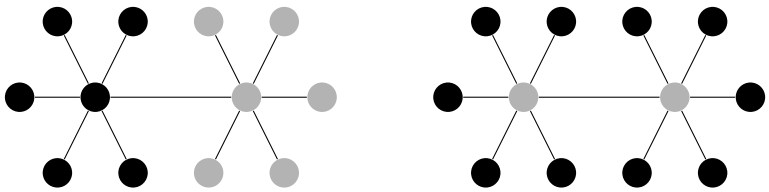

FIGURE 6.2 – Exemple jouet de structures de classification différentes (couleurs gris/noir) obtenues à partir du même graphe. À gauche, le résultat d'une méthode de détection de communautés ou d'une méthode SBM. À droite, une classification qui pourrait également être obtenue à partir du SBM mais pas à partir de la détection de communautés : les hubs forment un premier groupe tandis que les nœuds 'périphériques' forment le second. Cette seconde classification ne peut pas s'obtenir avec du clustering spectral (ni normalisé ni absolu).

6.3.2 L'algorithme `EM`

Nous avons vu qu'une façon d'approcher le maximum de vraisemblance dans un modèle à variables latentes est d'utiliser l'algorithme `EM`. Cependant, l'étape `E` de l'algorithme requiert de pouvoir calculer facilement la loi des observations $\{A_{ij}\}_{i,j}$ sachant les variables latentes $\{Z_i\}_i$. C'est le cas par exemple pour des modèles de mélange finis classiques (pas sur des graphes), ou dans les modèles de Markov cachés. Dans le cas de variables latentes sur des graphes où chaque observation A_{ij} dépend de deux variables latentes Z_i, Z_j ce n'est plus possible.

Digression sur les modèles graphiques et les dépendances conditionnelles. Un modèle graphique est un modèle probabiliste dans lequel un graphe représente la structure de dépendance de la distribution d'un ensemble de variables aléatoires. Il existe deux types de modèles graphiques : les modèles dirigés (où le graphe de dépendances est dirigé) et les modèles non dirigés (ou le graphe de dépendances est non dirigé). On pourra se référer à Lauritzen (1996) ou au chapitre 8 de Bishop (2006) pour en savoir plus. On aura besoin également de deux définitions préliminaires.

Définition. Dans un graphe dirigé, les *parents* d'un nœud $j \in V$ sont tous les nœuds $i \in V$ tels qu'il existe une **arête** orientée de i vers j. Les *descendants* du nœud $i \in V$ sont tous les nœuds $j \in V$ tels qu'il existe un **chemin** orienté de i vers j.

Soit $\mathbb{P}$ une distribution sur $\mathcal{X}^V$ et $\mathcal{G} = (V, E)$ un graphe tel que

- l'ensemble $V = \{1, \dots, p\}$ des nœuds indexe un ensemble de variables aléatoires $\{X_i\}_{i \in V}$ à valeurs dans $\mathcal{X}^p$,
- L'ensemble des arêtes E décrit les relations de dépendance entre les v.a. $\{X_i\}_{i \in V}$ sous la loi $\mathbb{P}$ (plus de détails ci-dessous).

Dans un modèle graphique, on a

- Soit $\mathcal{G}$ est un graphe acyclique et dirigé (DAG), alors $\mathbb{P}$ se factorise selon $\mathcal{G}$ ie on a
$$\mathbb{P}(\{X_i\}_{i \in V}) = \prod_{i \in V} \mathbb{P}(X_i | pa(X_i, \mathcal{G})),$$
où $pa(X_i, \mathcal{G})$ sont les variables parents de X_i dans $\mathcal{G}$.
- Soit $\mathcal{G}$ est non dirigé, alors pour tout $\{i, j\} \notin E$, on a $X_i \perp\!\!\!\perp X_j \mid X_{V \setminus \{i,j\}}$ où $X_{V \setminus \{i,j\}}$ représente toutes les autres variables sauf X_i, X_j ; ie
$$\mathbb{P}(X_i, X_j | X_{V \setminus \{i,j\}}) = \mathbb{P}(X_i | X_{V \setminus \{i,j\}}) \mathbb{P}(X_j | X_{V \setminus \{i,j\}}).$$
Une formulation équivalente et que l'on utilise fréquemment est
$$\mathbb{P}(X_i | X_j; X_{V \setminus \{i,j\}}) = \mathbb{P}(X_i | X_{V \setminus \{i,j\}}).$$

Exemples.

- Réseaux bayésiens (modèle graphique dirigé). Ex : Chaînes de Markov, ou chaînes de Markov cachées (voir Figure 6.3).
- Champs de Markov (modèle non dirigé).
- Modèles graphiques gaussiens (modèle non dirigé).

Remarques.

- Attention : la terminologie « modèle graphique » n'a rien à voir avec des données organisées sous forme de graphes. Les variables aléatoires X_i ne traduisent pas (a priori) des interactions entre des entités. Le graphe est un objet abstrait qui structure la dépendance entre les variables aléatoires.

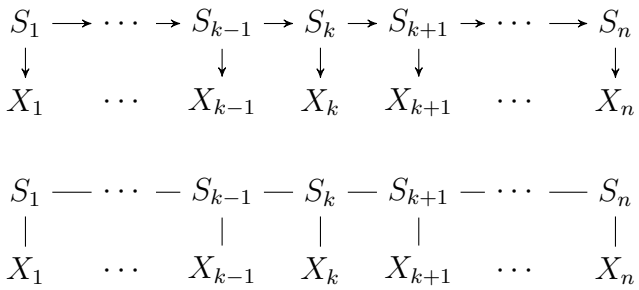

FIGURE 6.3 – Graphe acyclique dirigé (haut) et graphe moral (bas) correspondant à un modèle de Markov caché.

- Si $\mathbb{P}$ se factorise selon un DAG $\mathcal{G}$, alors $\mathcal{G}$ n'est pas unique en général.
  Ex : sans contrainte sur $\mathbb{P}$, on a $\mathbb{P}(X_1, X_2, X_3) = \mathbb{P}(X_3|X_1, X_2)\mathbb{P}(X_2|X_1)\mathbb{P}(X_1) = \mathbb{P}(X_{\sigma(3)}|X_{\sigma(1)}, X_{\sigma(2)})\mathbb{P}(X_{\sigma(2)}|X_{\sigma(1)})\mathbb{P}(X_{\sigma(1)})$, pour toute permutation σ (voir Figure 6.4).

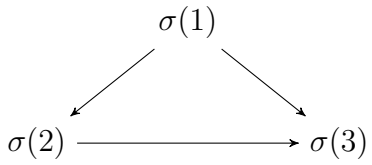

FIGURE 6.4 – DAG qui factorise n'importe quelle distribution sur 3 variables. Ici σ est n'importe quelle permutation de $\{1, 2, 3\}$.

Dans un modèle graphique, lorsque la structure de dépendance est représentée par un graphe acyclique dirigé, on construit le **graphe moral** associé au DAG $\mathcal{G}$. C'est un graphe non dirigé, qui est obtenu à partir de $\mathcal{G}$ en « mariant » les parents (*i.e.* on relie les parents par des arêtes) puis en retirant les directions des arêtes (voir Figure 6.3 pour un exemple dans le cas des chaînes de Markov cachées). Lorsque le graphe $\mathcal{G}$ est non dirigé, il est égal à son graphe moral.

Proposition 6.1 (Propriétés d'indépendance)**.** *Dans un modèle graphique caractérisé par un graphe $\mathcal{G} = (V, E)$, on a*

- *Si $\mathcal{G}$ est un DAG, alors conditionnellement à ses parents (dans $\mathcal{G}$), une variable est indépendante de ses non-descendants (dans $\mathcal{G}$). Autrement dit, si on note $desc(X_i, \mathcal{G})$ l'ensemble des descendants de X_i dans $\mathcal{G}$ et si K est un sous-ensemble de V tel que $K \cap desc(X_i, \mathcal{G}) = \emptyset$, alors*

$$\mathbb{P}(X_i|pa(X_i, \mathcal{G}), \{X_k\}_{k\in K}) = \mathbb{P}(X_i|pa(X_i, \mathcal{G})).$$

- *Soient I, J, K des sous ensembles disjoints de V. Alors dans le graphe moral associé à $\mathcal{G}$, si tous les chemins de I à J passent par K, alors* $\{X_i\}_{i\in I} \perp\!\!\!\perp \{X_j\}_{j\in J} \mid \{X_k\}_{k\in K}$.

Exemple . On considère le DAG et le graphe moral associé représentés à la Figure 6.5. On a par exemple

- X_1 et X_3 sont indépendantes ;
- Sachant X_2, les variables X_1 et X_3 ne sont pas indépendantes ;
- Sachant X_2, la variable X_5 est indépendante de X_1, X_3, X_4 ;
- X_2 est indépendante de X_6 sachant X_5 ;
- Sachant X_5, la variable X_6 est indépendante de X_1, X_2, X_3, X_4 ;
- X_1 est indépendante de X_4 sachant X_2 ;
- ...

FIGURE 6.5 – Exemple de DAG (gauche) et graphe moral associé (droite).

Exemple d'application. On se place dans le modèle de chaîne de Markov caché illustré à la Figure 6.3. Par conditionnement, on peut écrire

$$\mathbb{P}(S_1, \dots, S_n | X_1, \dots, X_n) = \mathbb{P}(S_n | S_{n-1}, \dots, S_1, X_1, \dots X_n)\mathbb{P}(S_{n-1}, \dots, S_1 | X_1, \dots X_n).$$

D'après le graphe moral (ou le DAG), on a

$$\mathbb{P}(S_n | S_{n-1}, \dots, S_1, X_1, \dots X_n) = \mathbb{P}(S_n | S_{n-1}, X_n)$$

et ainsi

$$\mathbb{P}(S_1, \dots, S_n | X_1, \dots, X_n) = \mathbb{P}(S_n | S_{n-1}, X_n)\mathbb{P}(S_{n-1}, \dots, S_1 | X_1, \dots X_n)$$

On procède récursivement en utilisant la propriété suivante (qui découle du graphe moral ou du DAG)

$$\mathbb{P}(S_k | S_{k-1}, \dots, S_1, X_1, \dots X_n) = \mathbb{P}(S_k | S_{k-1}, X_k, \dots X_n)$$

et on obtient au final

$$\mathbb{P}(S_1, \dots, S_n | X_1, \dots, X_n) = \prod_{k=2}^{n} \mathbb{P}(S_k | S_{k-1}, X_k, \dots X_n) \times \mathbb{P}(S_1 | X_1).$$

Ainsi, la loi des variables latentes sachant les observations est celle d'une chaîne de Markov (inhomogène). La forme factorisée de cette loi la rend aisément manipulable.

Retour sur les modèles à variables latentes pour graphes. L'algorithme `EM` requiert de pouvoir calculer facilement la loi des variables latentes sachant les observations. Nous allons voir sur la Figure 6.6 pourquoi cette distribution n'a pas une structure simple. En effet, la figure de gauche montre le DAG associé à un modèle de graphes avec variables latentes et à droite, son graphe moral associé. Dans ce dernier, on voit que sachant les observations, on a toujours des dépendances entre les variables Z_i (présence de chemins entre Z_i et Z_j que l'on ne peut pas « bloquer » avec les variables observées). Ainsi, la distribution des Z_i sachant les A_{ij} n'est pas factorisée ! (Alors que c'est le cas pour un modèle de mélange, pour les HMMs, etc). C'est cette propriété qui empêche d'appliquer l'algorithme `EM` ici.

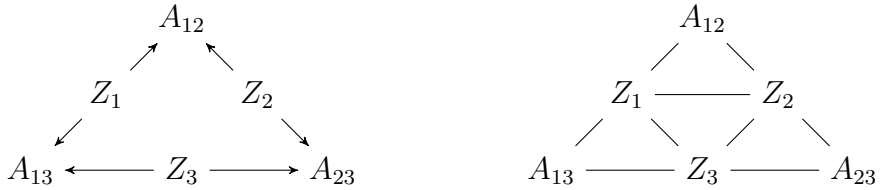

FIGURE 6.6 – À gauche : DAG d'un modèle à variable latentes pour un graphe ($n = 3$). À droite : graphe moral associé.

Nous allons nous intéresser à une stratégie d'approximation variationnelle qui permet de pallier ce problème.

6.3.3 Estimation des paramètres par approximation variationnelle de `EM`

La raison qui empêche l'utilisation de l'algorithme `EM` dans notre cadre est le fait que la loi des variables latentes $\{Z_i\}_i$ sachant les observations $\{A_{ij}\}_{ij}$ n'est pas factorisée. Une solution naturelle consiste à remplacer cette loi par la meilleure approximation possible dans la classe des lois factorisées. C'est le principe de l'approximation variationnelle. Pour l'expliquer, nous allons d'abord revenir sur le principe détaillé de l'algorithme `EM`, en le présentant avec un point de vue légèrement différent.

La log-vraisemblance des observations peut se décomposer sous la forme

$$\mathcal{L}_{\mathbf{A}}(\boldsymbol{\theta}) := \log \mathbb{P}_{\boldsymbol{\theta}}(\mathbf{A}) = \log \mathbb{P}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{Z}) - \log \mathbb{P}_{\boldsymbol{\theta}}(\mathbf{Z}|\mathbf{A}).$$

Si $\mathbb{Q}$ est une distribution de probabilité sur l'ensemble des variables $\{Z_i\}_i$, on peut prendre l'espérance par rapport à $\mathbb{Q}$ de chaque côté de l'égalité précédente et on obtient

$$\mathcal{L}_{\mathbf{A}}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbb{Q}}(\log \mathbb{P}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{Z})) - \mathbb{E}_{\mathbb{Q}}(\log \mathbb{P}_{\boldsymbol{\theta}}(\mathbf{Z}|\mathbf{A})).$$

En notant $\mathcal{H}(\mathbb{Q})$ l'entropie de la loi $\mathbb{Q}$ et $\mathcal{KL}(\mathbb{Q}\|\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{Z}|\mathbf{A}))$ la divergence de Kullback-Leibler entre les lois $\mathbb{Q}$ et $\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{Z}|\mathbf{A})$, c'est-à-dire

$$\mathcal{H}(\mathbb{Q}) = -\sum_z \mathbb{Q}(z)\log\mathbb{Q}(z) = -\mathbb{E}_{\mathbb{Q}}(\log\mathbb{Q}(Z))$$

$$\mathcal{KL}(\mathbb{Q}\|\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{Z}|\mathbf{A})) = \sum_z \mathbb{Q}(z)\log\frac{\mathbb{Q}(z)}{\mathbb{P}_{\boldsymbol{\theta}}(z|\mathbf{A})} = \mathbb{E}_{\mathbb{Q}}\Big(\log\frac{\mathbb{Q}(\mathbf{Z})}{\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{Z}|\mathbf{A})}\Big),$$

on obtient alors l'égalité suivante

$$\mathcal{L}_{\mathbf{A}}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbb{Q}}(\log\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{A},\mathbf{Z})) + \mathcal{H}(\mathbb{Q}) + \mathcal{KL}(\mathbb{Q}\|\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{Z}|\mathbf{A})). \tag{6.3}$$

Partant de cette relation (6.3), l'algorithme `EM` (qui cherche à maximiser $\mathcal{L}_{\mathbf{A}}(\boldsymbol{\theta})$) consiste à itérer les deux étapes suivantes. À partir de la valeur courante du paramètre $\theta^{(t)}$, on effectue

- `E`-step : on maximise la quantité $\mathbb{E}_{\mathbb{Q}}(\log\mathbb{P}_{\boldsymbol{\theta}^{(t)}}(\mathbf{A},\mathbf{Z})) + \mathcal{H}(\mathbb{Q})$ par rapport à $\mathbb{Q}$. D'après (6.3), puisque $\mathcal{L}_{\mathbf{A}}(\boldsymbol{\theta}^{(t)})$ ne dépend pas de $\mathbb{Q}$, c'est équivalent à minimiser $\mathcal{KL}(\mathbb{Q}\|\mathbb{P}_{\boldsymbol{\theta}^{(t)}}(\mathbf{Z}|\mathbf{A}))$ par rapport à $\mathbb{Q}$. La solution optimale est donc la loi conditionnelle $\mathbb{P}_{\boldsymbol{\theta}^{(t)}}(\mathbf{Z}|\mathbf{A})$ pour la valeur courante du paramètre $\boldsymbol{\theta}^{(t)}$;
- `M`-step : on garde à présent $\mathbb{Q}$ fixé et on maximise la quantité $\mathbb{E}_{\mathbb{Q}}(\log\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{A},\mathbf{Z}))+\mathcal{H}(\mathbb{Q})$ par rapport à $\boldsymbol{\theta}$. Puisque $\mathbb{Q}$ ne dépend pas de $\boldsymbol{\theta}$, c'est équivalent à maximiser l'espérance conditionnelle $\mathbb{E}_{\mathbb{Q}}(\log\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{A},\mathbf{Z}))$ par rapport à $\boldsymbol{\theta}$. Avec notre choix de $\mathbb{Q}$, cette quantité est exactement l'espérance conditionnelle de la log-vraisemblance des données complètes, sachant les observations, sous le paramètre courant, ie $\mathbb{E}_{\boldsymbol{\theta}^{(t)}}(\log\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{A},\mathbf{Z})|\mathbf{A})$ que l'on maximise en $\boldsymbol{\theta}$. En effet, on a

  $$\begin{aligned}\mathbb{E}_{\mathbb{Q}}(\log\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{A},\mathbf{Z})) =& \sum_{\mathbf{Z}} \mathbb{Q}(\mathbf{Z})\log\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{A},\mathbf{Z}) = \sum_{\mathbf{Z}} \mathbb{P}_{\boldsymbol{\theta}^{(t)}}(\mathbf{Z}|\mathbf{A})\log\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{A},\mathbf{Z})\\ =&\mathbb{E}_{\boldsymbol{\theta}^{(t)}}(\log\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{A},\mathbf{Z})|\mathbf{A}).\end{aligned}$$

  Comme on l'a vu précédemment, maximiser cette quantité par rapport à θ va automatiquement accroître la log-vraisemblance des observations $\mathcal{L}_{\mathbf{A}}(\boldsymbol{\theta})$ parce que le terme de divergence de Kullback-Leibler est égal à 0 ici par l'étape `E` !

Lorsque la vraie loi $\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{Z}|\mathbf{A})$ n'est pas manipulable (par exemple parce que ce n'est pas une loi factorisée), la solution exacte du `E`-step ne peut pas être calculée. Dans l'approximation variationnelle, au lieu de calculer la solution exacte à l'étape `E`, on va chercher une solution optimale dans une classe restreinte de distributions, par exemple dans la classe des lois factorisées (et l'étape `M` reste inchangée mais utilise la solution approchée $\mathbb{Q}$ de l'étape dite `VE` pour variational-expectation).

Au final, on peut remarquer en reprenant (6.3) et en utilisant le fait qu'une divergence de Kullback-Leibler est toujours positive (par l'inégalité de Jensen), qu'on a la borne inférieure suivante

$$\mathcal{L}_{\mathbf{A}}(\boldsymbol{\theta}) \geq \mathbb{E}_{\mathbb{Q}}(\log \mathbb{P}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{Z})) + \mathcal{H}(\mathbb{Q}) := \mathcal{J}(\mathbb{Q}, \boldsymbol{\theta}). \tag{6.4}$$

Ainsi, l'approximation variationnelle optimise une borne inférieure de la log-vraisemblance ($\mathcal{J}(\mathbb{Q}, \boldsymbol{\theta})$ optimisée d'abord en $\mathbb{Q}$ puis en $\boldsymbol{\theta}$). On n'a aucune garantie d'approcher l'estimateur de maximum de vraisemblance avec cette procédure. En général, on ne l'approche d'ailleurs pas. Dans le cas particulier du SBM, cette procédure fonctionne cependant très bien empiriquement et il existe également des résultats théoriques qui justifient son utilisation.

Ainsi, dans le cas du modèle à blocs stochastiques, on prend donc pour $\mathbb{Q}$ une loi factorisée (*i.e.* marginales indépendantes)

$$\mathbb{Q}(\mathbf{Z}) = \prod_{i=1}^{n} \mathbb{Q}_i(Z_i) = \prod_{i=1}^{n} \prod_{q=1}^{Q} \tau_{iq}^{Z_{iq}},$$

où $\tau_{iq} = \mathbb{Q}_i(Z_i = q) = \mathbb{E}_{\mathbb{Q}}(Z_{iq})$, avec $\sum_q \tau_{iq} = 1$ pour tout i.

L'approximation variationnelle est parfois appelée *approximation champ moyen* parce que tout se passe comme si dans l'approximation de la loi conditionnelle de Z_i sachant les observations, toutes les autres variables $\{Z_{jq}\}_{j \neq i, q}$ étaient fixées à leur moyennes (conditionnelles) τ_{jq}. Les paramètres τ_{iq} sont appelés paramètres variationnels. Ils représentent l'approximation de la probabilité que le nœud i appartienne au groupe q. À la fin de l'algorithme `VEM` (pour variational expectation maximization), on peut utiliser un maximum a posteriori pour retrouver les groupes latents et faire la classification

$$\forall 1 \leq i \leq n, \quad \hat{Z}_i = \operatorname*{Argmax}_{1 \leq q \leq Q} \tau_{iq}.$$

Calculs dans le cas SBM. On va entrer dans les détails de l'implémentation de `VEM` dans le cas du SBM. On reprend l'expression (6.1) de la log-vraisemblance des données complètes

$$\log \mathbb{P}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{Z}) = \sum_{q=1}^{Q} \sum_{i=1}^{n} Z_{iq} \log \pi_q + \sum_{1 \leq q, \ell \leq Q} \sum_{i,j} Z_{iq} Z_{j\ell} \log F(A_{ij}; \gamma_{q\ell}).$$

En prenant l'espérance par rapport à la loi $\mathbb{Q}$ de cette quantité, puisque $\mathbb{E}_{\mathbb{Q}}(Z_{iq} Z_{j\ell}) = \tau_{iq}\tau_{j\ell}$ (propriété d'indépendance sous la loi $\mathbb{Q}$) et $\mathbb{E}_{\mathbb{Q}}(Z_{iq}) = \tau_{iq}$ (par définition), on

obtient l'expression suivante

$$\mathbb{E}_{\mathbb{Q}}(\log \mathbb{P}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{Z})) = \sum_{q=1}^{Q} \sum_{i=1}^{n} \tau_{iq} \log \pi_q + \sum_{1 \leq q, \ell \leq Q} \sum_{i,j} \tau_{iq} \tau_{j\ell} \log F(A_{ij}; \gamma_{q\ell}).$$

Ainsi, la quantité qui nous intéresse est

$$\begin{aligned} \mathcal{J}(\mathbb{Q}, \boldsymbol{\theta}) &= \mathbb{E}_{\mathbb{Q}}(\log \mathbb{P}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{Z})) + \mathcal{H}(\mathbb{Q}) \\ &= \sum_{q=1}^{Q} \sum_{i=1}^{n} \tau_{iq} \log \Big(\frac{\pi_q}{\tau_{iq}} \Big) + \sum_{1 \leq q, \ell \leq Q} \sum_{i,j} \tau_{iq} \tau_{j\ell} \log F(A_{ij}; \gamma_{q\ell}), \end{aligned}$$

et on alterne une maximisation de $\mathcal{J}$ par rapport aux τ_{iq} avec une maximisation par rapport aux paramètres $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{\gamma})$.

Ainsi, à l'étape `E`, on maximise cette quantité par rapport aux paramètres variationnels τ_{iq} pour une valeur $\boldsymbol{\theta}$ fixée. En cherchant les points critiques (ne pas oublier les contraintes $\forall i, \sum_q \tau_{iq} = 1$), on obtient que la solution $\hat{\tau} = \{\hat{\tau}_{iq}\}_{i,q}$ vérifie une équation de point fixe

$$\forall 1 \leq i \leq n, \forall 1 \leq q \leq Q, \quad \hat{\tau}_{iq} \propto \pi_q \prod_{j} \prod_{\ell=1}^{Q} [F(A_{ij}; \gamma_{q\ell})]^{\hat{\tau}_{j\ell}},$$

où $\propto$ signifie 'proportionnel à' (la constante est obtenue à partir de la contrainte de loi de probabilité !).

À l'étape `M`, on doit maximiser en $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{\gamma})$ cette même quantité. Concernant la maximisation par rapport aux π_q, on obtient facilement

$$\forall 1 \leq q \leq Q, \quad \hat{\pi}_q = \frac{1}{n} \sum_{i=1}^{n} \tau_{iq}.$$

(Ne pas oublier la contrainte $\sum_q \hat{\pi}_q = 1$). Pour ce qui est de la maximisation en les $\gamma_{q\ell}$, cela dépend de la famille de lois $F(\cdot; \gamma)$ que l'on considère. Prenons le cas simple d'un graphe binaire (non dirigé) où $F(\cdot; \gamma)$ est une loi de Bernoulli de paramètre α. Alors on doit maximiser par rapport aux $\alpha_{q\ell}$ la quantité,

$$\begin{aligned} &\sum_{1 \leq q, \ell \leq Q} \sum_{i<j} \tau_{iq} \tau_{j\ell} \log[\alpha_{q\ell}^{A_{ij}} (1 - \alpha_{q\ell})^{1 - A_{ij}}] \\ &= \sum_{1 \leq q, \ell \leq Q} \sum_{i<j} \tau_{iq} \tau_{j\ell} \Big[A_{ij} \log \alpha_{q\ell} + (1 - A_{ij}) \log(1 - \alpha_{q\ell}) \Big]. \end{aligned}$$

La solution s'obtient simplement avec

$$\hat{\alpha}_{q\ell} = \frac{\sum_{i \neq j} \tau_{iq} \tau_{j\ell} A_{ij}}{\sum_{i \neq j} \tau_{iq} \tau_{j\ell}}.$$

Il s'agit de la fréquence moyenne des arêtes entre les groupes q, ℓ. En fait, puisque chaque τ_{iq} estime la probabilité que le nœud i appartienne au groupe q, on estime les paramètres d'interaction γ_{ql} en utilisant les interactions A_{ij} pondérées par le poids $\tau_{iq}\tau_{j\ell}$. Par exemple si on veut estimer la valeur moyenne de la loi conditionnelle $G(\cdot;\eta_{q\ell})$, notée $m_{q\ell}$ on trouvera en cherchant les points critiques de la quantité à maximiser

$$\hat{m}_{q\ell} = \frac{\sum_{i\neq j} \tau_{iq}\tau_{j\ell}A_{ij}}{\sum_{i\neq j} \tau_{iq}\tau_{j\ell}1_{A_{ij}\neq 0}}.$$

(Attention ici pour estimer la moyenne de la loi conditionnelle $G(\cdot;\eta_{q\ell})$, on ne prend en compte que les arêtes présentes, c'est-à-dire $A_{ij} \neq 0$).

6.3.4 Sélection de modèles

La plupart du temps le nombre de classes Q est inconnu et doit être estimé à partir des données. À partir de l'algorithme `VEM`, on peut utiliser le critère ICL (integrated classification likelihood). C'est un critère pénalisé, analogue du BIC mais au lieu de prendre la log-vraisemblance des observations (qui est inconnue ici) on utilise l'espérance de log-vraisemblance des données complètes sous l'approximation variationelle. Ainsi, pour chaque valeur du nombre de groupes Q, on obtient via l'algorithme `VEM` ajusté avec Q groupes, la quantité

$$\mathbb{E}_{\hat{\mathbb{Q}}}(\log \mathbb{P}(\mathbf{A}, \mathbf{Z}; \hat{\boldsymbol{\theta}})) = \sum_{q=1}^{Q}\sum_{i=1}^{n} \hat{\tau}_{iq}\log(\hat{\pi}_q) + \sum_{1\leq q,\ell\leq Q}\sum_{i,j} \hat{\tau}_{iq}\hat{\tau}_{j\ell}\log F(A_{ij};\hat{\gamma}_{q\ell}).$$

Ici, $\hat{\mathbb{Q}}, \hat{\boldsymbol{\theta}}$ sont les quantités obtenues à la fin des itérations de `VEM` (ou plus précisément à la fin de la meilleure itération de `VEM` quand on a fait plusieurs initialisations, ce qui est recommandé).

Là encore, l'expression de la pénalité va dépendre du choix de la famille de lois $F(\cdot;\gamma)$ que l'on considère. La forme générale du critère est

$$ICL(Q) := \mathbb{E}_{\hat{\mathbb{Q}}}(\log \mathbb{P}(\mathbf{A}, \mathbf{Z}; \hat{\boldsymbol{\theta}})) - \frac{1}{2}(Q-1)\log n - \frac{1}{2}\dim(\boldsymbol{\gamma})\log\frac{n(n-1)}{2},$$

où $\dim(\boldsymbol{\gamma})$ est la dimension du paramètre $\boldsymbol{\gamma} = (\boldsymbol{\alpha}, \boldsymbol{\eta})$ et $Q-1$ correspond à la dimension du paramètre $\boldsymbol{\pi}$.

Par exemple, dans le cas d'un graphe binaire non dirigé, on a $\boldsymbol{\gamma} = (\alpha_{ql})_{q,\ell}$ est de dimension $Q(Q+1)/2$ et Q^2 dans le cas dirigé. Si $F(\cdot;\gamma)$ est le mélange entre une Dirac en 0 et une loi de Poisson (tronquée en 0), de paramètre η, on obtient $\boldsymbol{\gamma} = (\alpha_{q\ell}, \eta_{q\ell})_{q,\ell}$ qui est de dimension $Q(Q+1)$ dans le cas non dirigé. Si par souci de parcimonie on a imposé que les paramètres de densité du graphe $\alpha_{q\ell}$ sont constants

pour tous les groupes q, ℓ alors on a $\boldsymbol{\gamma} = (\alpha; (\eta_{q\ell})_{q,\ell})$ qui est de dimension $1 + Q(Q+1)/2$ dans le cas non dirigé encore.

Noter que dans l'expression de l'ICL, le premier terme de pénalité $1/2(Q-1)\log n$ pénalise pour le paramètre $\boldsymbol{\pi} = (\pi_q)_{1 \leq q \leq Q}$ (de dimension $Q-1$) et qui porte sur n variables $Z_1, \ldots, Z_n$; tandis que le second terme vient pénaliser le paramètre d'interaction $\boldsymbol{\gamma}$ et se fonde lui sur $n(n-1)/2$ observations, à savoir les $\{A_{i,j}\}_{i<j}$.

Finalement, on va sélectionner le nombre de groupes Q en se fixant une borne $Q_{\max}$ et en utilisant

$$\hat{Q} = \underset{1 \leq Q \leq Q_{\max}}{\text{Argmax}}\ ICL(Q).$$

Aucun résultat théorique n'existe sur les propriétés asymptotiques de ce critère, mais ses performances empiriques sont très bonnes.

Chapitre 7

Plongement de graphes (Graphs embedding)

Les techniques de plongement (*embedding*) sont une classe d'approches génériques qui consistent à prendre des objets *complexes* et à les plonger dans un espace vectoriel (de dimension appropriée). On parle parfois de **vectorisation** des objets. L'intérêt de tels plongements est qu'une fois les objets résumés dans un espace vectoriel, on peut avoir recours à toutes les méthodes classiques de l'analyse statistique multivariée pour les manipuler. Bien sûr, pour que cela ait un intérêt, il faut que l'application qui plonge l'object complexe dans l'espace vectoriel capture suffisamment les propriétés de cet objet.

Un bon plongement n'est donc pas universel : tout dépend de l'objectif final qui se pose, *i.e.* qu'est-ce que vous voulez faire avec vos objects complexes une fois qu'ils sont dans un espace vectoriel ? Est-ce que vous voulez comparer ces objets deux à deux ? Est-ce que vous voulez faire des groupes, ie du clustering de ces objets ? Est-ce que vous voulez prédire des groupes ? etc

On distingue parfois les méthodes **transductives** et les méthodes **inductives** : dans le premier cas, si un nouvel objet arrive, on doit refaire tout le plongement, tandis que dans le second, on peut plonger le nouvel objet sans recalculer le plongement.

La question du choix de la dimension de l'espace vectoriel d'arrivée est assez peu abordée dans la littérature ; elle dépend essentiellement de la méthode utilisée.

Pour finir, mentionnons que les plongements sont bien sûr intimement liés à la construction de noyaux pour les support vector machines (SVMs). Dans la théorie des SVMs, on cherche des fonctions noyaux pour mesurer la dissimilarité entre des objets complexes, et de sorte que cette dissimilarité s'exprime en fait comme le produit scalaire entre les images de ces objets dans un certain espace vectoriel.

Attention, le même terme « graph embedding » est utilisé pour désigner (au moins) deux choses pourtant très distinctes :

- **Nodes graph embedding** ou plongement des nœuds d'un graphe dans un espace vectoriel : c'est un processus qui permet de représenter **les sommets (ou nœuds) d'un graphe dans un espace vectoriel** de façon à conserver des propriétés topologiques du graphe dans la représentation vectorielle. On considère donc a priori un seul graphe de n nœuds et ce sont les n nœuds qui sont envoyés dans un espace vectoriel et deviennent n points de cet espace.
- **Whole-graph embedding** ou plongement d'un graphe dans un espace vectoriel : ici on a un ensemble de m graphes (chacun ayant un nombre de nœuds $n_1, \ldots, n_m$ potentiellement distincts), et **chacun de ces graphes va être envoyé dans un espace vectoriel**, on obtient ainsi m points dans cet espace qui représentent nos m graphes de départ.

Dans le cas des graphes, les tâches d'intérêt peuvent être : le clustering des nœuds (nodes graph embedding) ou des graphes eux-mêmes (whole graph embedding), la prédiction de liens, la détection de motifs, la détection d'anomalies,...

On va décrire dans ce chapitre quelques unes des (très nombreuses) méthodes de plongement de nœuds d'un graphe et de graphes entiers.

7.1 Méthodes de plongement des nœuds d'un graphe

Dans cette partie on considère un seul graphe et on souhaite plonger les n nœuds dans un espace vectoriel en tenant compte de la topologie de ce graphe.

7.1.1 Plongement via les vecteurs propres d'un Laplacien

Les approches vues au chapitre 5 sont fondées sur des idées de plongement ! En effet, la façon la plus courante de plonger les nœuds d'un graphe dans un espace vectoriel est d'utiliser les vecteurs propres d'une matrice laplacienne de ce graphe.

Dans le cadre du chapitre 5, les K vecteurs propres principaux (*i.e.* associés aux plus grandes valeurs propres en valeur absolue) formaient une matrice U de taille $n \times K$. Cette matrice, vue en lignes, est une matrice de n points dans $\mathbb{R}^K$ qui sont les images des n nœuds de départ dans l'espace vectoriel $\mathbb{R}^K$.

On a vu que ce plongement permet de conserver les propriétés de communautés au sein du graphe : des nœuds qui appartiennent à la même communauté (*i.e.* groupe de nœuds très connectés entre eux et peu connectés au reste des nœuds) ont

des images proches dans $\mathbb{R}^K$, ce qui permet à l'algorithme k-means de facilement retrouver ces communautés.

Ces méthodes ne peuvent traiter que des graphes non dirigés mais possiblement valués. Elles sont transductives. Des variantes existent, fondées sur des factorisations de matrice.

7.1.2 Plongement via des marches aléatoires sur le graphe

Marche aléatoire sur un graphe. On construit une marche aléatoire sur un graphe de la façon suivante : on tire un nœud au hasard (uniformément), puis à partir de ce nœud on sélectionne un de ses voisins au hasard (uniformément) et on "emprunte" l'arête qui va du nœud initial au nœud sélectionné. Puis on itère ce processus k fois. On obtient alors un chemin de longueur k dans le graphe.

Les marches aléatoires construites sur un graphe ont la propriété suivante : intuitivement, on comprend que si le graphe est structuré en communautés, alors la marche visite plusieurs fois les nœuds d'une même communauté puis éventuellement "sort" et va visiter une autre communauté. Quand on regarde la proportion de temps passé dans les nœuds on identifie les communautés. Les marches aléatoires sont donc communément utilisées pour faire de la détection de communautés dans des graphes.

DeepWalk. L'algorithme DeepWalk (Perozzi et al., 2014) utilise des processus de marches aléatoires courtes sur le graphe pour générer un grand nombre de petites séquences de nœuds. Celles-ci sont ensuite utilisées pour entraîner un modèle de langage : chaque suite de nœuds est traitée comme une suite de mots dans une phrase, et on fait alors appel aux algorithmes de plongement connus sur les mots de textes.

DeepWalk utilise donc les marches aléatoires pour générer des contextes de voisinage pour chaque nœud dans un graphe. Il considère chaque séquence de nœuds visités lors d'une marche aléatoire comme une phrase dans un corpus de textes. Il utilise ensuite un modèle de traitement du langage naturel appelé Skip-Gram pour apprendre une représentation vectorielle des nœuds du graphe.

Le modèle Skip-Gram est utilisé pour prédire le prochain mot dans une phrase, en maximisant la probabilité de co-occurrence des mots qui apparaissent dans une fenêtre de taille donnée (avant et après ce mot, voir figure 7.1).

En ce sens, DeepWalk est similaire à la méthode Word2Vec pour traiter les mots dans un corpus de textes. La différence fondamentale est que DeepWalk utilise les contextes de voisinage de chaque nœud dans un graphe pour générer des vecteurs, tandis que Word2Vec utilise les contextes de voisinage des mots dans un corpus de

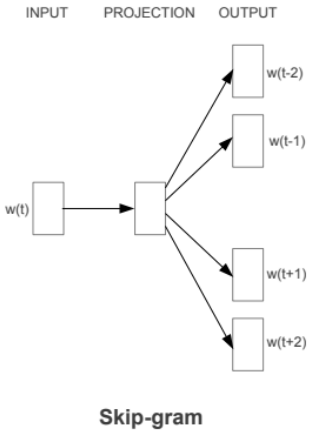

FIGURE 7.1 – Représentation du modèle SkipGram.

textes.

DeepWalk est un outil efficace pour la vectorisation des nœuds d'un grand graphe. Il n'est pas capable de traiter des graphes valués et c'est une méthode transductive.

Node2Vec. L'algorithme Node2Vec (Grover and Leskovec, 2016) est une amélioration de DeepWalk qui prend mieux en compte la structure de voisinage de chaque nœud dans un graphe. Pour cela, la marche aléatoire sur le graphe est *biaisée* : les voisins d'un nœud ne sont pas choisis uniformément au cours de la marche. Plus précisément, deux paramètres p, q vont modifier les probabilités de

- revenir immédiatement sur le nœud d'où on vient ; il s'agit de contrôler les sous-chemins de la forme $i \to j \to i$;
- aller dans un nœud voisin de celui d'où on vient ; après un mouvement $i \to j$, on traite différemment les voisins k de j s'ils sont également voisins de i ou pas.

Ces modifications sont illustrées sur la Figure 7.2.

Le paramètre p est lié à une exploration du graphe de type Breadth-First Search (BFS, c'est une exploration en largeur). Les voisinages explorés sont plus locaux et on cherche à apprendre des rôles structurels des nœuds. Le paramètre q est plutôt lié à une exploration du graphe de type Depth-First Search (DFS, c'est une exploration en profondeur). Il permet d'apprendre des variables plus globales et il tend à mettre l'accent sur les communautés. Les types d'exploration BFS et DFS sont illustrées sur la figure 7.3.

Le biais introduit sur la marche aléatoire permet de contrôler la probabilité

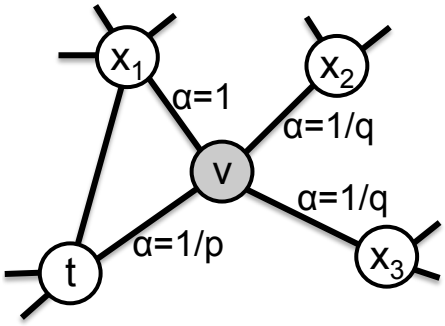

FIGURE 7.2 – Illustration du biais de marche aléatoire dans Node2Vec. La marche part du nœud t et visite le nœud v. À ce stade, la probabilité de visite de ses voisins n'est pas uniforme et diffère selon qu'on retourne en t, qu'on visite un voisin de t (ici x_1) ou qu'on visite un tout nouveau nœud (x_2 ou x_3). Source Grover and Leskovec (2016).

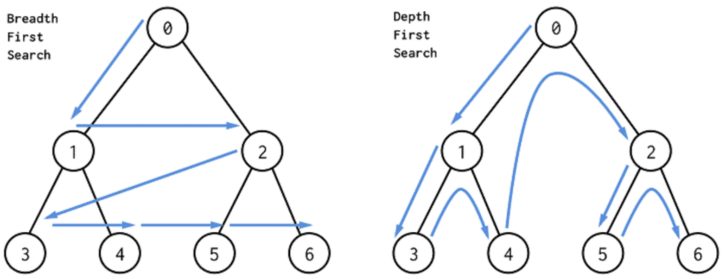

FIGURE 7.3 – Illustration des explorations BFS et DFS.

de transition entre les nœuds en fonction de leur distance dans le graphe. Cette technique permet de capturer la structure de voisinage de chaque nœud de façon plus fine que DeepWalk.

Noter enfin que Node2Vec peut prendre en compte des valuations (positives) du graphe : le poids de chaque arête est pris en compte avec les paramètres p, q qui modifient les probabilités d'explorer chaque nœud . Comme DeepWalk, c'est une méthode transductive.

7.1.3 Graph Convolutional Networks (GCN) et variational auto-encoders (VAE).

Les Graph Convolutional Networks (GCN) sont des réseaux de neurones, inspirés des Convolutional Neural Networks (CNN) qui sont utilisés en image. Les CNNs utilisent la structure de voisinage des pixels dans une image pour mieux apprendre cette donnée, via des "convolutions" c'est à dire l'application de fonctions filtres (multiplications et additions) sur la valeur d'un pixel et de ses voisins. De la même façon, les GNNs vont apprendre l'information encodée à un nœud en utilisant celle de ses voisins dans le graphe.

Il est fondamental de comprendre que le principe des GCN repose sur l'apprentissage à partir d'un graphe dont les nœuds sont munis de variables (nodes features). Tout comme la valeur d'un pixel est convolée avec celle de ses voisins sur une image, ici la valeur de la variable en un nœud va être convolée avec celle de ses voisins. **Pourtant, les GCN sont couramment utilisés sur des graphes sans valeurs a priori sur les variables**. Dans ce cas, un choix par défaut (dont l'impact n'est jamais discuté) est d'utiliser le degré du nœud comme valeur de la variable en ce nœud . Ce choix est naturel et fait du sens dans le test WL et le noyau WL qu'on décrira ci-dessous ; c'est pour ça qu'il a été adopté dans le présent contexte (mais sans prise de recul). Cette information des degrés est redondante avec la structure du voisinage.

Les variational auto-encoders (VAE) sont des réseaux de neurones composés de deux parties : un encoder qui envoie la donnée dans un espace latent, et un decoder qui renvoie la variable latente dans l'espace de la donnée d'origine. L'apprentissage du modèle fait en sorte que la donnée encodée puis décodée soit proche de la donnée d'origine.

La méthode `VGAE` pour variational graph auto-encoder (Kipf and Welling, 2016) utilise un réseau de neurones de type VAE, combiné avec des GCN, pour produire un embedding des nœuds du graphe. En pratique, l'auto-encoder (qui prend en entrée le graphe $A = (A_{ij})$ et apprend une représentation latente z_i de chaque nœud i) est

un GCN à deux couches ; le decoder est réalisé par un simple produit scalaire et une sigmoïde : $\mathbb{P}(A_{ij} = 1|z_i, z_j) = \sigma(z_i^\intercal z_j)$ où σ est la sigmoïde.

7.2 Méthodes de whole-graph embedding

Ici, nous allons donc nous intéresser au plongement d'un ensemble de graphes dans un espace vectoriel, avec des objectifs qui peuvent être variés (comparer ces graphes entre eux, faire du clustering de ces graphes - attention ce n'est pas la même chose que le clustering des nœuds d'un seul graphe vu précédemment !).

Un des enjeux majeurs ici est de pouvoir plonger dans un même espace (de dimension fixée et identique pour tous les graphes), des graphes potentiellement très différents (par exemple sur leur nombre de nœuds). Il faut réfléchir à ce facteur de l'ordre du graphe : veut-on qu'il soit déterminant (ou au contraire pas du tout) pour distinguer les graphes ? Autrement dit, veut-on un embedding où des graphes sont proches parce qu'ils ont des ordres similaires ou au contraire, parce que même s'ils ont des ordres très différents, certaines de leurs caractéristiques structurelles sont proches ? C'est un choix de modélisation qui dépend du jeux de données et de ce qu'on souhaite en faire.

7.2.1 À partir de statistiques descriptives

De nombreux auteurs ont proposé d'utiliser des statistiques descriptives de graphes (densité, coefficient de centralité, diamètre, ...), pour résumer l'information des graphes et les plonger dans un espace de dimension finie (le nombre de descripteurs).

7.2.2 À partir de motifs

Graphlets. Les graphlets sont des motifs (petits sous-graphes). On peut faire un plongement très simple de graphe à partir du vecteur de fréquences d'une liste fixée de motifs (en pratique d'ordres 3,4,5).

Le comptage du nombre d'occurrence de motifs dans un graphe peut vite devenir intensif en calcul (avec l'ordre du motif). De ce point de vue, les motifs du type sous-arbres (subtree patterns) sont intéressants car on peut explorer de façon plus efficace les sous-arbres d'un graphe. C'est l'idée qu'explore la méthode suivante.

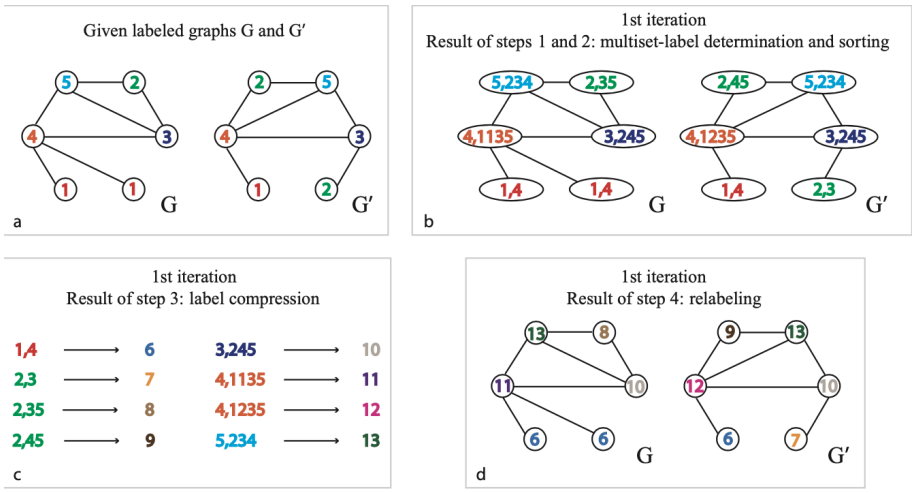

FIGURE 7.4 – Le test d'isomorphisme de 2 graphes de Weisfeiler-Lehman (source Shervashidze et al. (2011)).

Test WL d'isomorphisme. En théorie des graphes, le test de **Weisfeiler-Lehman** est un outil fondamental pour détecter si deux graphes sont isomorphes. C'est un algorithme itératif, dont l'idée est de comparer une suite de labels associée aux nœuds de chaque graphe. Le label d'un nœud est construit en explorant toujours plus profondément le voisinage de ce nœud . Tant que les labels sont équivalents, les 2 graphes peuvent être isomorphes ; et on s'arrête dès que les suites de labels sont distinctes (ou que l'exploration est finie et alors les graphes sont isomorphes). Ce processus est illustré à la figure 7.4. On débute avec deux graphes $G = (V, E, L)$ et $G' = (V', E', L')$ dont les nœuds ont des suites de labels $L := L_1 = (\ell_1(u))_{u \in V}$ et de même pour L'. S'il n'y a pas de labels, il suffit de prendre les degrés des nœuds comme labels de départ. On classe les labels dans l'ordre croissant pour former les suites, et on compare les deux suites : si elles diffèrent on s'arrête, sinon on continue. À l'étape t, on a deux graphes avec des suites de labels rangés dans l'ordre croissant $L_t = (\ell_t(u))_{u \in V}$ (et idem pour L'_t) sur les nœuds. En chaque nœud $u \in V$ (ou V'), on construit le nouveau label $\ell_{t+1}(u)$ en concaténant $\ell_t(u)$ avec les labels $\ell_t(v)$ pour tout v voisin de u dans G (idem pour G'). Les nouveaux labels sont re-numérotés de façons distinctes et on itère le processus.

Note : à l'origine, on pensait que le WL-test était une solution en temps polynomial du problème de test d'isomorphisme de 2 graphes, mais en fait ce n'est pas le cas (un contre-exemple a été exhibé ; le test est une condition nécéssaire mais pas suffisante pour l'isomorphisme de deux graphes) ; mais en pratique le test fonctionne pour presque toutes les paires de graphes (en un sens probabiliste).

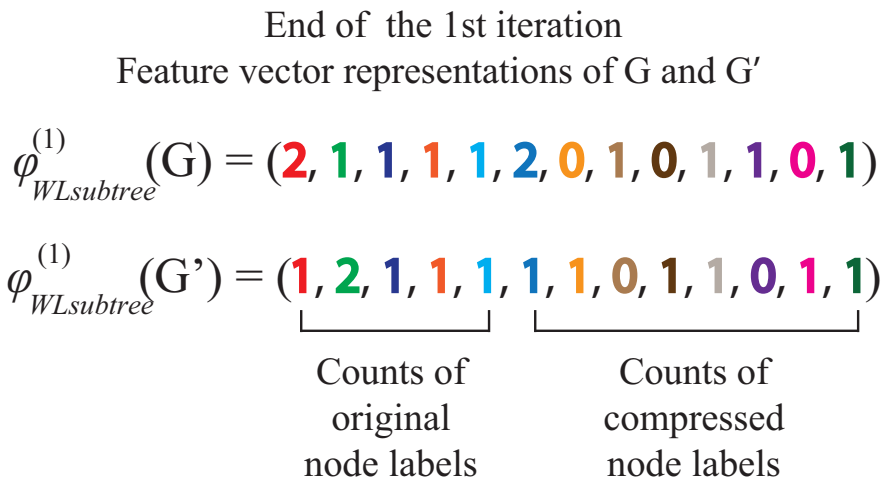

FIGURE 7.5 – Le plongement des deux graphes G, G' de la figure 7.4 pour la hauteur finale $h = 1$ (source Shervashidze et al. (2011)).

Noyau WL de graphes et plongement associé. À partir du test d'isomorphisme de WL, Shervashidze et al. (2011) ont proposé un *noyau* (au sens des support vector machine, SVM), ie une fonction de deux variables à valeurs réelles, pour comparer des graphes, via leur composition en sous-arbres. Plus précisément, on va créer un plongement d'un graphe (avec labels sur les noeuds, qui par défaut pourront être les degrés des nœuds) dans un espace vectoriel, à partir des comptages du nombre de nœuds dans chaque label, et ce à chaque "hauteur" ou itération du processus de WL test.

Ainsi, on se donne une hauteur finale $h \in \mathbb{N}$. Le test WL entre les deux graphes G, G' fabrique des suites de graphes aves labels sur les nœuds G_t avec suite de labels L_t et G'_t avec suite de labels L'_t pour toute valeur $1 \leq t \leq h$.

À partir de n'importe quel noyau k entre 2 graphes labellisés, Shervashidze et al. (2011) ont proposé de construire le noyau global entre G, G' comme la somme des noyaux entre chaque paire G_t, G'_t.

Pour préciser les choses et réaliser le plongement d'un graphe, , ils proposent d'utiliser le vecteur des comptages du nombre de nœuds de chaque type de label pour chaque hauteur t considérée ; voir figure 7.5.

L'intérêt de cette construction provient du fait qu'à chaque itération t, les nouveaux labels contiennent l'information des sous-arbres de profondeur t du graphe. Si deux labels sont identiques à ce stade, c'est que les deux noeuds ont les mêmes sous-arbres de profondeur t.

À t fixé, le vecteur des comptages du nombre de nœuds de chaque type de label

apparaissant à l'itération t fournit un plongement $\phi_t(G)$ de chaque graphe dans un espace vectoriel ; et par la même occasion cela définit un noyau via le produit scalaire

$$k_t(G_t, G'_t) =< \phi_t(G), \phi_t(G') >$$

Plus généralement, le noyau WL noté $k_{WL}^{(h)}$ de hauteur h est défini par

$$k_{WL}^{(h)}(G, G') = \sum_{t=1}^{h} k_t(G_t, G'_t).$$

Noter qu'ici on décide de regarder à chaque itération t le noyau qui compte les nœuds de chaque type de label, mais la construction s'applique pour d'autres graphes. Un des inconvénients du choix qui est proposé ici est que les ensembles de labels à chaque itération doivent être (les mêmes ou du moins) très proches. Par exemple à l'étape 1 pour des graphes sans labels, l'ensemble des labels est a priori l'ensemble des degrés possibles, donc les entiers $\{0, 1, \ldots, d_{max}\}$ où d_{max} est la plus grande valeur observée sur les deux graphes G, G'. Si cette valeur est très différente dans les deux graphes, le noyau (et le plongement) perdent en pertinence.

7.2.3 À partir de modèles génératifs

On peut réaliser un plongement de graphes par exemple avec le modèle à positions latentes de Hoff et al. (2002) dont on a déjà parlé au chapitre 6.

Plus généralement, si on a une classe de modèles probabilistes pour un ensemble de graphes, on peut inférer les paramètres du modèle pour chaque graphe, et résumer chaque graphe par son vecteur de paramètre.

Bibliographie

Achlioptas, D., A. Clauset, D. Kempe, and C. Moore (2009). On the bias of traceroute sampling : Or, power-law degree distributions in regular graphs. *J. ACM 56*(4), 1–28.

Albert, R. and A.-L. Barabási (2002, Jan). Statistical mechanics of complex networks. *Rev. Mod. Phys. 74*, 47–97.

Bahoken, F., L. Beauguitte, and S. Lhomme (2013). La visualisation des réseaux. Principes, enjeux et perspectives. preprint.

Berge, C. (1976). *Graphs and hypergraphs* (revised ed.). North-Holland Publishing Co., Amsterdam-London ; American Elsevier Publishing Co., Inc., New York. Translated from the French by Edward Minieka, North-Holland Mathematical Library, Vol. 6.

Bhagat, S., M. Burke, C. Diuk, I. O. Filiz, and S. Edunov (2016). Three and a half degrees of separation. facebook research blog `https://research.fb.com/three-and-a-half-degrees-of-separation/`.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Information Science and Statistics. Springer, New York.

Bollobás, B. (1985). *Random Graphs*. Academic, London.

Erdős, P. and T. Gallai (1961). Graphs with points of prescribed degree. (Graphen mit Punkten vorgeschriebenen Grades.). *Mat. Lapok 11*, 264–274.

Erdős, P. and A. Rényi (1959). On random graphs. I. *Publ. Math. Debrecen 6*, 290–297.

Fruchterman, T. M. J. and E. M. Reingold (1991). Graph drawing by force-directed placement. *Software : Practice and Experience 21*(11), 1129–1164.

Grover, A. and J. Leskovec (2016). Node2vec : Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, New York, NY, USA, pp. 855–864. Association for Computing Machinery.

Handcock, M., A. Raftery, and J. Tantrum (2007). Model-based clustering for social networks. *Journal of the Royal Statistical Society : Series A (Statistics in Society) 170*(2), 301–54.

Hoff, P., A. Raftery, and M. Handcock (2002). Latent space approaches to social network analysis. *J. Amer. Statist. Assoc. 97*(460), 1090–98.

Ji, S., W. Zhang, and R. Li (2014). A probabilistic latent semantic analysis model for coclustering the mouse brain atlas. *IEEE/ACM Transactions on Computational Biology and Bioinformatics 10*(6), 1460–1468.

Kamada, T. and S. Kawai (1989). An algorithm for drawing general undirected graphs. *Information Processing Letters 31*(1), 7–15.

Kipf, T. N. and M. Welling (2016). Variational graph auto-encoders. Technical report, ArXiv :1611.07308.

Kolaczyk, E. D. (2009). *Statistical Analysis of Network Data : Methods and Models*. Springer.

Kolaczyk, E. D. and G. Csárdi (2014). *Statistical analysis of network data with R*. Use R ! Springer, New York.

Lauritzen, S. L. (1996). *Graphical models*, Volume 17 of *Oxford Statistical Science Series*. The Clarendon Press, Oxford University Press, New York. Oxford Science Publications.

Lhomme, S. (2012). Les modèles de graphes théoriques. preprint.

Luke, D. A. (2015). *A User's Guide to Network Analysis in R*. Springer.

Milgram, S. (1967). The small world problem. *Psychology Today 1*(60), –.

Ng, A. Y., M. I. Jordan, and Y. Weiss (2001). On spectral clustering : Analysis and an algorithm. In *Advances in neural information processing systems*, pp. 849–856. MIT Press.

Penrose, M. (2003). *Random Geometric Graphs*. Oxford University Press.

Perozzi, B., R. Al-Rfou, and S. Skiena (2014). Deepwalk : Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, New York, NY, USA, pp. 701–710. Association for Computing Machinery.

Rohe, K., S. Chatterjee, and B. Yu (2011). Spectral clustering and the high-dimensional stochastic blockmodel. *Annals of Statistics 39*(4), 1878–1915.

Shervashidze, N., P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt (2011). Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research 12*(77), 2539–2561.

Tufte, E. R. (2001). *The Visual Display of Quantitative Information.* Cheshire, CT : Graphics Press.

von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing 17*(4), 395–416.

Watts, D. and S. Strogatz (1998). Collective dynamics of 'small-world' networks. *Nature 393*(6684), 440–442.