Master Statistique et Économétrie
Université Toulouse 1 Capitole
2 Rue du Doyen-Gabriel-Marty
31042 Toulouse

Unité MIAT
INRA
24 Chemin de Borde-Rouge
31326 Castanet-Tolosan Cedex

Intégration de données pour la classification non supervisée de données biologiques

Florian Brunet

Stage de fin d'études - Master Statistique & Économétrie
2012/2013

Tutrices de stage : Christine Cierco-Ayrolles & Nathalie Villa-Vialaneix
Tuteur pédagogique : Christine Thomas-Agnan
Période de stage : Du 8 avril 2013 au 27 septembre 2013

Remerciements

Je tiens à remercier mes co-encadrantes, Nathalie Villa-Vialaneix et Christine Cierco-Ayrolles, pour leur disponibilité et l'aide qu'elles ont pu m'apporter au cours de mon stage.

Je remercie également Régis Sabbadin et Fabienne Ayrignac pour m'avoir accueilli au sein de l'unité MIAT.

Table des matières

Résumé :

Ce stage de fin d'études de 6 mois, du 8 avril au 27 septembre 2013, a été effectué au sein de l'INRA de Toulouse, un organisme public qui a pour but la recherche agronomique. Plus précisément, j'ai intégré l'unité Mathématiques et Informatique Appliquées de Toulouse (UR 875) dont la mission est de mettre à la disposition de l'INRA des méthodes et des compétences en mathématiques et en informatique appliquées, en particulier dans le cadre de collaborations avec les autres départements.

L'objectif du stage était de mettre en place une approche de classification non supervisée à noyaux multiples afin de détecter des micro-ARNs. Ce travail a été effectué en utilisant le logiciel libre R : des scripts génériques et des fonctions ont été développés dans le but de pouvoir être utilisés sur des problématiques similaires. Différents algorithmes de classification ont ainsi été développés et comparés, sur des données simulées avant d'être appliqués aux données réelles et à la problématique de détection des micro-ARNs.

Ce travail a permis d'extraire des micro-ARNs potentiels qui pourront ainsi être étudiés plus en détails par les biologistes. Une partie du travail a par ailleurs été intégrée dans un article [4] qui sera présenté lors de la conférence MARAMI[1], se déroulant du 16 au 18 octobre 2013 à Saint-Étienne.

Ce stage m'a permis de m'initier aux missions de recherche appliquée, par la mise en place d'une méthodologie d'apprentissage statistique, de développer mon sens de la rigueur, dans les scripts ou les compte-rendus de mon avancée, mais aussi d'apprendre à présenter oralement mes travaux, comme je l'ai fait devant les membres de l'unité.

[1]Modèles et Analyse des Réseaux : Approches Mathématiques et Informatique, http://lipn.univ-paris13.fr/marami2013/MARAMI13/Accueil.html

Abstract:

This six-month internship took place from April, 8$^{\text{th}}$ to September, 27$^{\text{th}}$ at the INRA (French national institute for agricultural research). More precisely, I joined the department of applied mathematics and computer science of Toulouse (UR 875), whose aim is to provide some skills and mathematical or computer tools, in particular in projects realized with other departments.

The objective of this internship was to develop an unsupervised approach, based on kernels, to detect micro-RNAs. This work was done using the free program R: scripts and generic functions were developed for solving this problem as well as other similar problems. Differents algorithms were compared on simulated datasets and then applied to real data and to the problem of mi-RNAs detection.

This work permised us to extract potential mi-RNAs, on which biologists can now focused. Some of the obtained results are presented in an article published in the proceedings of the conference MARAMI[2], which will take place in Saint-Etienne from October 16^{th} to 18^{th} 2013.

This internship allowed me to discover an applied research framework by constructing a statistical methodology about unsupervised learning. It also helped me to become more thorough, about development of scripts or presentation of my work, and to be able to present my results, as I did in a presentation to the members of my INRA department.

[2] http://lipn.univ-paris13.fr/marami2013/MARAMI13/Accueil.html

Première partie .

Introduction

Après avoir effectué mes précédents stages dans des secteurs variés, j'ai souhaité conclure ma formation par une mission orientée vers la recherche appliquée en statistique. J'ai ainsi intégré l'INRA et plus précisément l'unité *Mathématiques et Informatique Appliquées de Toulouse* où j'ai été retenu afin de travailler sur des méthodes de classification non supervisée.

Ce stage présente un double intérêt en répondant aux attentes de différentes communautés. Pour un public de scientifiques ou de statisticiens, il met en place une méthodologie et propose des algorithmes afin d'effectuer des tâches de classifications non-supervisées, basées sur le principe des k-means, et permettant d'intégrer différents types d'informations (graphe, numérique, catégorielle, textuelle, etc). Pour cela, nous utiliserons une approche combinant des noyaux (voir section 4.3.1) et ce, en veillant à obtenir des résultats à la fois stables et pertinents, à partir des différentes sources d'informations disponibles.

Pour un public de biologistes, l'utilisation de cette méthodologie sur leurs données devrait permettre de détecter des micro-ARNs. Ces derniers, impliqués dans un grand nombre de fonctions physiologiques essentielles, sont fortement étudiés par les biologistes. En effet, ces molécules ont un rôle de régulation des gènes, notamment durant le développement embryonnaire et sont aussi impliquées dans diverses maladies comme le cancer ou les maladies cardio-vasculaires. Il est donc important de pouvoir identifier et caractériser de nouveaux micro-ARNs afin de pouvoir étudier de manière plus précise leur implication dans le fonctionnement du vivant.

Dans la suite de ce rapport, je présenterai dans un premier temps ma structure d'accueil dans la section II. Ensuite je présenterai la méthodologie statistique mise en place dans la section 4 puis je présenterai et comparerai les résultats obtenus entre différentes approches mises en place et ce, sur des jeux de données simulées en 6.1. Enfin, j'appliquerai les algorithmes développés sur les données que l'on m'a confiées afin de répondre à la problématique de détection de micro-ARNs en section 6.2.

Deuxième partie .

Description de l’entreprise d’accueil

1. L'INRA

L'INRA[1], Institut National de la Recherche Agronomique, est un Établissement Public à caractère Scientifique et Technologique (EPST, au même titre que le CNRS ou que l'INSERM), essentiellement financé par des fonds publics (il rend compte de son activité et de sa gestion à ses ministères de tutelle, le ministère de l'Enseignement supérieur et de la Recherche et le ministère de l'Alimentation, de l'Agriculture et de la Pêche).

Lors de sa fondation en 1946, sa mission était d'accompagner les changements du monde agricole, mais aussi des filières alimentaires afin de répondre aux attentes de la société, par exemple dans le domaine de la suffisance alimentaire. Depuis sa création, les prérogatives de l'INRA ont toutefois évolué. Dans un contexte climatique, démographique et énergétique complexe, la recherche agronomique doit étudier des enjeux majeurs à des échelles variées. Imaginer la disponibilité et la sécurité alimentaire mondiale en 2050, contribuer à la limitation du gaz à effet de serre d'origine agricole, favoriser l'adaptation de l'agriculture et des forêts au changement climatique non réversible sont autant de préoccupations mondialement partagées. Elles impliquent, entre autres, de connaître les comportements des individus à l'échelle des territoires ou des marchés, d'étudier les liens entre la santé des plantes, des animaux et des hommes, de rechercher de nouvelles voies pour la production d'énergie et de matériaux issus de l'agriculture et d'en limiter en général l'impact environnemental...

Pour cela, l'Institut national de la recherche agronomique (Inra) produit des connaissances scientifiques et accompagne l'innovation économique et sociale dans les domaines de l'alimentation, de l'agriculture et de l'environnement.

Partout dans le monde, par sa capacité d'impulsion, de coordination, de diffusion et de transfert, l'établissement évolue au contact d'une grande diversité d'acteurs : académiques, économiques, associatifs ou territoriaux. Tous participent à la définition des orientations de l'Institut. Ces ancrages à la société font de l'Inra un organisme de recherche "finalisée" qui mobilise nombreuses disciplines scientifiques : principalement les sciences de la vie (68 % des compétences scientifiques de l'Institut), mais aussi les sciences des milieux et des procédés (12 %), l'ingénierie écologique, les écotechnologies et les biotechnologies (8 %), ainsi que les sciences économiques et sociales (8 %) et les sciences du numérique (4 %).

Pour remplir ces objectifs, l'INRA peut s'appuyer sur environ 400 unités de recherche réparties dans 17 centres (localisations) et 14 départements (grandes thématiques) de recherche (une unité correspond à un centre et à un ou plusieurs départements de rattachement). Les unités expérimentales de l'INRA couvrent environ 12 000 hectares dont 3 000 hectares de forêts. Parmi le cheptel de l'INRA, on peut compter environ 6000 bovins, 16 000 ovins, 8 000 porcins, 300 équins, 34 000 volailles, une centaine de cervidés et une dizaine de lamas. L'organigramme de l'INRA est fourni dans la figure 1.

[1] http://www.inra.fr


CONSEIL D'ADMINISTRATION
Président François Houllier
CONSEIL SCIENTIFIQUE
Président Frédéric Dardel
COMITÉ D'ÉTHIQUE
Président Louis Schweitzer
DIRECTIONS D'APPUI À LA RECHERCHE
Philippe Chemineau
Directeur de l'Action régionale, de l'Enseignement supérieur et de l'Europe
François Chambelin
Directeur du Financement et de l'Administration générale
Fabrice Marty
Directeur des Ressources humaines
Jean-Michel Perez-Cano
Directeur du Système d'information
Gérard Jacquin
Directeur de la Valorisation
COLLÈGE DE DIRECTION
PRÉSIDENCE - DIRECTION GÉNÉRALE
François Houllier
Président-Directeur général
Olivier Le Gall
Directeur général délégué aux Affaires scientifiques
Claude Ronceray
Directeur général délégué à l'Appui à la recherche
DIRECTEURS SCIENTIFIQUES
Christine Cherbut
Directrice scientifique Alimentation
Hervé Guyomard
Directeur scientifique Agriculture
Jean-François Soussana
Directeur scientifique Environnement
Jean-François Launay
Directeur de cabinet et de la communication
Françoise Vedele
Conseillère scientifique
Pierre-Henri Duée
Mission d'audit interne et d'inspection
Bertrand Schmitt
Délégué à l'Expertise scientifique collective, à la Prospective et aux Études
Christine Charlot
Secrétaire générale du collège de direction
Mission de coordination des actions avec les partenaires agricoles (par intérim)
Jean-Luc Pujol
Mission d'anticipation recherche/société et développement durable
Robert Habib
Délégué aux Affaires internationales
Gilles Aumont
Délégué aux Infrastructures scientifiques collectives
Edith Legouy
Déléguée à l'Évaluation
Paul Colonna
Délégué scientifique au Développement durable
Robert Thivet
Administrateur du siège
Philippe Martin
Agent comptable principal
Pierre Paris
Mission de coordination des services décentrés d'appui à la recherche
CHEFS DE DÉPARTEMENTS
Jean Dallongeville
Alimentation humaine - ALIMH
Carole Caranta
Biologie et amélioration des plantes - BAP
Monique Axelos
Caractérisation et élaboration des produits issus de l'agriculture - CEPIA
Thierry Caquet
Écologie des forêts, prairies et milieux aquatiques - EFPA
Guy Richard
Environnement et agronomie - EA
Denis Milan
Génétique animale - GA
Frédérick Garcia
Mathématiques et informatique appliquées - MIA
Emmanuelle Maguin
Microbiologie et chaîne alimentaire - MICA
Benoit Malpaux
Physiologie animale et systèmes d'élevage - PHASE
Thierry Pineau
Santé animale - SA
Didier Andrivon et Xavier Reboud
(par intérim)
Santé des plantes et environnement - SPE
Benoit Dedieu
Sciences pour l'action et le développement - SAD
Alban Thomas
Sciences sociales, agriculture et alimentation, espace et environnement - SAE2
PRÉSIDENTS DE CENTRES ET DÉLÉGUÉS RÉGIONAUX
Henri Seegers
Président du centre Angers-Nantes
Harry Ozier-Lafontaine
Président du centre Antilles-Guyane
Hubert De Rochambeau
Président du centre Bordeaux-Aquitaine
Jean-Baptiste Coulon
Président du centre Clermont-Ferrand - Theix - Lyon
Frédérique Pelsy
Présidente du centre Colmar
François Casabianca
Président du centre Corse
Patrick Etievant
Président du centre Dijon
Mariane Lefort
Déléguée régionale Île-de-France
Muriel Mambrini-Doudet
Présidente du centre Jouy-en-Josas
Gilles Gandemer
Président du centre Lille
Laurent Bruckler
Président du centre Montpellier
Erwin Dreyer
Président du centre Nancy
Jean-Marc Chabosseau
Président du centre Poitou-Charentes
Michel Bariteau
Président du centre Provence-Alpes-Côte-d'Azur
Patrick Herpin
Président du centre Rennes
Yvonne Couteaudier
Déléguée régionale Rhône-Alpes
Michèle Marin
Présidente du centre Toulouse Midi-Pyrénées
Catherine Beaumont
Présidente du centre Val de Loire
Laurent Hémidy
Président du centre Versailles-Grignon

2. Le centre de Toulouse

Le centre INRA de Toulouse Midi-Pyrénées est un lieu d'activités scientifiques pluridisciplinaires, partagées avec un partenariat académique dense et diversifié (plus de 400 titulaires non INRA présents dans les 12 unités mixtes de recherche). Il présente la particularité d'accueillir des unités de tous les départements scientifiques de l'INRA.

Outre les collaborations ciblées avec le CNRS et l'émergence d'activités avec l'INSERM, l'INRA est associé aux universités (UT1 Capitole, UT3 Paul Sabatier), à l'école vétérinaire (INP-ENVT) et à des écoles d'ingénieurs (INP-ENSAT, INP-EIP, INP-ENSIACET, INSA). Les activités de recherche du centre sont adossées à un large socle d'unités expérimentales (grandes cultures, élevages ovins et cunicole). Membre actif du Groupement d'Intérêt Scientifique (GIS) GENOTOUL, l'INRA participe au développement des infrastructures de recherche collectives en sciences du vivant de la génomique à la bioinformatique. L'INRA est fondateur du pôle de compétences en sciences agronomiques et vétérinaires « TOULOUSE AGRI CAMPUS » et il est un acteur du pôle de compétitivité sur les agro-chaînes « Agrimip Sud-Ouest Innovation ».

Fort de la représentation d'un large éventail de disciplines, les équipes du centre INRA Toulouse Midi-Pyrénées privilégient des activités de recherche et d'innovation en réponse à trois grands enjeux :

1. des systèmes de production agricoles (végétaux, animaux) et forestiers plus durables et adaptés au changement climatique,
2. une alimentation attentive aux questions de santé,
3. de nouvelles filières de transformation des agro-ressources en faveur d'une valorisation du carbone renouvelable.

Ces recherches transdisciplinaires se déclinent en 7 axes majeurs :

1. Génétique et biologie animale intégrative, maladies animales infectieuses et santé publique, conception des systémes d'élevage durable
2. Biologie intégrative des interactions plantes-environnement
3. Nutrition et prévention : toxicologie, biomarqueurs
4. Biotechnologies industrielles
5. Méthode et plate-forme pour la biologie intégrative animale, végétale et microbienne
6. Agro-écologie dans les territoires agricoles et forestiers
7. Économie de l'environnement et des marchés

Le centre INRA Toulouse Midi-Pyrénées fait partie des 5 plus grands centres INRA (en terme d'effectifs), présent dans un environnement académique important. Il est le pôle de référence en toxicologie alimentaire mais également pour les recherches sur le tournesol et le lapin. Le centre continue de maintenir le niveau d'excellence de ses plateformes technologiques avec le renforcement de la génomique, des biotechnologies vertes et blanches, de la modélisation et avec la construction d'un data center.

3. L'Unité MIAT

L'Unité de Mathématiques et Informatique Appliquées de Toulouse (MIAT, anciennement Unité de Biométrie et Intelligence Artificielle) est une unité propre (UR875) du département de Mathématiques et Informatique Appliquées (MIA) de l'INRA. Comme toutes les unités de MIA, l'unité MIAT a pour mission scientifique de développer et mettre en œuvre des méthodes mathématiques et/ou informatiques pertinentes pour résoudre des problémes identifiés avec des collaborateurs qui sont issus principalement d'autres départements de l'INRA. L'unité comporte actuellement (depuis janvier 2011) deux équipes de recherche (MAD et SaAB) et trois équipes de service (plateformes Bioinformatique du GIS Genotoul, RECORD et SIGENAE) :

- MAD (Modélisation des Agro-écosystémes et Décision)
- SaAB (Statistiques et Algorithmique pour la Biologie)
- plateforme Bioinformatique du GIS GENOTOUL - Génopole Toulouse Midi-Pyrénées
- RECORD (plateforme de modélisation et de simulation des agro-écosystèmes)
- SIGENAE (plateforme « Systèmes d'information des génomes des animaux d'élevage »)

Troisième partie .

Description du stage

4. Contexte et problématique du stage

4.1. Objectifs généraux

L'objectif de ce stage est de mettre en place une approche permettant de combiner l'information contenue dans un graphe (voir 4.2) à d'autres types de données (numériques, catégorielles, etc) pour réaliser des tâches de classification. Nous utiliserons pour cela une approche multi-noyaux pour de la classification non supervisée, un noyau étant une application permettant de définir une notion de distance entre individus, quel que soit le type de données caractérisant ces derniers. Les approches que nous avons mises en œuvre ont été testées et comparées sur différents jeux de données (simulées ou non) puis appliquées à un jeu de données recueilli par des biologistes de l'INRA de Toulouse.

Ce jeu de données, pour lequel est développée cette approche multi-noyaux, consiste en un réseau de co-expression de micro-ARNs[1] potentiels caractérisés par des variables numériques et catégorielles. En appliquant notre stratégie multi-noyaux à ce jeu de données, nous espérons permettre la détection de vrai micro-ARNs. En effet, la détection de ces vrais micro-ARNs, impliqués dans la régulation des gènes, est un travail relativement chronophage et difficile pour les biologistes. Pour cela, nous disposerons d'individus annotés, ce qui signifie que nous connaissons des vrais micro-ARNs (vrais positifs), cependant nous n'avons pas d'individus clairement identifiés comme n'étant pas des micro-ARNs (vrais négatifs).

4.2. Graphes, graphes étiquetés

Les graphes sont des outils mathématiques utilisés pour modéliser des données relationnelles, c'est-à-dire, des données dans lesquelles les individus ne sont pas décrits par un ensemble de variables numériques ou catégorielles mais par les relations qu'ils entretiennent entre eux. Ils sont ainsi très utilisés dans des domaines variés et sont parfois dénommés *réseaux* (réseau social, réseau informatique, etc). Dans notre cas, les graphes permettront de modéliser des relations de régulation entre micro-ARNs.

4.2.1. Notions élémentaires de théorie des graphes

Définition 1. *Un* ***graphe*** $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ *est constitué de deux ensembles : un ensemble de* ***sommets*** $\boldsymbol{V}$ *et un ensemble d'****arêtes*** $\boldsymbol{E}$ *qui est un sous-ensemble de l'ensemble des paires de sommets.*

Dans la suite, on ne considérera que des graphes simples (sans arête entre un sommet et lui-même), non orienté (les paires de sommets (u, v) et (v, u) sont équivalentes) et non pondérés (aucun poids n'est associé à une arête).

[1] Les notions biologiques nécessaires à la compréhension seront introduites et expliquées dans la section 6.2

Les graphes interviennent de manière naturelle pour modéliser de nombreuses situations de la vie réelle. Un exemple célèbre est issu du média social Facebook. Mon propre réseau Facebook est représenté, à titre d'illustration, en figure 4.1. Dans ce réseau,

- les sommets représentent mes « amis »,
- une arête représente un lien d'amitié entre deux de mes amis.

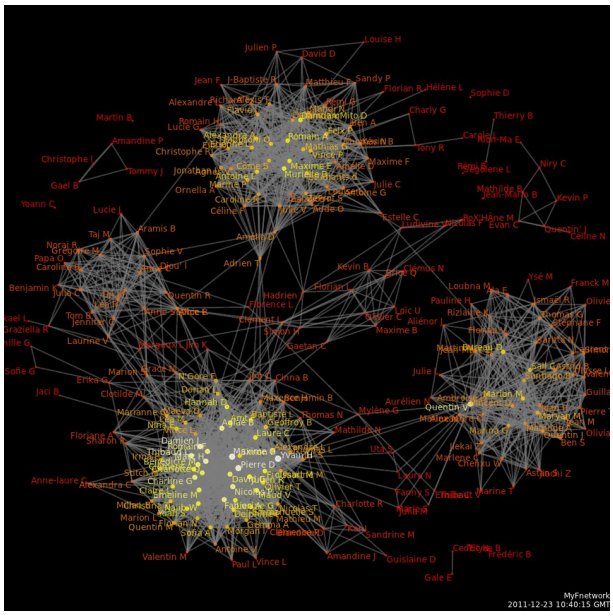


FIGURE 4.1.: Graphe d'un réseau Facebook

Parfois, des informations supplémentaires sont disponibles sur les sommets du graphe. On parle alors de *graphe étiqueté* :

Définition 2. *Un graphe est dit* ***étiqueté*** *si ses sommets sont caractérisés par un ensemble de mesures de variables pouvant être de différentes natures (textuelles, quantitatives, qualitatives, etc). Ces mesures sont appelées* ***étiquettes****.*

En revenant à l'exemple de mon réseau Facebook, on pourrait notamment caractériser les sommets par les étiquettes suivantes : âge, lieu d'habitation, nombre d'amis, couleur des yeux, etc. Dans la suite, la problématique sera de pouvoir effectuer des classifications des sommets étiquetés d'un graphe en tenant compte à la fois de la structure du graphe (retrouver des groupes de sommets fortement connectés) mais aussi de la « similarité des étiquettes ». Plusieurs auteurs ont déjà travaillé sur la classification non supervisée de ces graphes étiquetés. Certains effectuent une classification principalement basée sur les étiquettes des sommets qui est ensuite corrigée par un principe de seuillage basé sur les poids des arêtes entre sommets [27]. À l'inverse, d'autres auteurs favorisent la structure du graphe dans leur classification [6]. La méthode que nous allons mettre en place tend quant à elle à équilibrer les apports respectifs des différents types de données, car nous n'avons pas d'a priori sur la source d'information la plus importante. Nous utiliserons pour cela une approche multi-noyaux dont la méthodologie, la mise en œuvre et les résultats vont vous être présentés.

4.2.2. Les graphes en biologie

Dans le contexte biologique, on recherche des liaisons (de co-expression) entre un ensemble de gènes : dans ce problème, les gènes sont donc les nœuds du graphe. Cette liaison sera mesurée à l'aide d'une corrélation à partir de mesure sur plusieurs individus de l'expression de chaque gène. Ces corrélations sont la base pour définir des *réseaux de co-expression génique*. Dans de tels réseaux, on représente chaque gène par un nœud, et chaque corrélation directe et significative par une arête entre les deux nœuds concernés. On entend par **significative** le fait que cette corrélation dépasse un certain seuil, éventuellement déterminé par un test statistique, et par **directe** le fait qu'elle s'exprime indépendamment des autres gènes.

4.3. Intégration de données par classification à noyau

La classification est une procédure consistant à diviser un groupe d'individus en plusieurs sous-groupes qui sont composés d'individus aux caractéristiques similaires et de telle manière que les individus de sous-groupes distincts soient les plus différents possibles.

Afin de mesurer la proximité entre les individus, on dispose dans la majorité des cas de données numériques ou qualitatives, nous permettant de calculer une distance pour chacun des couples. On utilise notamment la distance euclidienne pour des données numériques et la distance du χ^2 [2] pour des variables qualitatives. Ensuite, différents algorithmes permettent de regrouper ces individus en fonction de critères de qualité. Par exemple, la classification hiérarchique ou l'algorithme des k-moyennes en sont des exemples bien connus. Si ces méthodes sont relativement simples à mettre en œuvre et couramment utilisées, elles peuvent cependant soulever certains problèmes. Lorsque les variables décrivant les individus ne sont pas numériques, ou bien constituent des groupes de variables de natures diverses (numériques, textuelles, catégorielles, relationnelles...), il n'y a pas de « distance » naturelle entre individus. C'est dans cet objectif de combiner différentes sources d'informations, certaines non numériques, que j'ai été amené à travailler sur les méthodes à noyaux.

4.3.1. Algorithme de k-moyennes à noyau

Comme indiqué précédemment, j'ai utilisé des méthodes à noyaux pour réaliser des tâches de classification et plus précisément l'algorithme de k-moyennes à noyau (appelé aussi « kernel k-means » en anglais). Dans un premier temps, je ferai un bref rappel sur la méthode des k-moyennes, ce qui nous permettra de mieux comprendre notre motivation à utiliser des méthodes à noyau. Je détaillerai ensuite le principe des méthodes à noyau.

Rappel sur l'algorithme des k-moyennes

L'algorithme des k-moyennes est un algorithme de partitionnement de données [25], permettant de partitionner des observations en C classes (ou *clusters*) dans lesquelles chaque observation appartient à la partition dont le centre de gravité est le plus proche. L'algo-

[2] Cette distance représente, d'une certaine manière, la distance entre les données observées et la loi théorique supposée. C'est une réalisation d'une variable aléatoire qui dérive d'une loi du χ^2 à (n-1) degrés de liberté.

rithme procède de manière itérative pour minimiser un critère de variance intra-classes. Il est toutefois nécessaire de spécifier à l'avance le nombre de classes C recherchées.

Par ailleurs, il existe différentes versions de cet algorithme, et notamment les versions *stochastiques* (dites aussi *on-line*) et *batch*. Dans la version *stochastique* de l'algorithme, une fois que les centres de classes sont définis, à chaque itération, on affecte **un seul** individu, tiré au hasard, à la classe dont le centre est le plus proche, puis on met à jour les centres. À l'inverse dans la version *batch*, on affecte à chaque fois **tous** les individus avant de mettre à jour les centres. Bien que la version *stochastique* donne de meilleurs résultats et soit moins sensible à l'initialisation [23], nous avons choisi d'implémenter la version *batch*, car c'est la méthode utilisée classiquement.

Pour des individus décrits par d variables numériques, l'apprentissage des classes se déroule comme décrit dans l'algorithme 1. Une illustration des différentes étapes de l'al-

Algorithm 1 Algorithme des k-moyennes

1: **require** Choix d'un nombre de classes C
2: **require** Données $(x_i)_{i=1,\dots,n}$, $x_i \in \mathbb{R}^d$
3: **initialization** Initialiser (aléatoirement) les centres des classes $\mu_1, \dots, \mu_C \in \mathbb{R}^d$
4: **repeat**
5: **Étape d'affectation** de chaque individu à la classe de centre le plus proche : $\forall\, i = 1 \dots, n$

$$\mathcal{C}(x_i) \leftarrow \mathcal{C}_l \text{ tel que } l = \arg\min_{k=1,\dots,C} d(x_i, \mu_k)$$

où $d(x_i, \mu_k) = \|x_i - \mu_k\|$ est distance euclidienne dans $\mathbb{R}^d$
6: **Étape de représentation** des centres μ_k de chaque classe : $\forall\, k = 1, \dots, C$,

$$\mu_k \leftarrow \frac{1}{N_k} \sum_{x_i \in \mathcal{C}_k} x_i \text{ avec } N_k = \text{Card}(\mathcal{C}_k)$$

7: **until** Convergence
8: **return** classification $(\mathcal{C}_k)_{k=1,\dots,C}$

gorithme, sur un jeu de données numériques en deux dimensions dans lequel apparaissent deux classes « naturelles » est donnée dans la figure 4.2. Dans cette figure, le paramètre C a été initialisé à 2 et les centres de classes sont visualisés par deux croix de deux couleurs différentes. La classification des différentes observations est matérialisée par une couleur différente (rouge et bleu) pour chacune des deux classes.

La convergence de l'algorithme des k-moyennes est assurée, mais pas nécessairement vers le minimum global du critère d'inertie intra-classes

$$\sum_{k=1,\dots,C} \sum_{x_i \in \mathcal{C}_k} \|x_i - \mu_k\|^2 \text{ où } \mu_k = \frac{1}{N_k} \sum_{x_i \in \mathcal{C}_k} x_i.$$

Aussi, les résultats finaux obtenus peuvent dépendre assez fortement de son initialisation, qui est aléatoire. Aussi, un des problèmes que l'on rencontre est que le critère de qualité étant monotone en le nombre de classes fixé, C, il n'y a aucun moyen simple pour déterminer un nombre de classes optimal. Enfin, l'algorithme fait intervenir la norme $\|x_i - p_j\|^2$, qui est naturellement définie pour des données numériques, mais n'est pas adaptée pour mesurer des proximités entre sommets d'un graphe ou bien entre chaînes de caractères. C'est pourquoi nous allons définir une nouvelle notion de distance par l'intermédiaire des noyaux.

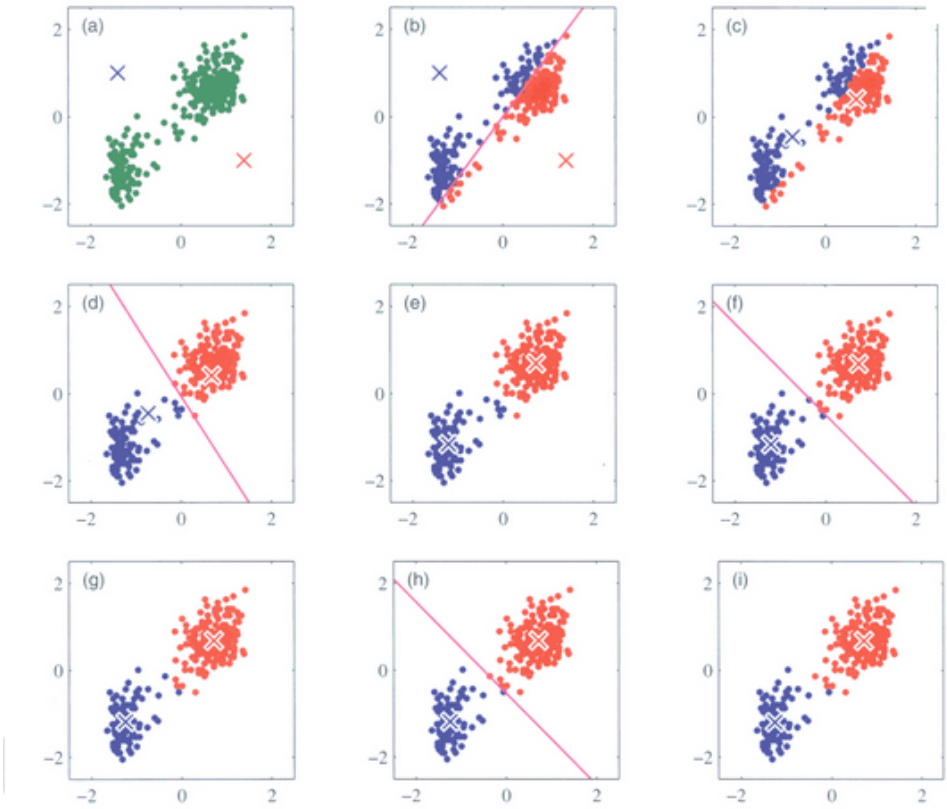


FIGURE 4.2.: Illustration de l'algorithme des k-moyennes sur un jeu de données "jouet"

Présentation des noyaux

Dans cette section, nous définirons tout d'abord la notion de noyau puis nous expliquerons comment intégrer ceux-ci dans une méthode de classification et les avantages que cette approche présente. Enfin, nous présenterons l'algorithme de classification par méthode à noyaux que nous avons utilisé.

Définition 3. *(voir [28]) Soit un espace abstrait $\mathcal{G}$. Un noyau est une fonction $K : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ telle que :*

1. $\forall\, x, x' \in \mathcal{G},\ K(x, x') = K(x', x)$ *(condition de symétrie)*
2. $\forall m \in \mathbb{N},\ \forall x_1, ..., x_m \in \mathcal{G},\ \forall \alpha_1, ..., \alpha_m \in \mathbb{R},\ \sum_{i,j=1}^{m} \alpha_i \alpha_j K(x_i, x_j) \geq 0$ *(condition de positivité)*

L'intérêt des noyaux réside dans la propriété suivante :

Théorème 1. *[1] Soit K un noyau sur l'espace abstrait $\mathcal{G}$, alors, il existe un espace de Hilbert $(\mathcal{H}, \langle ., . \rangle)$ et une application $\phi : \mathcal{G} \to \mathcal{H}$ tels que*

$$\forall\, x, x' \in \mathcal{G}, \qquad K(x, x') = \langle \phi(x), \phi(x') \rangle$$

Cette dernière équation indiquant que le noyau K joue le rôle de produit scalaire dans l'espace de Hilbert $\mathcal{H}$ qui est appelé espace de Hilbert à noyau reproduisant (RKHS,

Reproducing Kernel Hilbert Space [28]). Ainsi, sans connaître ni $\mathcal{H}$, ni ϕ, il est possible de déterminer une matrice de distances entre les données $x_1, \ldots, x_n$, mesurées dans $\mathcal{G}$ (non nécessairement euclidiennes), par le biais de K : par exemple, la distance induite par K entre les observations x_i et x_j est donnée par :

$$\sqrt{K(x_i, x_i) + K(x_j, x_j) - 2K(x_i, x_j)}$$

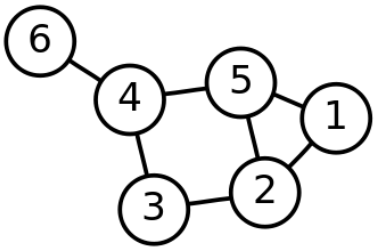
L'avantage de ces noyaux est double : ils permettent non seulement de changer d'espace de représentation et ainsi de travailler sur des problèmes non linéairement séparables, de travailler sur des données non numériques, mais aussi de combiner différents noyaux « simples » afin d'en obtenir un plus complexe et plus adapté au problème de classification. On pourra par exemple combiner un noyau sur données numériques à un noyau sur graphe (voir la section 4.3.1). Voici, à valeur d'illustration, quelques noyaux couramment utilisés :

- Données numériques (x, $y \in \mathbb{R}^d$) : Noyau gaussien $K(x, y) = e^{-\alpha||x-y||^2}$
- Sommets d'un graphe (x, y sont deux sommets d'un graphe) : Noyau de la chaleur $K(x, y) = e^{-\beta L(x,y)}$ [17], où L est le Laplacien du graphe définit comme suit :

$$L(x, y) := \begin{cases} \text{Deg}(x) & \text{si } x = y \\ -1 & \text{si } x \neq y \text{ et } x \text{ est adjacent à } y \\ 0 & \text{sinon} \end{cases}$$

  où $\text{Deg}(x)$ est le degré du sommet x, c'est-à-dire, le nombre de sommets adjacents à x. On peut interpréter ce noyau comme la modélisation d'un processus de conduction thermique, qui consiste en un transfert de chaleur le long des arêtes du graphe. Il existe également d'autres noyaux basés sur des régularisations du Laplacien [26], comme le noyau de temps moyen de parcours, « commute time »[11], qui est l'inverse généralisée du Laplacien et qui s'interprète comme le temps moyen pour passer du sommet x au sommet y en suivant une marche aléatoire (passant par les arêtes).

  La figure 4.3 illustre la valeur du Laplacien pour un graphe simple à 6 sommets.



Laplacian matrix

$$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$$

FIGURE 4.3.: Graphe simple et son Laplacien

L'algorithme de k-moyennes utilisant la métrique induite par le noyau est décrit dans la section suivante.

Description de l'algorithme de classification

Dans la version « à noyau » de l'algorithme des k-moyennes, les centres de classes prennent leurs valeurs dans l'espace image $\mathcal{H}$ et sont définis comme la moyenne des images par ϕ des observations de l'espace image classées dans la classe considérée. Ainsi, ils peuvent être écrits sous la forme :

$$\forall\, k = 1, \dots, C, \qquad \mu_k = \sum_{i=1}^{n} \gamma_{ki} \phi(x_i), \qquad \text{avec} \qquad \gamma_{ki} \geq 0 \text{ et } \sum_{i=1}^{n} \gamma_{ki} = 1,$$

c'est-à-dire, sous la forme d'une combinaison convexe des images par ϕ des observations initiales $\{x_1, \dots, x_n\}$. La procédure d'apprentissage de la classification est décrite dans l'algorithme 2 [29].

Algorithm 2 Algorithme k-moyennes à noyau (version batch)

1: **initialization** $\forall\, j = 1, \dots, C$, $\forall\, i = 1, \dots, n$, initialiser γ_{ji} aléatoirement dans $[0, 1]$ tq $\forall\, j = 1, \dots, C$, $\sum_{i=1}^{n} \gamma_{ji} = 1$

2: **repeat**

3: **Étape d'affectation** : $\forall\, i = 1, \dots, n$, x_i est affecté à la classe dont le centre est le plus proche :

$$\mathcal{C}(x_i) \leftarrow \mathcal{C}_l \qquad \text{tq } l = \arg\min_{k=1,\dots,C} \|\phi(x_i) - \mu_k\|_K^2 \text{ (distance induite par } K\text{, vue au 4.3.1)}$$

Étape de représentation : $\forall\, k = 1, \dots, C$, le centre de classe est recalculé :

$$\mu_k \leftarrow \arg\min_{\mu \in \mathcal{H}} \sum_{x_i \in \mathcal{C}_k} \|\phi(x_i) - \mu_k\|^2$$

4: **until** Convergence

Il est facile de montrer que l'étape de représentation, consistant à mettre à jour les coefficients de la combinaison convexe de $\{\phi(x_1), ..., \phi(x_n)\}$, se réduit simplement au calcul suivant :

$$\forall\, k = 1, \dots, C,\ \forall\, i = 1, \dots, n \qquad \gamma_{ki} \leftarrow \frac{1}{N_k} \mathbb{I}_{\{x_i \in \mathcal{C}_k\}}.$$

Maintenant que nous disposons d'un algorithme pour réaliser des tâches de classification sur différents types de données, nous allons voir comment combiner différents noyaux pour pouvoir effectuer la classification d'individus décrits par plusieurs jeux de données de natures diverses.

Combinaison convexe de noyaux

Supposons à présent que nous disposons de D jeux de données, $(x_i^d)_{i=1,\dots n, d=1,\dots,D}$, tous mesurés sur les mêmes n individus, chacun à valeurs dans un espace abstrait différent $\mathcal{G}_d$.

Supposons également que nous disposons d'un noyau K_d pour chacun de ces D jeux de données. On pourra par exemple avoir un noyau gaussien pour des données numériques, le noyau de la chaleur pour un graphe... On peut alors définir un nouveau noyau K : $\mathcal{G} \times \mathcal{G} \to \mathbb{R}$, où $\mathcal{G} = \mathcal{G}_1 \times ... \times \mathcal{G}_d$, comme la combinaison convexe de ces D noyaux :

$$\forall\, i, j = 1, \dots, n, \qquad K(x_i, x_j) = \sum_d \alpha_d K_d(x_i^d, x_j^d), \qquad \text{avec} \qquad \alpha_d \geq 0 \text{ et } \sum_d \alpha_d = 1.$$

Cette combinaison de noyaux aboutit ainsi au noyau K, vérifiant bien les propriétés de symétrie et de positivité définissant les noyaux [24, 22].

Cependant, les ordres de grandeur des noyaux peuvent différer d'un noyau à l'autre. Si ces biais peuvent être corrigés par l'application d'un coefficient approprié, le choix des coefficients peut s'avérer difficile. Pour répondre à ce problème, nous introduisons une normalisation, nommée « normalisation cosinus »[2], permettant de normaliser nos D noyaux initiaux et définie comme suit :

$$\tilde{K}_d(x_i, x_j) = \frac{K_d(x_i, x_j)}{\sqrt{K_d(x_i, x_i)K_d(x_j, x_j)}}$$

Le noyau issu de la combinaison convexe, sur laquelle on pourra par exemple choisir des pondérations $\alpha_1 = ... = \alpha_D = \frac{1}{D}$, devient alors :

$$\tilde{K}(x_i, x_j) = \sum_d \alpha_d \tilde{K}_d(x_i^d, x_j^d)$$

Une fois cette combinaison mise en place, nous pourrions démarrer nos classifications. Seulement, à l'instar de l'algorithme des k-moyennes, l'algorithme des k-moyennes à noyau fournit des résultats très instables, qui dépendent fortement de l'initialisation. Nous allons ainsi mettre en place une stratégie pour limiter la dépendance à l'initialisation.

4.3.2. Stabilisation des résultats

Introduction

L'algorithme des k-moyennes nous permet de prédire la classe à laquelle appartient chacun des individus. Cependant, les résultats de cette méthode dépendent fortement de l'initialisation des centres. De plus, il peut s'avérer difficile de choisir le « meilleur »résultat parmi un ensemble de classifications.

Le *Bagging* [3](pour *Bootstrap aggregating*) est un ensemble d'algorithmes d'apprentissage mis en place pour augmenter la stabilité et la précision d'algorithmes d'apprentissage utilisés en classification (ou en régression). Ces algorithmes, en combinant les résultats de classifications sur des ré-échantillonnages des données d'origine, permettent ainsi de réduire la variance mais aussi d'éviter le sur-apprentissage[3]. Ces multiples classifications, dont on agrège les prédictions, sont formées par bootstrap [10], c'est-à-dire en tirant des échantillons obtenus par tirage aléatoire avec remise de l'échantillon initial.

Dans un premier, nous allons définir un critère d'information mutuelle, que nous chercherons à maximiser, afin de juger de la qualité de nos classifications. Puis, nous implémenterons différents algorithmes de bagging pour stabiliser et améliorer les prédictions de notre algorithme. Plus tard, nous comparerons les résultats obtenus pour ces différentes méthodes de stabilisation des résultats afin de déterminer la plus adaptée à notre approche multi-noyaux.

Maximisation de l'information mutuelle

La première méthode que nous avons mis en place consiste à maximiser un critère, appelé *information mutuelle*. L'information mutuelle, qui est une mesure symétrique, permet de quantifier l'information partagée entre deux distributions [5].

[3]Le sur-apprentissage est un phénomène qui se produit lorsque l'on stocke trop d'informations que l'on ne peut alors généraliser

Soient $(\mathcal{C}_k^{(a)})_k$ et $(\mathcal{C}_k^{(b)})_k$, deux classifications en C classes. Alors on peut calculer l'information mutuelle comme suit :

$$I(\mathcal{C}^{(a)}, \mathcal{C}^{(b)}) = \sum_{i,j=1}^{C} \frac{n_{h,l}}{n} \times \log\left(\frac{n \times n_{ij}}{n_i^{(a)} \times n_j^{(b)}}\right)$$

avec $n_{ij} = |\mathcal{C}_i^a \cap \mathcal{C}_j^b|$, $n_i^{(a)} = |\mathcal{C}_i^a|$ et $n_j^{(b)} = |\mathcal{C}_j^b|$.

Nous chercherons, dans cette méthode, à trouver la classification $\mathcal{C}^*$ maximisant $\sum_{b=1}^{B} I(\mathcal{C}^*, \mathcal{C}^{(b)})$ où $\mathcal{C}^{(1)}, ..., \mathcal{C}^{(B)}$ représentent les résultats de B classifications des données par noyaux multiples dont les initialisations diffèrent (en ce sens, on ne fait pas réellement de bagging car on ne tire pas d'échantillons). Pour maximiser cette information mutuelle totale, nous utiliserons une approche par recuit simulé [16], qui est un algorithme d'optimisation classiquement utilisé dans la littérature et facile à implémenter, comme décrit dans l'algorithme 3.

Algorithm 3 Recuit simulé et information mutuelle

1: Construire B classification $\mathcal{P}^1, ..., \mathcal{P}^B$ par kernel k-means
2: Choisir le nombre d'itérations *iter*
3: Affecter chacun des individus de manière aléatoire à un cluster (parmi C), formant la partition $\mathcal{P}$
4: Calculer $Info = \sum_{i=1}^{B} \left(\frac{\sum_{h=1}^{C}\sum_{l=1}^{C} \frac{n_{h,l}^{(i)}}{n} log\left(\frac{n.n_{h,l}^{(i)}}{n_h.n_l^{(i)}}\right)}{\sqrt{\left(\sum_{h=1}^{C} n_h.log\frac{n_h}{n}\right)\left(\sum_{l=1}^{C} n_l^{(i)}.log\frac{n_l^{(i)}}{n}\right)}} \right)$
5: **for** i=1,...,*iter* **do**
6: Choisir aléatoirement deux clusters $\mathcal{P}_{(1)}$ et $\mathcal{P}_{(2)}$ de $\mathcal{P}$
7: Choisir aléatoirement i tq $i \in \mathcal{P}_{(1)}$ ou $i \in \mathcal{P}_{(2)}$
8: **if** $i \in \mathcal{P}_{(1)}$ **then** $\mathcal{P}^{(i)} \leftarrow \mathcal{P}_{(2)}$ **else** $\mathcal{P}^{(i)} \leftarrow \mathcal{P}_{(1)}$
9: Calculer $\Delta I = Info_{apres} - Info_{avant}$
10: **if** $\Delta I > 0$ **then**
11: On garde le changement
12: **end if**
13: **if** $\Delta I < 0$ **then**
14: On garde le changement avec une probabilité $p = exp^{(-\gamma \times log(iter) \times \Delta I)}$
15: **end if**
16: **end for**
17: **return** $\mathcal{P}$

Méthode de Leisch

Le premier algorithme de *bagging* que nous avons mis en place est quant à lui basé sur la méthode de Leisch [18], que nous adaptons pour de la classification à noyau. Cette méthode consiste à créer un nouveau jeu de données contenant les C centres de classes obtenus après application de la procédure de partitionnement (pour nous, les k-moyennes à noyaux multiples) et ce, sur B échantillons bootstrap issus du jeu de données initial. Une nouvelle procédure de classification est ensuite appliquée sur la matrice des $B \times C$ centres obtenus et formant le nouveau jeu de données. Une fois ce nouveau partitionnement

effectué, on peut alors récupérer les centres des classes finaux et affecter chacun des individus initiaux à la classe dont le centre est le plus proche. L'algorithme 4 présente la méthode de façon détaillée.

Algorithm 4 Bagging de Leisch

1: Choisir C et construire B échantillons d'apprentissage bootstrap $L^1, ..., L^B$
2: Appliquer kernel k-means à chacun des échantillons induisant $B \times C$ centres $\mu_{11}, \mu_{12}, ..., \mu_{1C}, \mu_{21}, ..., \mu_{BC}$
3: Construire $\mathcal{C}^B(C) = \{\mu_{11}, ..., \mu_{BC}\}$
4: Appliquer kernel k-means sur $\mathcal{C}^B$ et stocker les C centres finaux $\mu_{f1}, ..., \mu_{fC}$
5: $\forall i, \mathcal{C}(i) \leftarrow \arg\min_c \|\phi(x_i) - \mu_{fc}\|^2$

Méthode de Dudoit et Fridlyand

Le deuxième algorithme que nous avons mis en place est tiré de la procédure *BagClust2* [9]. Le principe de cet algorithme est de construire une matrice d'association entre les individus sur laquelle on appliquera une procédure de partitionnement.

Pour cela, nous analyserons les résultats de classification de B échantillons, en calculant pour chaque couple d'individus, la fréquence à laquelle ils apparaissent dans une même classe. Nous obtiendrons ainsi une nouvelle matrice de dissimilarité sur laquelle nous appliquerons une méthode de classification plus standard comme les k-means.

L'algorithme 5 décrit de manière détaillée cette méthode.

Algorithm 5 Bagging de Dudoit et Fridlyand

1: **Require** Nombre de clusters, C, et d'échantillons, B :
2: Initialiser deux matrices $\underset{(n\times n)}{\mathbf{A}} = (a_{ij}) = 0$ et $\underset{(n\times n)}{\mathbf{M}} = (m_{ij}) = 0$
3: **for** $b = 1...B$ **do**
4: Former l'échantillon bootstrap $\mathcal{L}^b = (x_1^b, ..., x_n^b)$
5: Appliquer les kernel k-means à $\mathcal{L}^b$
6: $\forall i \in \mathcal{L}^b_{test}, \mathcal{C}(x_i) \leftarrow \arg\min_c \|\phi(x_i) - \mu_{bc}\|^2_K$ (distance induite par le noyau)
7: $\forall i, \forall j$:

$$a_{ij} \leftarrow a_{ij} + \mathbb{1}\left(x_i \in \mathcal{L}^b_{test}, x_j \in \mathcal{L}^b_{test}, \mathcal{C}^b_{test}(x_i) = \mathcal{C}^b_{test}(x_j)\right)$$

où $\mathcal{C}^b_{test}(x_i)$ est la classe à laquelle l'individu i a été affecté

$$m_{ij} \leftarrow m_{ij} + \mathbb{1}\left(x_i \in \mathcal{L}^b_{test}, x_j \in \mathcal{L}^b_{test}\right)$$

8: **end for**
9: Définir la matrice de dissimilarité $D = (d_{ij})$ par $d_{ij} = 1 - \frac{a_{ij}}{m_{ij}}$
10: Lancer 100 fois l'algorithme des k-moyennes sur D
11: Retourner celle minimisant $\sum_{i=1}^n \sum_{k=1}^C \mathbb{1}_{\{i\in\mathcal{C}_k\}}(\phi(x_i) - \mu_k)^2$

5. Outils et méthodes

5.1. Méthodologie de travail

5.1.1. Git

Dans l'objectif de fournir un suivi de meilleure qualité des évolutions de mes travaux, j'ai été amené à utiliser un logiciel de contrôle de versions nommé Git[1]. Il a été développé par Linus Torvalds en 2005 pour permettre le développement collaboratif du noyau Linux. Utilisé dans un répertoire, il permet une gestion fine des versions de fichiers, en sauvegardant toutes leurs modifications et la possibilité de synchroniser le répertoire entre plusieurs ordinateurs via un serveur. Cela a eu pour effet de simplifier la correction et le suivi des scripts R produits lors de mon stage. Voici un processus de travail Git typique :

1. un utilisateur modifie un fichier existant, qui se situe dans son dépôt Git local ;
2. il écrit un court message décrivant ses modifications, en tapant **git commit** [**nom du fichier**] dans son invite de commande ;
3. il envoie ses modifications au serveur Git (dépôt distant), avec `git push` :
4. tous les autres utilisateurs (qui ont droit d'accès au serveur git) peuvent récupérer ces modifications à l'aide d'un **git pull**.

Le logiciel **Git** propose également une interface graphique, nommée gitk, pour visualiser simplement les versions successives des différents fichiers, présentée en figure 5.1.

Chaque ligne dans la sous-fenêtre en haut de la capture d'écran est une date de **commit**, c'est à dire le moment où l'auteur a voulu enregistrer ses modifications sur le Git (il est de bon usage d'ajouter une description succincte des ajouts et/ou modifications lors d'un commit, ici en haut à gauche). Lorsque l'on sélectionne un **commit**, le ou les fichiers modifiés dans celui-ci apparaissent dans la fenêtre du bas ; une fois qu'un fichier est sélectionné, on peut ainsi visualiser les ajouts en vert et les suppressions en rouge. De par ses origines collaboratives, **Git** gère nativement les conflits d'édition ; dans le cas où un même fichier est modifié en même temps par deux utilisateurs, s'il est modifié à des endroits différents, le fichier est automatiquement fusionné : les modifications des deux utilisateurs sont incorporées au fichier. Si le fichier est modifié au même endroit, le deuxième utilisateur qui envoie son commit au serveur avec un `push` voit son `push` refusé par le serveur. Il doit faire un `pull` pour récupérer les modifications du premier, choisir quelle version garder, puis faire un `commit`/`push` pour propager la modification retenue au serveur.

[1] Site web logiciel : `http://git-scm.com`

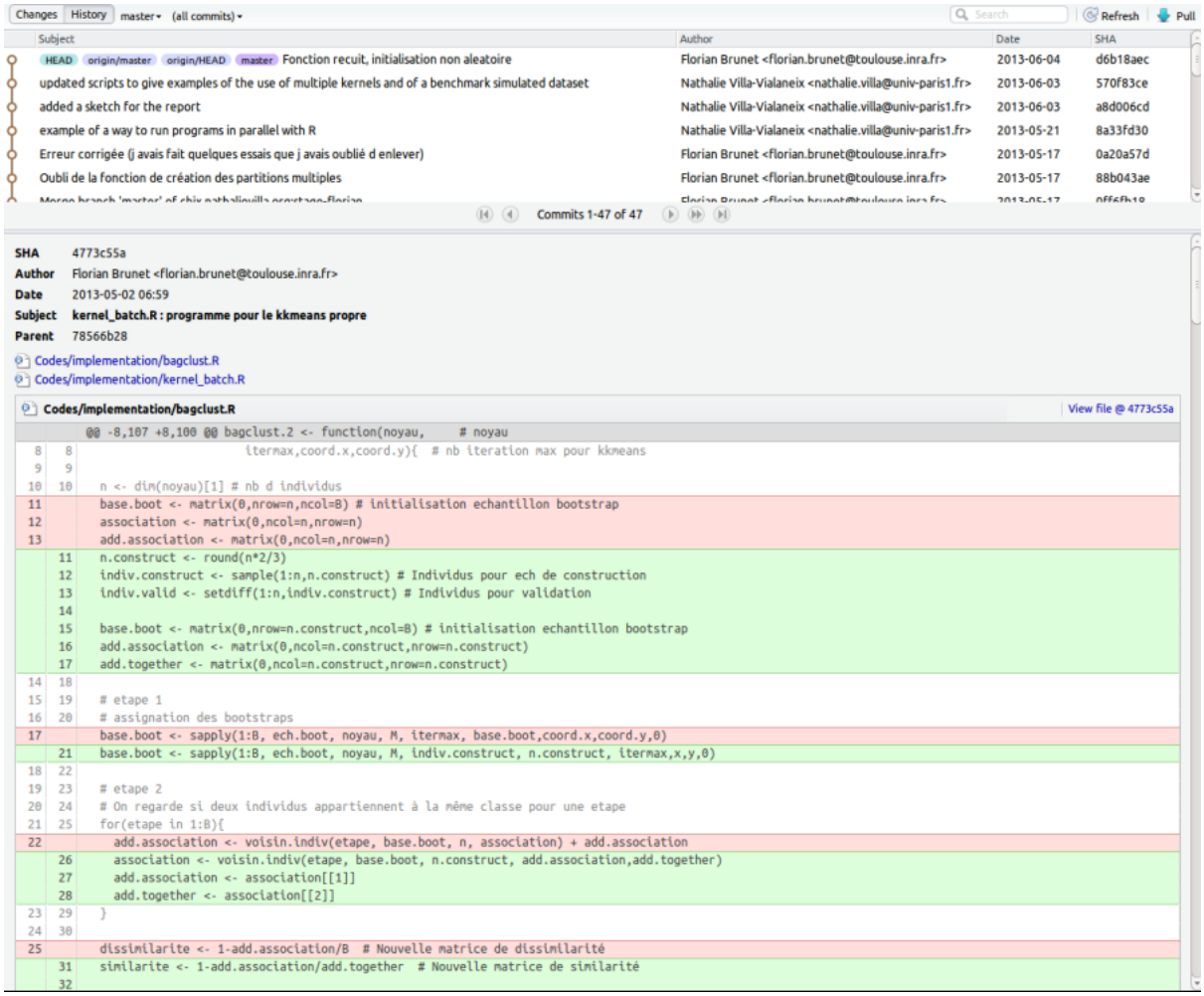


FIGURE 5.1.: Capture d'écran d'une fenêtre gitk

6. Travail effectué

6.1. Exemples jouets : classification de sommets de graphes étiquetés

Dans la section 4.3, nous avons étudié la méthodologie mise en place au cours de ce stage. Désormais, nous allons présenter les résultats, sur divers jeux de données, obtenus par les méthodes décrites. Pour cela, nous appliquerons tout d'abord nos algorithmes sur des exemples « jouet », simulés ou non, puis nous pourrons ensuite les déployer sur les données miRNA fournies par les biologistes de l'INRA de Toulouse.

6.1.1. Apport des méthodes à noyaux

Changement d'espace de représentation

Nous avons vu dans la section 4.3.1 que les méthodes à noyau permettent de changer l'espace de représentation de nos données d'origine. Afin de se représenter visuellement l'apport de ce changement d'espace, nous avons simulé un jeu de données numériques pour lequel l'algorithme des k-moyennes n'est pas adapté. En effet, ce jeu de données est constitué de 200 individus représentés dans $[0;1]^2$ pour lesquels :

1. les 100 premiers individus forment un « grand » cercle ;
2. les 100 individus suivants constituent un deuxième « petit » cercle enclavé dans le premier.

Ainsi, nous voulons séparer ces deux cercles en spécifiant à notre algorithme que nous attendons deux classes. La figure 6.1 nous montre le résultat obtenu avec l'algorithme des k-moyennes.

Comme attendu, l'identification des deux cercles est impossible (changer l'initialisation amène toujours à un résultat similaire) à cause de la non-séparabilité linéaire des deux cercles dans $\mathbb{R}^2$.

Nous allons maintenant utiliser l'algorithme des k-moyennes à noyau avec un noyau gaussien. La figure 6.2 présente les résultats obtenus avec neuf initialisations différentes.

On s'aperçoit que nous sommes parvenus à identifier les deux cercles pour six des neuf initialisations. On voit ainsi l'apport du changement d'espace des méthodes à noyau puisque l'on a pu résoudre un problème auparavant insolvable. Cependant, un tiers des résultats ne sont pas satisfaisants et mettent en lumière l'une des faiblesses de l'algorithme : la dépendance à l'initialisation. Avant de traiter ce problème de variabilité des résultats (section 6.1.2), nous allons auparavant étudier l'utilisation des k-moyennes à noyau sur un graphe.

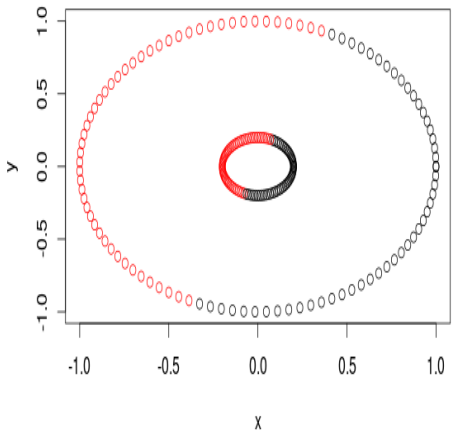


FIGURE 6.1.: Résultat de l'algorithme des k-moyennes ordinaire.

Classification de sommets d'un graphe

Pour tester l'utilisation des méthodes à noyau sur un graphe, nous avons utilisé le jeu de données « political books », constitué de 105 sommets et 441 arêtes. Ce jeu de données représente un réseau de livres (les sommets) traitant de la politique américaine durant l'élection présidentielle de 2004 et vendus par l'entreprise de commerce électronique *Amazon.com*. Les arêtes entre les livres modélisent, quant à elles, la fréquence de co-achat de livres par les mêmes acheteurs.

Nous allons utiliser le noyau de la chaleur, qui permet de regrouper des individus fortement connectés, pour réaliser la classification, espérant ainsi obtenir des classes aux caractéristiques bien marquées et susceptibles d'être plus ou moins fortement connectées. La figure 6.3 représente le résultat obtenu pour 7 classes : les cercles représentent les classes identifiées, dont l'aire est proportionnelle au nombre d'individus qui les composent. Les cercles sont positionnés par un algorithme de force [13] et la position n'a donc pas de signification particulière en elle-même. Chacun des cercles peut être composé de trois couleurs, dont la surface représente la proportion de livres de la classe étant pro-républicains (en rouge), pro-démocrates (en bleu) ou neutres (en rose). Notons que cette information n'a pas été utilisée dans la classification. Enfin l'épaisseur des arêtes entre deux cercles est proportionnelle au nombre d'arêtes entre les individus des deux classes.

Ainsi, on voit que cette classification semble pertinente, avec notamment la présence de deux classes principales contenant des livres d'une seule appartenance politique (ou presque), républicain pour le rouge, démocrate pour le bleu. Ces classes sont d'autant plus pertinentes que la classe des livres pro-démocrates est surtout reliée aux classes contenant des livres également démocrates ou neutres. On peut également noter la présence d'une classe « intermédiaire »contenant des livres des 3 familles politiques.

6.1.2. Apports du bagging

Nous allons maintenant chercher à vérifier la pertinence de nos algorithmes de bagging et, si possible, détecter celui susceptible d'être « le meilleur ». Pour cela, nous allons à nouveau commencer par tester nos trois algorithmes sur un jeu de données simulées contenant deux cercles dont l'un est enclavé dans l'autre, auxquels on ajoute une perturbation. Rappelons que l'objectif est d'obtenir des résultats plus précis et surtout plus

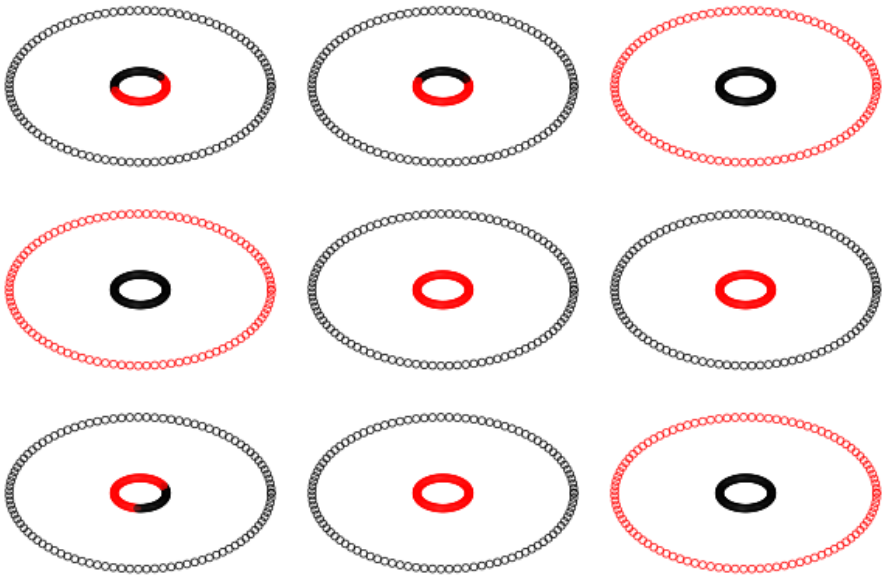


FIGURE 6.2.: Résultats de l'algorithme des k-moyennes à noyau

stables qu'avec l'algorithme de k-moyennes à noyau « simple ». La figure 6.4 présente les résultats obtenus pour les trois méthodes de bagging, avec trois initialisations différentes pour chacune d'elles et 200 classifications initiales.

On remarque tout de suite que l'algorithme basé sur la méthode de Leisch semble moins performant. En effet, les classifications ne sont pas stables puisque les trois observées avec cette méthode donnent des résultats différents. À l'inverse, on constate que l'algorithme basé sur la méthode de Dudoit et celui par maximisation de l'information mutuelle mènent à des résultats stables[1] et satisfaisants. Par ailleurs, on peut remarquer que la méthode basée sur celle de Dudoit est ici la plus rapide (voir 6.5).

Les résultats de deux des trois algorithmes se sont ici révélés satisfaisants. Cependant, on avait pu remarquer en 6.1.1 que les classifications par multi-noyaux sur un jeu de données semblable mais sans utiliser de méthode de bagging menait le plus souvent au résultat attendu, bien que ceux-ci soient légèrement instables.

Nous allons maintenant étudier le comportement de nos algorithmes, lorsque les classifications issues de k-moyennes à noyau sont à la fois très instables et clairement insatisfaisantes[2]. En effet, nous venons de voir que nos algorithmes étaient capables d'améliorer la stabilité, mais nous ne savons pas encore s'ils peuvent extraire un résultat plus pertinent, plus précis, que ceux obtenus sans bagging.

Pour tester cela, nous avons simulé un jeu de données contenant non plus deux, mais trois cercles (composés chacun de 100 individus) légèrement perturbés et enclavés les uns dans les autres. On peut voir sur la figure 6.6 que les résultats obtenus en spécifiant trois classes et avec quatre initialisations différentes fournissent des partitions à la fois très instables et ne distinguant absolument pas les cercles générés.

Voici les résultats obtenus pour les trois méthodes de bagging, avec trois initialisations différentes pour chacune d'elles : figure 6.7 sachant que l'on a choisi en entrée :

[1] Nous ne présentons les résultats que sur trois initialisations différentes, mais la stabilité a été vérifiée sur un plus grand nombre d'initialisations

[2] Nous ne nous préoccupons pas ici du fait que le noyau utilisé est le plus adapté ou non.

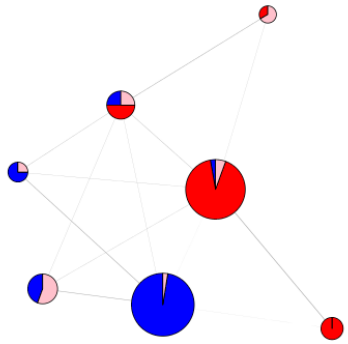


FIGURE 6.3.: Résultat de l'algorithme de k-moyennes à noyau sur les données « political books »

- 500 échantillons bootstrap pour les méthodes de Dudoit et Leisch ;
- 500 classifications de référence et 50000 itérations pour la maximisation de l'information mutuelle.

Pour commencer, on constate que les résultats obtenus avec la méthode de Leisch (qui n'était pas satisfaisante précédemment) ne sont, une fois de plus, pas bons. En effet, l'algorithme ne détecte presque qu'une seule classe et est incapable de distinguer nos cercles. Cet algorithme n'étant pas performant pour nos classifications à noyaux, nous ne l'utiliserons pas par la suite sur nos données réelles.

L'algorithme par maximisation de l'information mutuelle donnait précédemment de bons résultats. On voit ici qu'il n'est absolument pas adapté. Non seulement, les résultats n'ont pas du tout été stabilisés, mais les classifications sont même plus mauvaises que celles obtenues sans bagging, se rapprochant de résultats que l'on pourrait recueillir après un partitionnement aléatoire des individus. Ces résultats proviennent probablement du critère choisi. En effet, on cherche à maximiser l'information **mutuelle** alors que nos classifications initiales sont très différentes. On peut donc penser que cet algorithme est adapté dans le cas où les classifications de référence sont relativement proches, mais inadapté quand elles fluctuent trop.

Enfin, l'algorithme basé sur la méthode Dudoit donne quant à lui des résultats beaucoup plus satisfaisants. Alors que la classification à noyaux « simple » donnait des résultats très instables et était incapable de distinguer les cercles, on voit ici une importante amélioration puisque le cercle intérieur est presque parfaitement identifié, le cercle intermédiaire plutôt bien identifié et une identification certes un peu moins bonne pour le cercle extérieur, mais bien meilleure que sans le bagging. Par ailleurs, les résultats ne sont pas parfaitement stables avec 500 échantillons bootstrap, mais restent tout de même assez semblables. De plus, on peut augmenter le nombre d'échantillons bootstrap en entrée de notre algorithme pour améliorer sa stabilité (qui finit cependant par atteindre un plateau). Concernant la vitesse d'exécution, le tableau de la figure 6.8 nous indique que cet algorithme est une fois encore le plus rapide.

Ainsi, nous privilégierons par la suite l'utilisation de l'algorithme basé sur la méthode de Dudoit qui est le plus performant et qui a également le temps d'exécution le plus court. Avant de mettre en œuvre cette méthodologie sur les données réelles, il nous reste

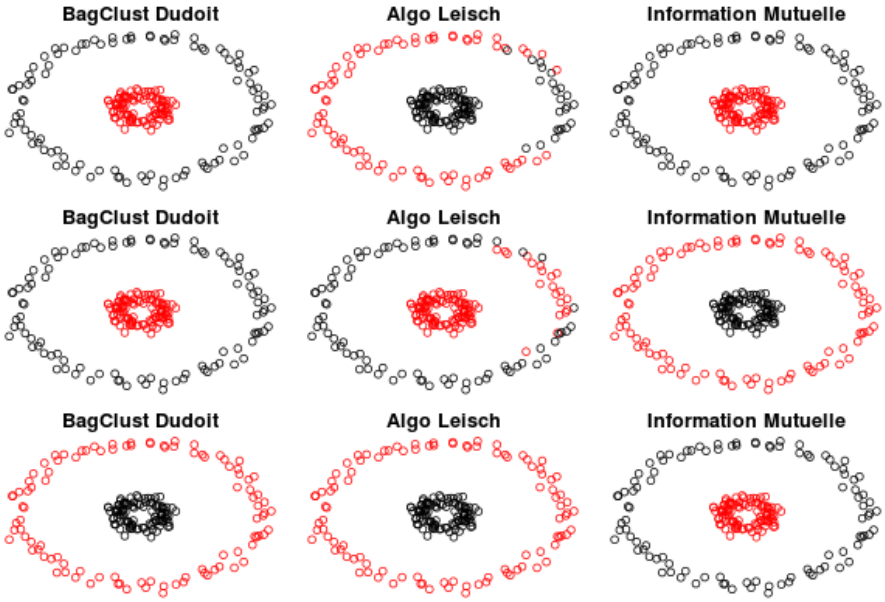


FIGURE 6.4.: Bagging sur deux cercles

Vitesse d'exécution des algorithmes (en secondes)

Dudoit	Leisch	Information mutuelle
3.2	4.7	10.5

FIGURE 6.5.: Comparaison de la vitesse des algorithmes (1)

cependant à illustrer l'utilisation de la classification **multi**-noyaux.

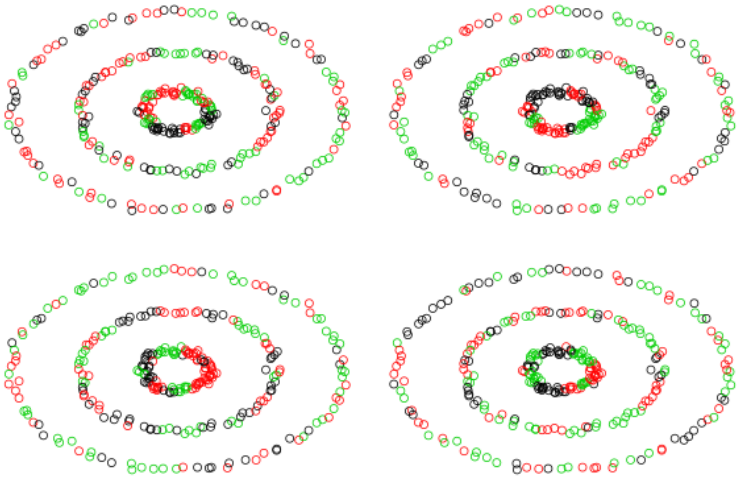


FIGURE 6.6.: Kernel k-moyennes sur trois cercles

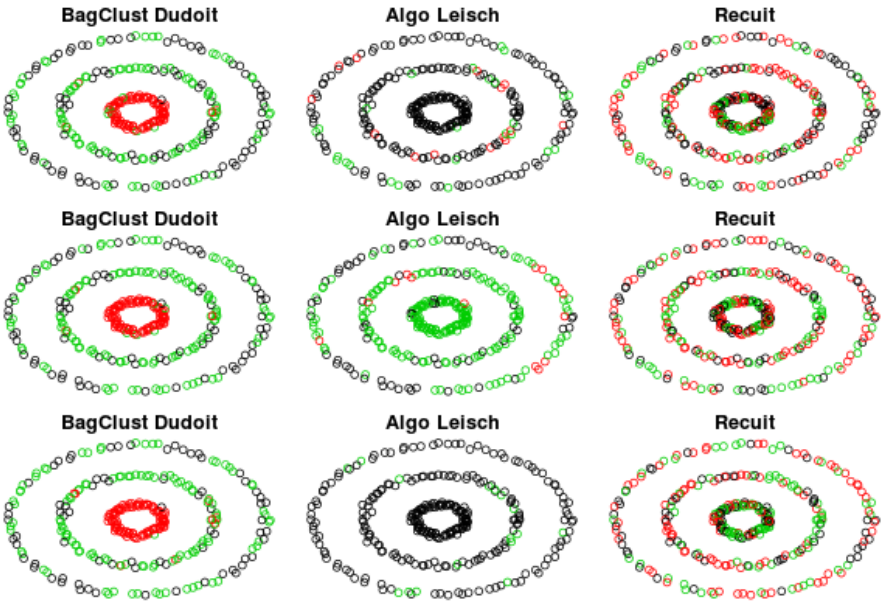


FIGURE 6.7.: Bagging sur trois cercles

Vitesse d'exécution des algorithmes (en secondes)

Dudoit	Leisch	Recuit
14	35	900

FIGURE 6.8.: Comparaison de la vitesse des algorithmes (2)

6.1.3. Illustration de l'approche multi-noyau

Ce jeu de données simulé utilisé pour illustrer l'approche multi-noyau, provient de l'article [30]. Il contient 200 individus appartenant à 8 classes, numérotées dans la suite de 1 à 8. Pour les identifier, nous disposons de 3 types de données (numérique, qualitative, graphe). Chaque jeu de données nous permet de séparer les individus en deux groupes (2 groupes pour chacun des 3 jeux de données impliquant $2^3 = 8$ classes). Nous allons ainsi pouvoir mettre en place notre stratégie multi-noyaux et observer de premiers résultats.

Description des données Les données numériques décrivant nos individus sont générées par des lois normales bidimensionnelles. La moitié des individus (ceux des groupes 1 à 4) sont caractérisés par une abscisse et une ordonnée générées selon une loi normale bidimensionnelle $\mathcal{N}_2(\mu_1, \Sigma_1)$ avec $\mu_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ et $\Sigma_1 = \begin{pmatrix} 0,3 & 0 \\ 0 & 0,3 \end{pmatrix}$ tandis que les coordonnées de l'autre moitié des individus sont générées selon une $\mathcal{N}_2(\mu_2, \Sigma_2)$ avec $\mu_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ et $\Sigma_2 = \begin{pmatrix} 0,3 & 0 \\ 0 & 0,3 \end{pmatrix}$. La valeur des variables numériques pour les 200 individus générés aléatoirement est illustrée dans la figure 6.9. Notons que dans cette figure, les points sont représentés de huit couleurs différentes représentant la vraie classe de l'individu.

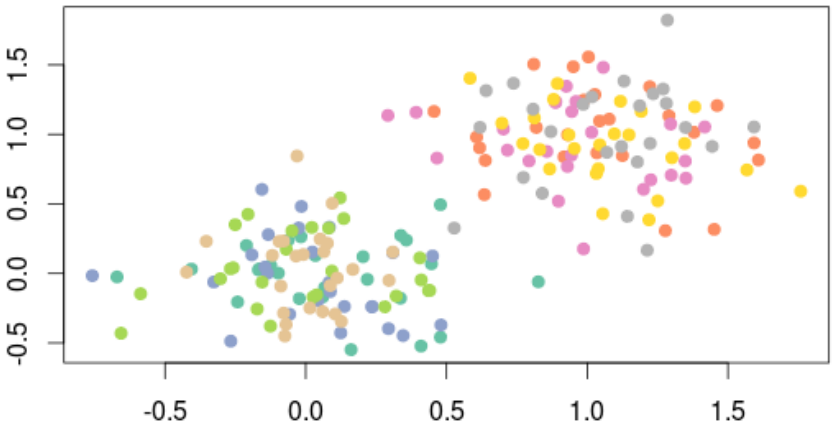


FIGURE 6.9.: Les données numériques

Le graphe a été généré selon un modèle aléatoire « planted partition tree » [30] dans lequel les sommets correspondant aux individus des classes paires ont une probabilité d'être reliée égale à $0,3$ (et de même pour les sommets des individus des classes impaires) tandis qu'un sommet d'une classe paire n'a qu'une probabilité égale à $0,01$ d'être relié à un sommet d'une classe impaire. La figure 6.10 représente le graphe produit de cette façon, la couleur des sommets représentant toujours la classe d'appartenance.

Enfin, les données catégorielles sont un facteur à deux niveaux : les individus des groupes 1, 2, 5 et 6 ont pour valeur le premier niveau du facteur et les autres, le deuxième niveau.

Classification multi-noyaux Nous avons choisi d'appliquer un noyau gaussien sur les données numériques et sur les données catégorielles (codées en 0/1) ainsi que le noyau

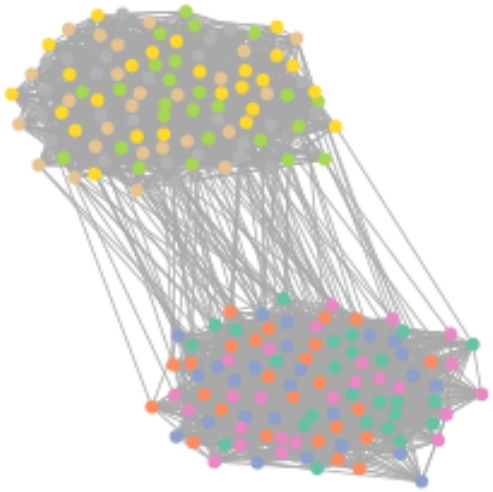


FIGURE 6.10.: Le graphe

du temps de parcours pour le graphe. Nous avons ensuite normalisé ces trois noyaux par l'intermédiaire de la « normalisation cosinus ». Ensuite, ces trois noyaux normalisés ont été combinés avec des poids équivalents $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. L'algorithme de bagging a ensuite été paramétré avec 1000 échantillons bootstrap. Le tableau 6.1 permet de comparer la classification ainsi obtenue aux vraies classes des individus.

		Classe prédite							
		1	2	3	4	5	6	7	8
Vraie classe	1	0	0	0	0	0	24	0	**1**
	2	0	0	0	0	0	0	0	25
	3	0	0	25	0	0	0	0	0
	4	25	0	0	0	0	0	0	0
	5	0	24	0	0	0	**1**	0	0
	6	0	0	0	25	0	0	0	0
	7	0	0	0	0	0	0	25	0
	8	0	0	0	0	24	0	**1**	0

TABLE 6.1.: Résultats de la classification

Idéalement, nous attendions que pour chaque vraie classe, les 25 individus soient tous affectés à une même classe « prédite » et que chacune des classes prédites contienne également 25 individus. Nous nous apercevons ainsi que la classification obtenue est tout à fait satisfaisante puisque seulement trois individus ne répondent pas à cette règle (en rouge et en gras dans le tableau 6.1).

Cet exemple nous ayant permis de valider la méthodologie de classification mise en œuvre, nous allons pouvoir l'appliquer aux données de micro-ARNs fournies par les biologistes de l'INRA.

6.2. Mise en œuvre sur les données miRNA

Rappelons que la problématique biologique du stage est de détecter de manière non supervisée des micro-ARNs probables. Pour ce faire, nous utilisons l'approche multi-noyaux précédemment décrite pour intégrer des informations sur le graphe de co-expression et des informations supplémentaires sur les sommets de ce graphe. Pour rendre cette approche plus robuste, nous la combinons à une approche par bagging. Nous attendons que les résultats obtenus donnent des groupes d'ARNs présentant des caractéristiques de micro-ARNs et dont certains n'ont pas encore été validés biologiquement.

6.2.1. Contexte

L'acide ribonucléique (ARN ou RNA en anglais) est une molécule biologique que l'on trouve dans les cellules des organismes vivants et qui est synthétisée à partir d'une matrice d'ADN dont il est une copie. En particulier, certains ARNs dits messagers sont utilisés par les cellules pour fabriquer des protéines nécessaires au fonctionnement du vivant, à partir de l'ADN. Les micro-ARNs (miRNAs) sont un ensemble de petits ARNs dont le rôle n'est pas la fabrication de protéines mais qui sont impliqués dans la régulation des gènes et la répression de la transcription. Chez les animaux, les miRNAs ont un rôle de régulation majeur durant le développement embryonnaire et sont aussi impliqués dans diverses maladies comme le cancer, les maladies cardio-vasculaires et certaines dégénérescences neurologiques. Il est donc important de pouvoir identifier et caractériser de nouveaux miRNAs afin de pouvoir étudier de manière plus précise leur implication dans le fonctionnement du vivant.

Les méthodes habituelles de détectionde micro-ARNs conduisent très souvent à de nombreux faux positifs, qu'il est possible de détécter à l'aide de méthodes d'apprentissage supervisé [7, 8, 20] (SVM) qui nécessitent l'utilisation d'un ensemble d'apprentissage, contenant exemples positifs et négatifs. Nous nous focalisons ici sur un problème intermédiaire dans lequel nous utilisons une approche que nous pourrons qualifier de semi-supervisée : classiquement les chaînes de traitements des données issues des séquenceurs fournissent une liste de miRNAs potentiels et un certain nombre de tests permettent d'éliminer les faux positifs les plus évidents. Par ailleurs, sur les miRNAs potentiels restants, un certain nombre sont identifiés comme étant des micro-ARNs par l'utilisation de banques de données publiques référençant les miRNAs tel que miRBase [14] ou des banques plus généralistes sur les ARNs non codants tel que RFAM [15], les autres étant de nature inconnue. Le but du travail est de s'appuyer sur la connaissance préalable de la nature de certains des ARNs pour rechercher, parmi l'ensemble des ARNs dont nous disposons, ceux qui sont les plus probablement des miRNAs. Nous utiliserons pour cela la méthodologie construite au cours du stage qui se démarque par le fait que, contrairement aux approches supervisées décrites plus haut, on ne dispose pas d'un ensemble d'apprentissage préalable puisque l'on ne dispose que de vrais positifs mais pas d'exemples négatifs.

Nous utilisons ainsi une approche par graphe : les données dont nous disposons sur les ARNs, candidats potentiels à être des miRNAs, sont un graphe de co-expression dans lequel les sommets sont

- des ARNs et les arêtes indiquent une forte co-expression entre deux ARNs (voir la section 6.2.2 pour plus de détails sur l'inférence de ce graphe) ;

- décrits par des variables numériques et catégorielles, appelées « étiquettes », pertinentes pour leur identification en tant que micro-ARNs.

Ainsi, les données peuvent être décrites sous la forme d'un graphe dont les sommets sont étiquetés par un ensemble de descripteurs, numériques et non numériques. Nous proposons le développement de notre méthodologie de classification non supervisée des sommets d'un graphe pour grouper les ARNs en groupes homogènes selon tous leurs descripteurs. Les clusters ainsi obtenus seront ensuite analysés selon qu'ils contiennent ou non un grand nombre de micro-ARNs déjà connus.

6.2.2. Description des données

Le graphe considéré dans cette section est issu de données d'expression, c'est-à-dire, de données mesurant l'activité d'un miRNA potentiel dans la cellule au moment de l'expérience. Cette expression correspond au comptage du nombre de lectures qui ont été séquencées et qui se sont alignées avec une forte similarité sur le génome de référence. L'expression de 500 miRNAs potentiels (250 annotés comme tels et 250 inconnus) a été obtenue sur 38 échantillons (données non encore publiées). Les données ont été préalablement normalisées par la méthode implémentée dans le package R **DESeq** puis le graphe a été inféré en utilisant une méthode graphique gaussienne (comme implémentée dans le package R **glasso**, voir [12]) bootstrappée.

Par ailleurs, chaque miRNA a été caractérisé par un certain nombre de méta-données ; pour notre problématique de classification, nous avons retenu comme étiquettes catégorielles :

- la présence ou non d'une structure particulière qui est une structure secondaire en tige-boucle, évaluée suite au repliement des régions flanquantes en structure secondaire à l'aide de RNAfold [19] ;
- la présence ou non du miRNA star qui est le complèmentaire du miRNA lorsque celui-ci est sous sa forme non mature ;

Ces deux informations sont des signaux biologiques qui penchent en faveur de la présence d'un vrai micro-ARN. Une seule étiquette numérique a été utilisée, la taille du pré-miRNA (précurseur du micro-ARN).

6.2.3. Méthodologie

La méthodologie mise en place a été la suivante :

- les noyaux choisis pour chacun des types de données ont été, le noyau de temps moyen de parcours (« commute time kernel »comme dans [11]) pour le réseau de co-expression, le noyau gaussien pour la variable numérique et le noyau gaussien sur le codage disjonctif des variables qualitatives. La normalisation présentée en section 4.3.1 a été appliquée aux trois noyaux ;
- on a fait varier le nombre de classes de la classification de 2 à 10 pour étudier la qualité du clustering en fonction du nombre de classes : la modularité [21], qui est

un critère de la qualité d'une classification de sommets dans un graphe, et l'inertie intra-classes, calculée dans l'espace image induit par le noyau :

$$\sum_{j=1,\dots,k} \sum_{i \in \mathcal{C}_j} \widetilde{K}(x_i, x_i) - 2 \sum_{l} \gamma_{jl} \widetilde{K}(x_i, x_l) + \sum_{ll'} \gamma_{jl} \gamma_{jl'} \widetilde{K}(x_l, x_l)$$

qui est le critère cible minimisé dans la classification par k-moyennes à noyau. La modularité optimale n'est pas un critère monotone en le nombre de classes et donne donc une indication du nombre de classes optimal, du moins pour la classification des sommets basée sur la structure seule du réseau, tandis que l'inertie intra-classes optimale est un critère qui décroît avec le nombre de classes ;

- 1 000 échantillons bootstrap ont été générés pour produire une classification sur la base de la méthode de bagging de Dudoit décrite dans la section 4.3.2. La classification finale, basée sur la similarité issue du bagging, était la meilleure de 100 itérations sur la base de l'inertie intra-groupes.

L'évolution de la modularité et de l'inertie intra-classes induite par le noyau est donnée dans la figure 6.11. À la vue de ces graphiques, deux classifications ont été retenues : celle

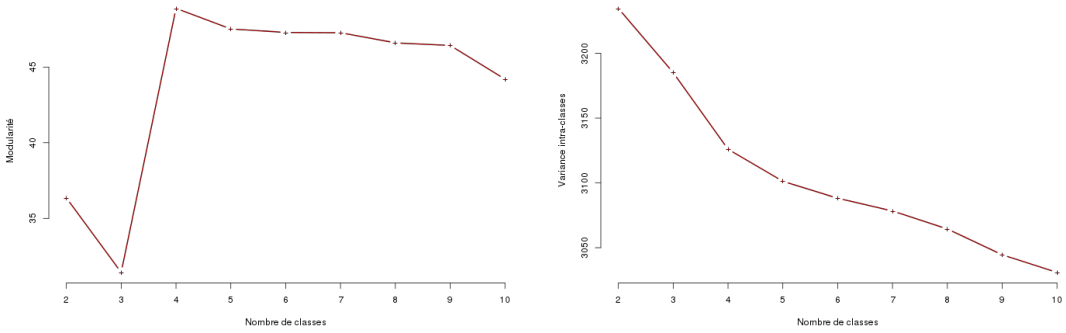


FIGURE 6.11.: Evolution de la modularité (à gauche) et de l'inertie intra-groupes (à droite)

à 4 classes (qui correspond au maximum de la modularité et au premier décrochage de l'inertie intra-classes) et celle à 9 classes (qui se trouve à la fin d'un « plateau »pour la modularité et avant une diminution de la pente dans la décroissance de l'inertie intra-classes). Dans la section suivante, nous analysons donc les résultats de ces deux classifications.

6.2.4. Résultats et discussion

Les caractéristiques des classes de la classification à 4 classes sont données dans le tableau 6.12. En particulier, on s'intéresse à la troisième colonne qui correspond au pourcentage d'ARNs annotés, c'est-à-dire des ARNs qui ont été validés comme étant des micro-ARNs, qui n'a pas été utilisée comme entrée de l'algorithme de classification. On constate que pour 3 classes sur 4 (hormis la classe 1), les résultats pour cette variable sont très tranchés ce qui plaide en faveur de classes relativement homogènes du point de vue de leur nature (vrais micro-ARNs ou possibles faux positifs). Également, la troisième

classe est très bien identifiée avec des micro-ARNs tous annotés[3], possédant une structure en tige-boucle (hairpin en anglais), la présence du star et d'une très petite taille du précurseur. Cependant, avec seulement 4 classes, la classification est encore un peu grossière pour identifier des candidats micro-ARNs : seule la classe 3 contient un fort pourcentage de micro-ARNs annotés mais elle ne contient aucun micro-ARN non annoté qui serait un bon candidat à être un vrai micro-ARN.

Classe	Nb de sommets	% annotés	densité (%)	% hairpin	% star	taille pre-miRNA
1	220	67,27	0,05	50,00	4,09	73,35
2	95	2,11	0,62	29,47	2,11	74,17
3	95	100,00	0,07	95,79	98,95	62,08
4	89	5,62	0,65	32,58	6,74	73,26

FIGURE 6.12.: Analyse des résultats : classification à 4 classes

De même, les résultats de la classification à 9 classes sont analysés dans la table 6.13.

Classe	Nb de sommets	% annotés	densité (%)	% hairpin	% star	taille pre-miRNA
1	140	68,57	0,03	52,86	5,71	73,04
2	95	2,11	0,62	28,42	2,11	74,17
3	88	4,55	0,65	32,95	5,68	73,42
4	75	100,00	0,06	94,67	100,00	61,56
5	37	78,38	0,44	40,54	2,70	73,65
6	30	36,67	0,26	50,00	0,00	74,27
7	18	100,00	0,58	100,00	100,00	63,17
8	14	100,00	0,48	57,14	7,14	73,43
9	2	50,00	1,00	50,00	50,00	66,50

FIGURE 6.13.: Analyse des résultats : classification à 9 classes

La classification donne des résultats intéressants dans le sens où elle permet bien de regrouper des miRNAs de caractéristiques similaires : fortement co-exprimés (avec des sous-graphes de forte densité, sauf pour les classes 1 et 4), avec des pourcentages de hairpin ou bien de star très forts ou très faibles (en particulier pour les classes 4 et 7) et des tailles de pré-miRNAs qui peuvent être assez différentes (en particulier ici aussi pour les classes 4 et 7). À l'exception de la classe 6, le pourcentage d'annotation parmi les miRNAs identifiés est soit fort, soit faible. Un test de Fisher indique que la classe 1 et la classe 5 sont significativement enrichies en miRNAs annotés par rapport à l'ensemble des ARNs disponibles au départ (au risque de 1%). Cela indique que les ARNs non annotés de ces classes ont un bon potentiel d'être de vrais miRNAs et sont donc des candidats à privilégier pour être analysés plus précisément par les biologistes afin d'identifier d'éventuels nouveaux miRNAs.

[3]Rappel : Un ARN annoté est un ARN qui a été validé comme étant un micro-ARN

Quatrième partie .

Conclusion

Durant ce stage, j'ai acquis des connaissances sur la théorie des graphes et sur l'utilisation des méthodes à noyaux dans un cadre de classification non supervisée. Ce travail de recherche m'a permis de développer des scripts R afin d'apprécier l'apport de ces méthodes, à partir de différents jeux de données. Ensuite, l'étude d'articles de recherche m'a permis de développer et de comparer différents algorithmes permettant de rendre des résultats de classifications plus précis et plus stables. Enfin, la méthodologie statistique mise en place a été confrontée à une problématique biologique de détection de micro-ARNs, à laquelle des éléments de réponse ont pu être apportés. Une partie de ces résultats sera par ailleurs présentée lors de la conférence MARAMI[4], se déroulant du 16 au 18 octobre 2013 à Saint-Étienne.

Sur un plan professionnel, j'ai eu l'occasion de me confronter à un sujet de recherche appliquée, ce que je n'avais pu faire lors de mes précédents stages, ajoutant ainsi une expérience très valorisante. Par ailleurs, ce stage m'a permis d'apprendre à développer et à organiser des programmes avec davantage de rigueur et d'optimalité.

D'un point de vue personnel, ce stage m'a enrichi sur différents aspects. Tout d'abord, en m'apportant une plus grande confiance en moi, car je doutais de mes capacités à mener des travaux de recherche appliquée. Un autre apport, plus attendu mais tout aussi enrichissant, est évidemment la découverte et l'appropriation de nouvelles connaissances dans le champ de la statistique. Enfin, cette expérience m'a apporté une nouvelle façon de travailler. En effet, le développement de cette méthodologie était destiné à des biologistes, dont il fallait comprendre les besoins et les problématiques.

En marge de ce stage, je tiens à remercier mes deux encadrantes pour m'avoir permis de participer activement à l'organisation des Journées de Statistique de la Société Française De Statistique et aux Journées Ouvertes Biologie, Informatique et Mathématiques se déroulant cette année à Toulouse. Cette immersion et la possibilité d'assister aux conférences furent très enrichissantes.

[4]Modèles et Analyse des Réseaux : Approches Mathématiques et Informatique, http://lipn.univ-paris13.fr/marami2013/MARAMI13/Accueil.html

Bibliographie

[1] Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society 68*, 3 :337–404, 1950.

[2] A. Ben-Hur and J. Weston. A user's guide to support vector machines. *Transactions of the American Mathematical Society 68*, 2010.

[3] Leo Breiman. Bagging predictors. In *Machine Learning*, pages 123–140, 1996.

[4] Florian Brunet, Nathalie Villa-Vialaneix, Christine Cierco-Ayrolles, and Jérôme Mariette. Classification non supervisée d'un graphe de co-expression avec des métadonnées pour la détection de micro-arns. 2013.

[5] Thomas M. Cover and Joy Thomas. *Elements of Information Theory*. Wiley, 1991.

[6] J.D. Cruz, C. Bothorel, and F. Poulet. Entropy based community detection in augmented social networks. pages 163–168, 2011.

[7] J. Ding, S. Zhou, and J. Guan. MiRenSVM : towards better prediction of microRNA precursors using an ensemble SVM classifier with multi-loop features. *BM*, 11 :S11, 2010.

[8] J. Ding, S. Zhou, and J. Guan. miRFams : an effective automatic miRNA classification method based on n-grams and a multiclass SVM. *BMC Bioinformatics*, 12 :216, 2011.

[9] Sandrine Dudoit and Jane Fridlyand. Bagging to improve the accuracy of a clustering procedure. *Bioinformatics*, 19 :1090–1099, 2003.

[10] B. Efron and R. Tibshirani. An introduction to the bootstrap. 1993.

[11] F. Fouss, A. Pirotte, J.M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph, with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3) :355–369, 2007.

[12] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3) :432–441, 2008.

[13] T. Fruchterman and B. Reingold. Graph drawing by force-directed placement. *Software, Practice and Experience*, 21 :1129–1164, 1991.

[14] S. Griffiths-Jones, R. Grocock, S. van Dongen, A. Bateman, and A. Enright. mirbase : microrna sequences, targets and gene nomenclature. *Nucleic Acids Research*, 34(suppl 1) :D140–D144, 2006.

[15] S. Griffiths-Jones, S. Moxon, M. Marshall, A. Khanna, S. Eddy, and A. Bateman. Rfam : annotating non-coding RNAs in complete genomes. *Nucleic Acids Research*, 33(suppl 1) :D121–D124, 2005.

[16] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220 :671–680, 1983.

[17] John Lafferty and Risi Imre Kondor. Diffusion kernels on graphs and other discrete structures. 2002.

[18] Friedrich Leisch. Bagged clustering. 1999.

[19] R. Lorenz, S.H. Bernhart, C. Höner zu Siederdissen, H. Tafer, C. Flamm, P.F. Stadler, and I.L. Hofacker. ViennaRNA package 2.0. *Algorithms for Molecular Biology*, 6 :26, 2011.

[20] D. Mapleson, S. Moxon, T. Dalmay, and V. Moulton. MirPlex : a tool for identifying miRNAs in high-throughput sRNA datasets without a genome. *Journal of Experimental Zoology, Part B : Molecular and Developmental Evolution*, 320B :47–56, 2013.

[21] M.E.J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review, E*, 69 :026113, 2004.

[22] Madalina Olteanu, Nathalie Villa-Vialaneix, and Christine Cierco-Ayrolles. Multiple kernel self-organizing maps. *ESANN 2013*, 2013.

[23] Madalina Olteanu, Nathalie Villa-Vialaneix, and Marie Cottrell. On-line relational som for dissimilarity data. *Workshop on Self-Organizing Maps*, 2012.

[24] Alain Rakotomamonjy, Francis Bach, Yves Grandvalet, and Stéphane Canu. More efficiency in multiple kernel learning. *24th international conference on Machine learning*, pages 775–782, 2007.

[25] Gilbert Saporta. *Probabilités, analyse des données et statistique*. Editions Technip, 2006.

[26] Alexander J. Smola and Risi Kondor. Kernels and regularization on graphs. 2003.

[27] K. Steinhaeuser and N.V. Chawla. Community detection in a large real-world social network. pages 168–175, 2008.

[28] Christine Thomas and Alain Berlinet. *Reproducing kernel Hilbert spaces in Probability and Statistics*. Kluwer Academic Publishers, 2004.

[29] Jean-Philippe Vert, Bernhard Schölkopf, and Koji Tsuda. *Kernel Methods in Computational Biology*. MIT Press, 2004.

[30] Nathalie Villa-Vialaneix, Madalina Olteanu, and Christine Cierco-Ayrolles. Carte auto-organisatrice pour graphes étiquetés. *Ateliers Fouille de Grands Graphes, Extraction et Gestion des Connaissances (EGC)*, 2013.

Cinquième partie .

Annexes

A. Principaux programmes

A.1. Description du contenu des scripts

```
##############################
########## README ##########
##############################

#############
# kernels.r #
#############

Contient de nombreuses fonctions permettant de
calculer le noyau à partir d'un jeu de données

###################
# kkmeans_batch.r #
###################

Contient l'ensemble des fonctions permettant l'algortihme
kernel k-means en version batch + fonctions graphiques

#################
# kkmeans_bag.r #
#################

Contient les fonctions nécessaires pour le bagging
de Dudoit et Fridlyand

############
# online.r #
############

Contient les fonctions permettant l'algo kernel k-means
en version online avec et sans bagging + quelques illustrations
```

A.2. Les principaux scripts

A.2.1. kernels.R

```
############################################################
##### kernel computation
############################################################
## Numerical kernels
# The first three kernels are for numerical variables; they use the
   library kernlab
# Still to be implemented: tanh, laplace, bessel, anova, spline
# See also stringdot
compute.rbf.kernel <- function(d, sigma=1, sigest=TRUE) {
  if (sigest) {
    sigma <- sigest(as.matrix(d))[2]
  }
  rbf <- rbfdot(sigma=sigma)
  kernelM <- kernelMatrix(rbf, as.matrix(d))
  list("param"=list("sigma"=sigma), "kernelM"= kernelM)
}

compute.poly.kernel <- function(d, degree=1, scale=1, offset=1) {
  poly <- polydot(degree=degree, scale=scale, offset=offset)
  kernelM <- kernelMatrix(poly, as.matrix(d))
  list("param"=list("degree"=degree, "scale"=scale, "offset"=offset),
     "kernelM"= kernelM)
}

compute.vanilla.kernel <- function(d) {
  vanilla <- vanilladot()
  kernelM <- kernelMatrix(vanilla, as.matrix(d))
  list("param"=list(), "kernelM"= kernelM)
}

## Kernels for graph
# The following kernels are for graphs: they use the library igraph
   and, for the second one, the library MASS
compute.heat.kernel <- function(d, sigma=sigma, weights=NULL) {
  laplacianM <- graph.laplacian(d, norm=FALSE)
  eigenL <- eigen(laplacianM)
  kernelM <- eigen(laplacianM)$vectors%*%diag(exp(-sigma*eigen(
     laplacianM)$values))%*%t(eigen(laplacianM)$vectors)
  list("param"=list("sigma"=sigma), "kernelM"=kernelM)
}

compute.commutetime.kernel <- function(d, weights=NULL) {
  laplacianM <- graph.laplacian(d, norm=FALSE, sparse=FALSE)
  kernelM <- ginv(laplacianM)
  list("param"=list(), "kernelM"=kernelM)
}

compute.shortest.paths <- function(d) {
  shortestP <- shortest.paths(d)
  kernelM <- max(shortestP)-shortestP
  list("kernelM"=kernelM)
}
```

```
## Kernels for strings
# The following kernel is for strings: it uses the library kernlab
compute.string.kernel <- function(d, length=4, lambda=1.1, typeSK="
    spectrum", normalized=TRUE) {
  stringK <- stringdot(length=length, lambda=lambda, type=typeSK,
      normalized=normalized)
  kernelM <- kernelMatrix(stringK, d)
  list("param"=list("length"=length, "lambda"=lambda, "typeSK"=typeSK,
      "normalized"=normalized), "kernelM"=kernelM)
}

## Generic function
compute.kernel <- function(d, type="rbfdot", reduce=FALSE, sigma=1,
    sigest=TRUE, degree=1, scale=1, offset=1, weights=NULL, length=4,
    lambda=1.1, typeSK="spectrum", normalized=TRUE) {
  if (reduce) {
    d <- scale(d)
  }
  res <- switch(type,
                rbfdot=compute.rbf.kernel(d, sigma=sigma, sigest=
                    sigest),
                polydot=compute.poly.kernel(d, degree=degree, scale=
                    scale, offset=offset),
                vanilladot=compute.vanilla.kernel(d),
                heat=compute.heat.kernel(d, sigma=sigma, weights=NULL)
                    ,
                commutetime=compute.commutetime.kernel(d, weights=
                    weights),
                stringdot=compute.string.kernel(d, length=length,
                    lambda=lambda, typeSK=typeSK, normalized=normalized
                    ),
                shortest.paths=compute.shortest.paths(d)
  )
  res
}

compute.multiple.kernel <- function(KList, alphaL) {
        multipleK <- lapply(as.list(1:length(KList)), function(x){
           KList[[x]]*alphaL[x]})
        multipleK <- Reduce("+",multipleK)
        multipleK
}
```

A.2.2. kkmeans_batch.R

```
library(igraph)
library(kernlab)
library(class)
library(RColorBrewer)
library(class)

##############################
# Calcul noyau de la chaleur #
##############################
compute.heat.kernel <- function(d, sigma=sigma, weights=NULL) {
  laplacianM <- graph.laplacian(d, norm=FALSE)
```

```
  eigenL <- eigen(laplacianM)
  kernelM <- eigen(laplacianM)$vectors%*%diag(exp(-sigma*eigen(
     laplacianM)$values))%*%t(eigen(laplacianM)$vectors)
  list("param"=list("sigma"=sigma), "kernelM"=kernelM)
}

#########################
# Calcul noyau gaussien #
#########################
compute.rbf.kernel <- function(d, sigma=1, sigest=TRUE) {
  if (sigest) {
    sigma <- sigest(as.matrix(d))[2]
  }
  rbf <- rbfdot(sigma=sigma)
  kernelM <- kernelMatrix(rbf, as.matrix(d))
  list("param"=list("sigma"=sigma), "kernelM"= kernelM)
}

#########################################################
# Classe minimisant la distance entre indiv et proto #
#########################################################
assignement.mat <- function(gamma,kernel.inbag,kernel.in.out){
  gamma[gamma=='NaN'] <- 0
  f.proto <- gamma %*% kernel.inbag %*% t(gamma) #partie du pb qui
     depend que du proto
  f.proto.i <- gamma %*% kernel.in.out #depend du proto et de l'
     individu
  all.distances <- diag(f.proto) - 2*f.proto.i   # matrice de distance
      individu-proto
  winner <- apply(all.distances, 2, which.min)   # pr chaque indiv :
     cluster minimisant distance
  return(winner)
}

###################################
# Vecteur gamma pour un prototype #
###################################
representation <- function(m,cluster){
  h <- rep(0,length(cluster))
  for (taille in 1:length(cluster)){
    if(cluster[taille]==m){
      h[taille] <- 1
    }
  }
  gamma <- h/sum(h)
  gamma[is.na(gamma)==T] <- 0
  return(gamma)
}

#####################################################
# Norme au carre entre un individu et sonprototype # (pour calcul var
   intra)
#####################################################
sq.dist.to.proto <- function(indiv,num.cluster,gamma.ij,noyau){
  gamma.ij[gamma.ij=='NaN'] <- 0
  proto.c <- gamma.ij %*% noyau %*% t(gamma.ij)
  proto.c.i <- gamma.ij %*% noyau
```

```
  norm.sq <- noyau[indiv,indiv] + proto.c[num.cluster,num.cluster] - 2
      * proto.c.i[num.cluster,indiv]
  return(norm.sq)
}


####################################
# Calcul de la variance du cluster # (pour calcul var intra)
####################################
var.group <- function(num.cluster,cluster,kernel,gamma.ij){
  ind.group <- which(cluster==num.cluster) # individus appartenant a
     ce cluster
  var.g <- 0
  if (length(ind.group)!=0){ # si cluster non vide
    var.g <- sum(sapply(ind.group, sq.dist.to.proto, num.cluster,
       gamma.ij, kernel))
  }
  return(var.g)
}


############################
# Algorithme Kernel Batch #
############################
kernel.batch <- function(noyau,   # Kernel
                         M,        # Nombre de clusters voulu
                         itermax   # Nombre d'iterations maximum
                         ){

  n <- dim(noyau)[1] #nombre d'individus
  cluster <- rep(0,n)     #initialisation des clusters des individus
  cluster.prev <- rep(1,n)     #initialisation vecteur de taille n,
     different de cluster

  # (1) initialisation gamma
  gamma <- matrix(runif(M*n,0,1),nrow=M,ncol=n)
  gamma <- sweep(gamma,1,apply(gamma,1,sum),"/")

  # (2) pour le nombre d'iterations voulu
  for (iter in 1:itermax){

      if (sum(cluster.prev != cluster)!=0){     # Critere d arret

        cluster.prev <- cluster

        # (3) etape d'assignement
        cluster <- assignement.mat(gamma,noyau,noyau)

        # (4) etape de representation
        gamma <- t(sapply(1:M,representation,cluster=cluster))
      }
      else{
        break
      }
  }
#   print(c("Convergence atteinte apres ", iter, " iterations"))
#
#   var.intra.tot <- sum(sapply(1:M,var.group,cluster,noyau,gamma))
```

```
#   print(c("Variance intra : ",var.intra.tot))

  return(list("cluster"=cluster,"centers"=gamma))
}


########################
# FONCTIONS GRAPHIQUES #
########################

proj.graph <- function(the.graph,the.classif) {
  # This function allows one to build a projected graph which nodes
     are the classes contained in the.classif, having "size" the
     number of nodes in the initial graph, and whose edges are the
     number of edges between the two classes, stored in an edge
     attribute called "ew"
  adj.mat <- get.adjacency(the.graph)
  the.classes <- sort(unique(the.classif))
  the.sizes <- table(the.classif)
  the.edges <- NULL
  for (ind1 in the.classes) {
    for (ind2 in the.classes) {
      if (ind1<ind2) {
        nodes.c1 <- which(the.classif==ind1)
        nodes.c2 <- which(the.classif==ind2)
        we <- sum(adj.mat[nodes.c1,nodes.c2])/sqrt(length(nodes.c1)*
           length(nodes.c2))
        if (we!=0)
          the.edges <- rbind(the.edges,c(ind1,ind2,we))
      }
    }
  }
  the.nodes <- data.frame("name"=the.classes,"sizes"=the.sizes)
  the.nodes <- the.nodes[,c(1,3)]
  names(the.nodes) <- c("name","sizes")
  if (nrow(the.edges)>1) {
    the.proj.graph <- graph.data.frame(the.edges[,1:2],directed=F,
       vertices=the.nodes)
  } else {
    the.proj.graph <- graph.data.frame(t(the.edges[,1:2]),directed=F,
       vertices=the.nodes)
  }
  E(the.proj.graph)$ew <- the.edges[,3]
  the.proj.graph
}

plot.proj.graph <- function(the.graph, m, n, the.classif, pal="
   Spectral", scale.param=40, add.factor=NULL, factor.col, scale.edges
   =1, the.lab=NA) {
  # This function allows one to represent the projected graph on the
     grid: the size of the nodes indicates the size of the cluster and
      the edges width the sum of the weights between the two clusters
  the.grid <- somgrid(xdim=m,ydim=n,topo="rect")
  the.proj.graph <- proj.graph(the.graph, the.classif)
  where.in.grid <- match(V(the.proj.graph)$name,as.character(1:nrow(
     the.grid$pts)))
  the.layout <- the.grid$pts[where.in.grid,]
  scaling <- scale.param/max(sqrt(V(the.proj.graph)$sizes))
```

```
  par(mar=rep(0,4))
  if (is.null(add.factor)) {
    if (!is.null(pal)) {
      dist.from.first <- as.matrix(dist(the.grid$pts,diag=T,upper=T))
          [,1]
      # Scale it from 0 to 11 (integer only)
      scaled.dist <- round(dist.from.first/max(dist.from.first)*10)+1
      clust.color <- brewer.pal(11,pal)[scaled.dist]
    } else {
      clust.color <- rep("purple",nrow(the.grid$pts))
    }
    plot(the.proj.graph, layout=layout.auto, vertex.label=the.lab[
        where.in.grid], vertex.label.cex=2, vertex.label.font=2, vertex
        .label.color="black", vertex.size=scaling*sqrt(V(the.proj.graph
        )$sizes), vertex.color=clust.color[where.in.grid], edge.width=
        scale.edges*E(the.proj.graph)$ew)
  } else {
    pie.tables <- lapply(split(factor(add.factor),factor(the.classif))
        ,table)
    # stupid trick to avoid frequency table with only zeros
    plot(the.proj.graph, layout=layout.auto, vertex.shape="pie",
        vertex.pie=pie.tables, vertex.label.cex=2, vertex.label.color="
        black", vertex.label.font=2, vertex.pie.color=list(factor.col),
         layout=the.layout, vertex.label=the.lab[where.in.grid], vertex
        .size=scaling*sqrt(V(the.proj.graph)$sizes), edge.width=scale.
        edges*E(the.proj.graph)$ew)
  }
}
```

A.2.3. kkmeans_bag.R

```
#
   ##############################################################################

################################## BAGCLUST 2
   ###################################
#
   ##############################################################################


bagclust.2.V2 <- function(noyau,      # noyau
                          B,          # nb de bootstraps
                          M,          # nb clusters
                          itermax){   # nb iteration max pour kkmeans

  n <- dim(noyau)[1] # nb d individus
  n.inbag <- round(n*2/3)
  add.association <- matrix(0,ncol=n,nrow=n)
  add.together <- matrix(0,ncol=n,nrow=n)

  # repeter nboot
  for (i in 1:B){

    #tirer les bootstraps
```

```
    inbag <- sort(sample(1:n,n.inbag,replace=T)) # Individus pour ech
       de construction
    outbag <- setdiff(1:n,inbag) # Individus pour validation
    n.outbag <- length(outbag)
    noyau.inbag <- noyau[inbag,inbag]
    noyau.in.out <- noyau[inbag,outbag]

    # classer echantillon inbag
    simulboot <- kernel.batch(noyau.inbag,M,itermax)


    # predire la classe pour l echantillon de validation
    gamma <- simulboot$centers

    validation <- assignement.mat(gamma,noyau.inbag,noyau.in.out)
    validation <- c(validation,rep(0,length(unique(inbag))))[order(c(
       outbag,unique(inbag)))]

    # Matrice indiquant si 2 individus sont dans le meme echantillon
       outbag
    add.together[outbag,outbag] <- add.together[outbag,outbag]+1
    # Matrice indiquant si 2 individus sont dans le meme cluster
    for (cluster in 1:M) {
      add.association[validation==cluster,validation==cluster] <- add.
         association[validation==cluster,validation==cluster] + 1
    }
  }

  similarite <- 1-add.association/add.together  # Nouvelle matrice de
     dissimilarite
  bad.asso <- length(similarite[similarite=="NaN"])/2
  if(bad.asso > 0){
    cat("WARNING : There are ", bad.asso, " different couples with 0
       association \n")
    similarite[similarite=="NaN"] <- 1
    # which(similarite=="NaN",arr.ind=T) si l on veut recuperer les
       couples en question
  }

  # Classi a partir de la matrice de dissimilarite
  resultat.final <- select.kmeans(similarite,M,100)$cluster #
     selection du meilleur kmeans

  return(resultat.final)
}


#
    ##########################################################################

# Aggregation gamma si plusieurs fois meme individu dans l echantillon
     #
#
    ##########################################################################

aggreg.gamma <- function(indiv, inbag, gamma){
  if(inbag[indiv-1]==inbag[indiv]){
    gamma[,indiv-1] <- gamma[,indiv-1] + gamma[,indiv]
```

```
    gamma <- gamma[,-indiv]
  }
  return(gamma)
}

################################
# Selection du meilleur kmeans ##
################################
select.kmeans <- function(donnees,nbclasse,repetition){
  classi <- kmeans(donnees,centers=nbclasse)
  b <- classi$tot.withinss        # stockage de la variance intra

  for(i in 2:repetition){
    new.classi <- kmeans(donnees,centers=nbclasse)
    a <- new.classi$tot.withinss
    if(a<=b){
      b <- a
      classi <- new.classi      # si variance intra reduite, classi
         est meilleure
    }
  }
  return(classi)
}

# On cherche les "voisins" d'un individu pour une etape donnee
voisin.indiv.2 <- function(ind1, n, validation, add.association,add.
   together){
    add.association[ind1,] <-sapply(1:n,same.association.2,ind1,
       validation,add.association)
    add.together[ind1,] <-sapply(1:n,same.together.2,ind1,validation,
       add.together)
  return(list(add.association,add.together))
}

# Verification si 2 individus appartiennent a la meme classe pour l
   etape B
same.association.2 <- function(ind2, ind1, validation, add.association
   ){
  if(validation[ind2]!=0 && validation[ind1]!=0 && validation[ind2]==
     validation[ind1]){
    add.association[ind1,ind2] <- add.association[ind1,ind2] + 1
  }
  return(add.association[ind1,ind2])
}

# Verification si 2 individus appartiennent au meme echantillon pour l
   etape B
same.together.2 <- function(ind2, ind1, validation,add.together){
  if(validation[ind2]!=0 && validation[ind1]!=0){
    add.together[ind1,ind2] <- add.together[ind1,ind2] + 1
  }
  return(add.together[ind1,ind2])
}
```

A.2.4. online.R

```
library(igraph)
library(kernlab)
library(class)

calculate.derivative2 <- function(K, gamma, x, winner, m, n, r, topo="
    rect") {
  M <- rep(K[x,x],nrow(gamma))-2*gamma%*%K[,x]+diag(gamma%*%K%*%t(
      gamma))
  sum(M)
}


reduced.grad.comp <- function(ind,gradient,best,alphaL) {
  if (ind==best) {
    kept <- setdiff(which(alphaL>0),best)
    res <- sum(gradient[kept]-gradient[ind])
  } else {
    if (alphaL[ind]>0) {
      res <- gradient[best] - gradient[ind]
    } else {res <- 0}
  }
  res
}


############################
# Multi kernel with adapt #
############################
k.online <- function(noyau,
                     M,
                     itermax,
                     eps=0.3
                     ){

  n <- dim(noyau)[1] # nombre d'individus
  evol.var <- c()   # evolution variance intra totale
  c0 <- 0.2

  # Initialization of the coefficients
  # (must be positive and sum to one for all clusters)
  gamma <- matrix(runif(M*n,0,1),nrow=M,ncol=n)
  gamma <- sweep(gamma, 1, apply(gamma,1,sum), "/")

  # (2) pour le nombre d'iterations voulu
  for (iter in 1:itermax) {

    indiv <- sample(1:n,1) # individu a modifier

    # (3) etape d'affectation
    # I didn't check this one again; if something still goes wrong, I'
       ll do it
    winner <- assignement.mat(gamma,noyau,noyau)[indiv]

    # (4) etape de mise a jour proto
    delta <- rep(0,n); delta[indiv] <- 1
    gamma[winner,] <- gamma[winner,] + (eps/(1+c0*iter/M))*(delta-
       gamma[winner,])
```

```
    # evolution de la variance
    if ((iter)%%(itermax%/%100) == 0){
      cluster <- assignement.mat(gamma,noyau,noyau)
      evol.var <- c(evol.var, sum(sapply(1:M,var.group,cluster,noyau,
          gamma)))
      # critere d arret
      if(length(evol.var)>1 && evol.var[length(evol.var)]==evol.var[
          length(evol.var)-1]){
        cat("La convergence est atteinte apres environ ", iter, "
            iterations")
        break
      }
    }
  }

  plot(evol.var,axes=F);axis(2)
  cluster <- assignement.mat(gamma,noyau,noyau)

  return(list("cluster"=cluster,"centers"=gamma))
}



###################################
# Vecteur gamma pour un prototype #
###################################
representation.online <- function(m,ind.k,n){
  h <- rep(0,n)
  for (taille in 1:n){
    h[taille] <- sum(ind.k[[m]]==taille)
  }
  gamma <- h/sum(h)
  gamma[is.na(gamma)==T] <- 0
  return(gamma)
}

#########################
# bagging sur du online #
#########################

bagclust.online <- function(noyau,      # noyau
                            B,          # nb de bootstraps
                            M,          # nb clusters
                            itermax,    # nb iteration max pour kkmeans
                            eps=0.3){

  n <- dim(noyau)[1] # nb d individus
  n.inbag <- round(n*2/3)
  add.association <- matrix(0,ncol=n,nrow=n)
  add.together <- matrix(0,ncol=n,nrow=n)

  # repeter nboot
  for (i in 1:B){

    #tirer les bootstraps
    inbag <- sort(sample(1:n,n.inbag,replace=T)) # Individus pour ech
        de construction
    outbag <- setdiff(1:n,inbag) # Individus pour validation
```

```
    n.outbag <- length(outbag)
    noyau.inbag <- noyau[inbag,inbag]
    noyau.in.out <- noyau[inbag,outbag]

    # classer echantillon inbag
    simulboot <- k.online(noyau.inbag,M,itermax,eps=eps)


    # predire la classe pour l echantillon de validation
    gamma <- t(sapply(1:M,representation.online,simulboot[[2]],n.inbag
       ))


    validation <- assignement.mat(gamma,noyau.inbag,noyau.in.out)
    validation <- c(validation,rep(0,length(unique(inbag))))[order(c(
       outbag,unique(inbag)))]

    # Matrice indiquant si 2 individus sont dans le meme echantillon
       outbag
    add.together[outbag,outbag] <- add.together[outbag,outbag]+1
    # Matrice indiquant si 2 individus sont dans le meme cluster
    for (cluster in 1:M) {
      add.association[validation==cluster,validation==cluster] <- add.
         association[validation==cluster,validation==cluster] + 1
    }
  }

  similarite <- 1-add.association/add.together  # Nouvelle matrice de
     dissimilarite
  bad.asso <- length(similarite[similarite=="NaN"])/2
  if(bad.asso > 0){
    cat("WARNING : There are ", bad.asso, " different couples with 0
       association \n")
    similarite[similarite=="NaN"] <- 1
    # which(similarite=="NaN",arr.ind=T) si l on veut recuperer les
       couples en question
  }

  # Classi a partir de la matrice de dissimilarite
  resultat.final <- select.kmeans(similarite,M,100)$cluster #
     selection du meilleur kmeans

  return(resultat.final)
}


######################################################################
######################################################################
######################################################################
######################## SIMULATIONS ################################
######################################################################
######################################################################

#############
# 2 cercles #
#############

angles <- seq(0, 2*pi, length=100)
```

```
x1 <- c(6*cos(angles),1*cos(angles)) + rnorm(200,0,1)
y1 <- c(6*sin(angles),1*sin(angles)) + rnorm(200,0,1)
sigma <- 1
distance <- dist(cbind(x1,y1), method = "euclidean")
noyau2c <-compute.rbf.kernel(distance,sigma=1,sigest=T)[[2]]


par(mfrow = c(3, 3))
par(mar = c(0.5, 3.5, 1.5, 0.5))
tps <- c(0,0)
batch2c <- list()
online2c <- list()
for(i in 1:9){
  p1 <- proc.time()[3]
  batch2c[[i]] <- kernel.batch(noyau2c,2,30)$cluster
  p2 <- proc.time()[3]
  tps[1] <- tps[1] + p2 - p1
  online2c[[i]] <- k.online(noyau2c,2,50000,0.3)$cluster
  p3 <- proc.time()[3]
  tps[2] <- tps[2] + p3 - p2
}
title(main="Evolution de la variance",outer=T,line=-1)
cat("Temps moyen batch  : ", tps[1]/9)
cat("Temps moyen on-line  : ", tps[2]/9)
for(i in 1:9){
  plot(x1,y1,col=batch2c[[i]],axes=F)
}
title(main="9 initialisations en batch",outer=T,line=-1)
for(i in 1:9){
  plot(x1,y1,col=online2c[[i]],axes=F)
}
title(main="9 initialisations en on-line",outer=T,line=-1)


# avec bagging
par(mfrow = c(2, 2))
par(mar = c(0.5, 0.5, 1.5, 0.5))
tps <- c(0,0)
batch2cbag <- list()
online2cbag <- list()

for (i in 1:4){
  p1 <- proc.time()[3]
  batch2cbag[[i]] <- bagclust.2.V2(noyau2c,200,2,30)
  p2 <- proc.time()[3]
  tps[1] <- tps[1] + p2 - p1
  online2cbag[[i]] <- bagclust.online(noyau2c,100,2,10000)
  p3 <- proc.time()[3]
  tps[2] <- tps[2] + p3 - p2
}

cat("Temps batch  : ", tps[1]/4)
cat("Temps online  : ", tps[2]/4)

for(i in 1:4){
  plot(x1,y1,col=batch2cbag[[i]],axes=F)
}
title(main="Bagging apres batch",outer=T,line=-1)
```

```
for(i in 1:4){
  plot(x1,y1,col=online2cbag[[i]],axes=F)
}
title(main="Bagging apres on-line",outer=T,line=-1)


##############################
###### multi noyaux #########
##############################



## Clusters
the.clusters <- rep(1:8,rep(25,8))
the.colors <- rep(brewer.pal(8,"Set2"),rep(25,8))


## Graph
A <- matrix(0,ncol=200,nrow=200)
A[1:100,1:100][upper.tri(A[1:100,1:100])] <- rbinom(50*99, 1, 0.3)
A[101:200,101:200][upper.tri(A[101:200,101:200])] <- rbinom(50*99, 1,
    0.3)
A[1:100,101:200] <- rbinom(100^2,1, 0.01)
A <- A+t(A)
the.graph <- graph.adjacency(A,mode="undirected")
plot(the.graph, layout=layout.fruchterman.reingold, vertex.label=NA,
    vertex.color=the.colors, vertex.frame.color=the.colors, vertex.size
    =5)


## Numerical data
X <- matrix(0,nrow=200,ncol=2)
X[c(1:25,51:75,101:125,151:175),] <- rnorm(200,0,0.3)
X[c(26:50,76:100,126:150,176:200),] <- rnorm(200,1,0.3)
plot(X,col=the.colors, pch=19, main="",xlab="",ylab="")


## Factors
Y <- matrix(0,nrow=200,ncol=2)
Y[c(1:50,101:150),1] <- 1
Y[c(51:100,151:200),2] <- 1



cosine.kernel <- function(kernel){
  n <- dim(kernel)[1]
  new.kernel <- matrix(0,nrow=n,ncol=n)
  for (i in 1:n){
    for(j in 1:n){
      new.kernel[i,j] <- kernel[i,j] / sqrt(kernel[i,i]*kernel[j,j])
    }
  }
  return(new.kernel)
}


#### Multiple kernel
```

```
sigma=1
KList1 <- list()

print("Gaussian kernel...")
KList1[[1]] <- compute.kernel(X, sigest=TRUE)$kernelM
KList1[[1]] <- cosine.kernel(KList1[[1]])

print("Commute time kernel")
KList1[[2]] <- compute.kernel(the.graph, type="commutetime")$kernelM
KList1[[2]] <- cosine.kernel(KList1[[2]])

print("Factor kernel")
KList1[[3]] <- compute.rbf.kernel(Y, sigma=sigma, sigest=TRUE)$kernelM
KList1[[3]] <- cosine.kernel(KList1[[3]])

alpha <- c(1/3,1/3,1/3)

#### Combined kernel
noyau.combine <- compute.multiple.kernel(KList1,alpha)


######################
# resultats
######################

# sans bagging
par(mfrow = c(3, 3))
par(mar = c(0.5, 3.5, 1.5, 0.5))
tps <- c(0,0)
batchcomb <- list()
onlinecomb <- list()
for(i in 1:5){
  p1 <- proc.time()[3]
  batchcomb[[i]] <- kernel.batch(noyau.combine,8,30)$cluster
  p2 <- proc.time()[3]
  tps[1] <- tps[1] + p2 - p1
  onlinecomb[[i]] <- k.online(noyau.combine,8,100000,0.3)$cluster
  p3 <- proc.time()[3]
  tps[2] <- tps[2] + p3 - p2
}
title(main="Evolution de la variance",outer=T,line=-1)
cat("Temps moyen batch  : ", tps[1]/5)
cat("Temps moyen on-line  : ", tps[2]/5)
for(i in 1:5){
  print(table(the.clusters,batchcomb[[i]]))
}
for(i in 1:5){
  print(table(the.clusters,onlinecomb[[i]]))
}

# for(i in 1:5){
#   print(xtable(table(the.clusters,batchcomb[[i]])))
# }
# for(i in 1:5){
#   print(xtable(table(the.clusters,onlinecomb[[i]])))
# }
```

```
# avec bagging
par(mfrow = c(3, 3))
par(mar = c(0.5, 0.5, 1.5, 0.5))
tps <- c(0,0)
batchcomb <- list()
onlinecomb <- list()
for(i in 1:3){
  p1 <- proc.time()[3]
  batchcomb[[i]] <- bagclust.2.V2(noyau.combine,200,8,30)
  p2 <- proc.time()[3]
  tps[1] <- tps[1] + p2 - p1
  onlinecomb[[i]] <- bagclust.online(noyau.combine,200,8,10000)
  p3 <- proc.time()[3]
  tps[2] <- tps[2] + p3 - p2
}
cat("Temps moyen batch  : ", tps[1]/5)
cat("Temps moyen on-line  : ", tps[2]/5)
for(i in 1:3){
  print(table(the.clusters,batchcomb[[i]]))
}
for(i in 1:3){
  print(table(the.clusters,onlinecomb[[i]]))
}
# for(i in 1:3){
#   print(xtable(table(the.clusters,batchcomb[[i]])))
# }
# for(i in 1:3){
#   print(xtable(table(the.clusters,onlinecomb[[i]])))
# }




##############
# 3 cercles #
##############


angles <- seq(0, 2*pi, length=100)
x1 <- c(12*cos(angles),5*cos(angles),1*cos(angles)) + rnorm(300,0,1)
y1 <- c(12*sin(angles),5*sin(angles),1*sin(angles)) + rnorm(300,0,1)
sigma <- 1
distance <- dist(cbind(x1,y1), method = "euclidean")
noyau3c <-compute.rbf.kernel(distance,sigma=1,sigest=T)[[2]]


# sans bagging
par(mfrow = c(3, 3))
par(mar = c(0.5, 3.5, 1.5, 0.5))
tps <- c(0,0)
batch3c <- list()
online3c <- list()
for(i in 1:9){
  p1 <- proc.time()[3]
  batch3c[[i]] <- kernel.batch(noyau3c,3,30)$cluster
  p2 <- proc.time()[3]
  tps[1] <- tps[1] + p2 - p1
```

```
  online3c[[i]] <- k.online(noyau3c,3,100000,0.3)$cluster
  p3 <- proc.time()[3]
  tps[2] <- tps[2] + p3 - p2
}
title(main="Evolution de la variance",outer=T,line=-1)
cat("Temps moyen batch  : ", tps[1]/9)
cat("Temps moyen batch  : ", tps[2]/9)
for(i in 1:9){
  plot(x1,y1,col=batch3c[[i]],axes=F)
}
title(main="9 initialisations en batch",outer=T,line=-1)

for(i in 1:9){
  plot(x1,y1,col=online3c[[i]],axes=F)
}
title(main="9 initialisations en on-line",outer=T,line=-1)



# avec bagging
par(mfrow = c(2, 2))
par(mar = c(0.5, 0.5, 1.5, 0.5))
tps <- c(0,0)
batch3cbag <- list()
online3cbag <- list()

for (i in 1:4){
  p1 <- proc.time()[3]
  batch3cbag[[i]] <- bagclust.2.V2(noyau3c,200,3,30)
  p2 <- proc.time()[3]
  tps[1] <- tps[1] + p2 - p1
  online3cbag[[i]] <- bagclust.online(noyau3c,20,3,1000,0.3)
  p3 <- proc.time()[3]
  tps[2] <- tps[2] + p3 - p2
}

cat("Temps batch  : ", tps[1]/4)
cat("Temps online  : ", tps[2]/4)

for(i in 1:4){
  plot(x1,y1,col=batch3cbag[[i]],axes=F)
}
title(main="Bagging apres batch",outer=T,line=-1)

for(i in 1:4){
  plot(x1,y1,col=online3cbag[[i]],axes=F)
}
title(main="Bagging apres on-line",outer=T,line=-1)
```