THÈSE DE DOCTORAT DE L'UNIVERSITÉ PARIS 6
Pierre et Marie Curie

Spécialité :
Électronique
EDITE

Présentée par
Aurélie GOULON-SIGWALT-ABRAM

Pour obtenir le grade de
DOCTEUR de L'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

Une nouvelle méthode d'apprentissage de données structurées : applications à l'aide à la découverte de médicaments

Soutenue le 21 mai 2008

Devant le jury composé de :

M. Jean-Pierre DOUCET	Professeur	Rapporteurs
M. Manuel SAMUELIDES	Professeur	
Mme Michèle SEBAG	Directeur de recherche CNRS	Examinateurs
M. Patrick GALLINARI	Professeur Paris 6	
M. Jacques PROST	Directeur de l'ESPCI	
M. Gérard DREYFUS	Professeur	Directeurs de thèse
M. Arthur DUPRAT	Maître de conférences	

REMERCIEMENTS

Je tiens tout d'abord à remercier l'ensemble du Laboratoire d'Électronique de l'ESPCI pour l'accueil qui m'a été réservé. J'ai trouvé au sein du Laboratoire une ambiance particulièrement favorable pour mener à bien ce travail. Mes remerciements vont en particulier à mes deux directeurs de thèse, Gérard Dreyfus et Arthur Duprat, pour leurs conseils, leurs encouragements, leur patience ainsi que leur disponibilité. Votre soutien a été pour moi très précieux durant ces années que j'ai passées en votre compagnie. Merci également à Yacine et Ginette pour leur gentillesse, à Pierre pour sa bonne humeur indéfectible, à Haini, Iness, Paola et Thomas pour le dynamisme qu'ils ont su, j'espère, me transmettre. Je tiens également à remercier Rémi et Bruce pour leur énergie ainsi que tous leurs conseils. Je garderai vraiment un excellent souvenir des années passées parmi vous tous! De nombreux étudiants ont contribué à ces recherches au cours de ces années. Je vous remercie tous pour les nouvelles idées que vous avez apportées, vous avez tous su vous investir et j'ai travaillé avec vous avec beaucoup de plaisir.

Mes remerciements vont également à Janine Cossy, ainsi qu'à l'ensemble du Laboratoire de Chimie Organique, pour leur aide, ainsi que pour les défis de modélisation qu'ils m'ont proposés, et qui m'ont permis de trouver de nouveaux développements des Graph Machines. Merci également à Jacques Lewiner pour m'avoir encouragé tout au long de des recherches, et pour l'exemple que vous représentez.

J'exprime toute ma reconnaissance à Jean-Luc Ploix et Patrice Kiener, de la société Netral, pour leur efficacité et leur réactivité, qui ont rendu possible l'implémentation informatique des Graph Machines, ainsi que pour tous les conseils qu'ils m'ont apportés.

Je remercie Jean-Pierre Doucet et Manuel Samuelides d'avoir accepté d'être rapporteurs, et pour les conseils ainsi que les idées qu'ils m'ont apportés. Merci également à Michèle Sebag, Patrick Gallinari et Jacques Prost de me faire l'honneur de participer au jury.

Je tiens à adresser mes remerciements à l'IFP, en particulier à Aziz Faraj, Marc Jacquin, et Fabien Porcheron pour l'intérêt porté à ce travail de thèse et les nouveaux défis que vous me présentez. Je ferai tout pour que les Graph Machines répondent à vos attentes !

J'ai eu l'occasion, lors d'un stage effectué chez Eli Lilly, de découvrir la recherche de médicaments telle qu'elle est menée dans un laboratoire industriel. Je remercie donc toute l'équipe qui m'a si bien accueillie, et en particulier Mike, Nicolas, Véronique et Michael.

Merci également à Tulay, ta gentillesse et ta générosité ont énormément compté pour moi. J'espère que nous aurons prochainement l'occasion de nous rencontrer lors de futures conférences. Merci aussi à tous les chercheurs que j'ai rencontrés lors de congrès, pour leurs remarques et leurs suggestions, qui m'ont permis d'avancer dans ce travail de recherche.

Je remercie tout particulièrement ma famille ainsi que Monique, Gérard et Florian, pour le soutien et l'affection qu'ils m'ont apportés ces dernières années. Vous avez su m'entourer et m'épauler à tout moment, je vous dois énormément. Merci Nicolas, pour tous tes encouragements, ta patience, tout ce que tu m'apportes. Enfin, je tiens particulièrement à dédier cette thèse à ma grand-mère, qui aurait aimé être présente lors de sa soutenance.

RÉSUMÉ :

La modélisation de propriétés et d'activités de molécules constitue un champ de recherche important, qui permet par exemple de guider la synthèse de médicaments. Les méthodes traditionnelles de modélisation établissent des relations non linéaires entre les propriétés étudiées et les caractéristiques structurelles des molécules, appelées descripteurs. Leurs principaux inconvénients résident dans la difficulté du choix des descripteurs et leur calcul préalable.

Nous avons mis au point une nouvelle technique de modélisation qui s'affranchit de ces problèmes, en établissant une relation directe entre la structure des données et la propriété modélisée. L'apprentissage s'effectue non plus à partir de vecteurs de données, mais à partir de graphes. Les molécules peuvent en effet être représentées par des graphes, qui tiennent compte des liaisons chimiques, de la nature des atomes ou encore de la stéréochimie du composé initial. Chaque graphe de la base étudiée est alors associé à une fonction de même structure mathématique, appelée *graph machine*, obtenue par combinaison de fonctions paramétrées identiques. Ces paramètres sont alors déterminés par apprentissage.

Nous montrons que les techniques traditionnelles de sélection de modèle peuvent être utilisées dans le cadre des *graph machines* ; elles permettent d'évaluer les capacités en généralisation des modèles proposés, mais aussi de détecter les catégories de molécules sous-représentées dans la base d'apprentissage, et d'estimer les intervalles de confiance des prédictions. De très bons résultats ont été obtenus par l'utilisation de cette technique sur un grand nombre de bases de données de propriétés ou d'activités moléculaires.

TITLE :

A new approach to learning from structured data and its applications to computer-aided drug design.

MOTS-CLÉS :

Apprentissage, données structurées, graph machines, graphes, sélection de modèle, toxicité, ecdystéroïdes, épothilone.

KEYWORDS :

Machine learning, structured data, graph machines, graphs, model selection, toxicity, ecdysteroids, epothilone.

Thèse préparée au Laboratoire d'Électronique de l'ESPCI, 10 Rue Vauquelin, 75231 Paris Cedex 05.

SOMMAIRE

INTRODUCTION

Découvrir de nouveaux médicaments de la manière la plus efficace et la moins coûteuse possible constitue un enjeu majeur pour les années à venir. Il est admis que, en moyenne, pour une molécule qui arrive sur le marché en tant que médicament innovant, 10 000 molécules sont synthétisées et testées. De plus, le développement d'un médicament demande généralement entre 10 et 15 ans de recherches. Il s'agit en effet de trouver une molécule qui doit à la fois présenter des propriétés thérapeutiques particulières, et posséder le minimum d'effets secondaires indésirables. Le prix de revient d'un médicament est essentiellement dû à ces synthèses longues, coûteuses et finalement inutiles. Pour cette raison, l'industrie pharmaceutique s'oriente vers de nouvelles méthodes de recherche, qui consistent à prédire les propriétés et activités de molécules avant même que celles-ci ne soient synthétisées. Deux disciplines de la « chimie computationnelle » se sont développées en réponse à ce besoin : les relations structure-activité ou QSAR (Quantitative Structure-Activity Relationships), et les relations structure-propriété ou QSPR (Quantitative Structure-Property Relationships). Elles consistent essentiellement en la recherche de similitudes entre molécules dans de grandes bases de données de molécules existantes dont les propriétés sont connues. La découverte d'une telle relation permet de prédire les propriétés physiques et chimiques et l'activité biologique de composés, de développer de nouvelles théories ou de comprendre les phénomènes observés. Elle permet également de guider les synthèses de nouvelles molécules, sans avoir à les réaliser, ou à analyser des familles entières de composés.

Les relations entre les structures des molécules et leurs propriétés ou activités sont généralement établies à l'aide de méthodes de modélisation par apprentissage statistique. Les techniques usuelles reposent sur la caractérisation des molécules par un ensemble de descripteurs, nombres réels mesurés ou calculés à partir des structures moléculaires. Il est alors possible d'établir une relation entre ces descripteurs et la grandeur modélisée. Ces méthodes présentent cependant plusieurs inconvénients. Elles nécessitent en effet la sélection des descripteurs pertinents ainsi que leur calcul. De plus, les molécules sont des structures, qui peuvent être représentées par des graphes ; leur représentation par un vecteur de données induit donc une perte d'information, qui peut avoir une influence sur la qualité des modèles.

Dans le cadre de cette thèse, nous avons mis au point une nouvelle méthode de modélisation qui s'affranchit de la caractérisation des molécules par des vecteurs de nombres réels, et établit directement des relations entre des graphes et des nombres. Au-delà de l'application à la modélisation de propriétés de molécules, cette technique s'avère donc être une nouvelle méthode de modélisation prédictive à partir de données structurées. Nous avons baptisé « graph machines » les modèles obtenus par cette méthode, qui permettent de faire de

la régression et de la classification à partir de graphes, par opposition aux traditionnelles « vector machines » qui effectuent les mêmes tâches à partir de vecteurs de descripteurs.

Le premier chapitre de cette thèse présente les méthodes traditionnelles de QSAR et de QSPR. Nous rappelons les principaux types de descripteurs, les problèmes liés à leur calcul et à leur sélection, ainsi que les principales techniques conventionnelles d'apprentissage et de sélection de modèle.
Le chapitre suivant introduit la notion de graphe, et décrit la genèse des *graph machines*, fonctions de même structure mathématique que les graphes auxquels elles sont associées. Nous montrons alors comment ces fonctions permettent d'établir une relation entre des données structurées et des nombres.
Le troisième chapitre présente l'utilisation de la méthode des *graph machines* pour la prédiction de propriétés et d'activités moléculaires, et décrit la méthodologie de conception de modèles que nous avons élaborée. Nous montrons notamment que les techniques usuelles de sélection de modèle peuvent être étendues aux *graph machines*.
Le chapitre 4 est consacré aux applications de notre méthode à la modélisation de propriétés et d'activités moléculaires. Nous y présentons la prédiction de propriétés permettant d'évaluer le devenir des molécules dans l'organisme, et de différentes activités pharmacologiques et toxiques. Nous soulignons ainsi la capacité des *graph machines* à apprendre différentes propriétés au prix d'un seul apprentissage. Nous montrons également que les modélisations obtenues se révèlent généralement de meilleure qualité que les modélisations par les méthodes traditionnelles.
Les annexes reproduisent plusieurs articles publiés durant ma thèse.

CHAPITRE 1
La prédiction de propriétés et d'activités de molécules

Les premiers essais de modélisation d'activités de molécules datent de la fin du 19ème siècle, lorsque Crum-Brown et Frazer [1] postulèrent que l'activité biologique d'une molécule est une fonction de sa constitution chimique. Mais ce n'est qu'en 1964 que furent développés les modèles de "contribution de groupes", qui constituent les réels débuts de la modélisation QSAR. Depuis, l'essor de nouvelles techniques de modélisation par apprentissage, linéaires d'abord, puis non linéaires, ont permis la mise en place de nombreuses méthodes ; elles reposent pour la plupart sur la recherche d'une relation entre un ensemble de nombres réels, descripteurs de la molécule, et la propriété ou l'activité que l'on souhaite prédire.
Nous montrerons tout d'abord comment les molécules peuvent être représentées par des vecteurs de réels, et comment ces descripteurs sont sélectionnés. Nous introduirons ensuite les outils de modélisation sans contrainte les plus utilisés, c'est-à-dire la régression linéaire multiple et la régression non linéaire à l'aide de réseaux de neurones, qui sont fondés sur le calcul de descripteurs. Nous présenterons le problème de la sélection de modèle, ainsi que les stratégies les plus efficaces pour le résoudre. Nous décrirons alors la théorie statistique de l'apprentissage de Vapnik et les méthodes de modélisation sous contraintes, ainsi que leurs applications en QSAR et QSPR. Enfin, de nouvelles méthodes de modélisation, telles que la méthode CoMFA, mises au point pour la modélisation d'activités biologiques, seront présentées.

I- Les descripteurs : sélection, calcul ou mesure, problèmes en résultant

De nombreuses recherches ont été menées, au cours des dernières décennies, pour trouver la meilleure façon de représenter l'information contenue dans la structure des molécules, et ces structures elles-mêmes, en un ensemble de nombres réels appelés descripteurs ; une fois que ces nombres sont disponibles, il est possible d'établir une relation entre ceux-ci et une propriété ou activité moléculaire, à l'aide d'outils de modélisation classiques. Ces descripteurs numériques réalisent de ce fait un codage de l'information chimique en un vecteur de réels. On en dénombre aujourd'hui plus de 3000 types, qui quantifient des caractéristiques physico-chimiques ou structurelles de molécules. Ils peuvent être obtenus de manière empirique ou non-empirique, mais les descripteurs calculés, et non mesurés, sont à privilégier : ils permettent en effet d'effectuer des prédictions sans avoir à synthétiser les molécules, ce qui est un des objectifs de la modélisation. Il existe cependant

quelques descripteurs mesurés : il s'agit généralement de données expérimentales plus faciles à mesurer que la propriété ou l'activité à prédire (coefficient de partage eau-octanol [2], polarisabilité, ou potentiel d'ionisation).
Avant toute modélisation, il est nécessaire de calculer ou de mesurer un grand nombre de descripteurs différents, car les mécanismes qui déterminent l'activité d'une molécule ou une de ses propriétés sont fréquemment mal connus. Il faut ensuite sélectionner parmi ces variables celles qui sont les plus pertinentes pour la modélisation.

I.1 - Les descripteurs moléculaires

Nous allons présenter les descripteurs moléculaires les plus courants, en commençant par les descripteurs les plus simples, qui nécessitent peu de connaissances sur la structure moléculaire, mais véhiculent peu d'informations. Nous verrons ensuite comment les progrès de la modélisation moléculaire ont permis d'accéder à la structure 3D de la molécule, et de calculer des descripteurs à partir de cette structure.

Les descripteurs 1D sont accessibles à partir de la formule brute de la molécule (par exemple C_6H_6O pour le phénol), et décrivent des propriétés globales du composé. Il s'agit par exemple de sa composition, c'est-à-dire les atomes qui le constituent, ou de sa masse molaire. On peut remarquer que ces descripteurs ne permettent pas de distinguer les isomères de constitution.

Les descripteurs 2D sont calculés à partir de la formule développée de la molécule. Ils peuvent être de plusieurs types.

- Les ***indices constitutionnels*** caractérisent les différents composants de la molécule. Il s'agit par exemple du nombre de liaisons simples ou multiples, du nombre de cycles…
- Les ***indices topologiques*** peuvent être obtenus à partir de la structure 2D de la molécule, et donnent des informations sur sa taille, sa forme globale et ses ramifications. Les plus fréquemment utilisés sont l'indice de Wiener [3], l'indice de Randić [4], l'indice de connectivité de valence de Kier-Hall [5] et l'indice de Balaban [6]. L'indice de Wiener permet de caractériser le volume moléculaire et la ramification d'une molécule : si l'on appelle distance topologique entre deux atomes le plus petit nombre de liaisons séparant ces deux atomes, l'indice de Wiener est égal à la somme de toutes les distances topologiques entre les différentes paires d'atomes de la molécule. L'indice de Randić est un des descripteurs les plus utilisés ; il peut être interprété comme une mesure de l'aire de la molécule accessible au solvant.

Ces descripteurs 2D reflètent bien les propriétés physiques dans la plupart des cas, mais sont insuffisants pour expliquer de façon satisfaisante certaines propriétés ou activités, telles que les activités biologiques. Des descripteurs, accessibles à partir de la structure 3D des

molécules, ont pu être calculés grâce au développement des techniques instrumentales et de nouvelles méthodes théoriques.

Les descripteurs 3D d'une molécule sont évalués à partir des positions relatives de ses atomes dans l'espace, et décrivent des caractéristiques plus complexes; leurs calculs nécessitent donc de connaître, le plus souvent par modélisation moléculaire empirique ou *ab initio*, la géométrie 3D de la molécule. Ces descripteurs s'avèrent donc relativement coûteux en temps de calcul, mais apportent davantage d'informations, et sont nécessaires à la modélisation de propriétés ou d'activités qui dépendent de la structure 3D. On distingue plusieurs familles importantes de descripteurs 3D :

- Les ***descripteurs géométriques*** les plus importants sont le volume moléculaire, la surface accessible au solvant, le moment principal d'inertie.
- Les ***descripteurs électroniques*** permettent de quantifier différents types d'interactions inter- et intramoléculaires, de grande influence sur l'activité biologique de molécules. Le calcul de la plupart de ces descripteurs nécessite la recherche de la géométrie pour laquelle l'énergie stérique est minimale, et fait souvent appel à la chimie quantique. Par exemple, les énergies de la plus haute orbitale moléculaire occupée et de la plus basse vacante sont des descripteurs fréquemment sélectionnés. Le moment dipolaire, le potentiel d'ionisation, et différentes énergies relatives à la molécule sont d'autres paramètres importants.
- ***Descripteurs spectroscopiques*** : les molécules peuvent être caractérisées par des mesures spectroscopiques, par exemples par leurs fonctions d'onde vibrationnelles. En effet, les vibrations d'une molécule dépendent de la masse des atomes et des forces d'interaction entre ceux-ci ; ces vibrations fournissent donc des informations sur la structure de la molécule et sur sa conformation. Les spectres infrarouges peuvent être obtenus soit de manière expérimentale, soit par calcul théorique, après recherche de la géométrie optimale de la molécule. Ces spectres sont alors codés en vecteurs de descripteurs de taille fixe. Le descripteur EVA [7] est ainsi obtenu à partir des fréquences de vibration de chaque molécule. Les descripteurs de type *MoRSE* [8] *(Molecule Representation of Structures based on Electron diffraction)* sont calculés à partir d'une simulation du spectre infrarouge ; ils font appel au calcul des intensités théoriques de diffraction d'électrons.

Par ailleurs, le calcul de certains descripteurs demande une étape préliminaire d'alignement des molécules, pour les placer dans une orientation commune. Nous reviendrons sur ces descripteurs, qui sont souvent spécifiques à certaines méthodes *(CoMFA, CoMSIA...)* dans la section IV.2 de ce chapitre.

I.2 - Réduction du nombre de variables

Comme nous l'avons rappelé, un grand nombre de descripteurs différents sont collectés pour la modélisation d'une grandeur donnée, car les facteurs déterminants du processus étudié ne sont a priori pas connus. Cependant, les descripteurs envisagés n'ont pas tous une influence significative sur la grandeur modélisée, et les variables ne sont pas toujours mutuellement indépendantes. De plus, le nombre de descripteurs, c'est-à-dire la dimension du vecteur d'entrée, détermine la dimension du vecteur des paramètres à ajuster. Si cette dimension est trop importante par rapport au nombre d'exemples de la base d'apprentissage, le modèle risque d'être surajusté à ces exemples, et incapable de prédire la grandeur modélisée sur de nouvelles observations (voir le paragraphe II.3 de ce chapitre).
Il est donc nécessaire de réduire la dimensionnalité des variables d'entrée. Plusieurs approches sont possibles pour résoudre ce problème :

- réduire la dimension de l'espace des entrées ;
- remplacer les variables corrélées par de nouvelles variables synthétiques, obtenues à partir de leurs combinaisons ;
- sélectionner les variables les plus pertinentes.

Nous allons maintenant décrire les méthodes les plus fréquemment utilisées.

I.2.1 - L'analyse en composantes principales

L'analyse en composantes principales (ou ACP) [9], est une technique d'analyse de données utilisée pour réduire la dimension de l'espace de représentation des données. Contrairement à d'autres méthodes de sélection, celle-ci porte uniquement sur les variables, indépendamment des grandeurs que l'on cherche à modéliser. Les variables initiales sont remplacées par de nouvelles variables, appelées composantes principales, deux à deux non corrélées, et telles que les projections des données sur ces composantes soient de variance maximale. Elles peuvent être classées par ordre d'importance.
Considérons un ensemble de n observations, représentées chacune par p données. Ces observations forment un nuage de n points dans $\mathbb{R}^p$. Le principe de l'ACP est d'obtenir une représentation approchée des variables dans un sous-espace de dimension k plus faible, par projection sur des axes bien choisis ; ces axes principaux sont ceux qui maximisent l'inertie du nuage projeté, c'est-à-dire la moyenne pondérée des carrés des distances des points projetés à leur centre de gravité. La maximisation de l'inertie permet de préserver au mieux la répartition des points. Dès lors, les n composantes principales peuvent être représentées dans l'espace sous-tendu par ces axes, par une projection orthogonale des n vecteurs d'observations sur les k axes principaux. Puisque les composantes principales sont des combinaisons linéaires des variables initiales, l'interprétation du rôle de chacune de ces composantes reste possible. Il suffit en effet de déterminer quels descripteurs d'origine leur sont le plus fortement corrélés. Les variables obtenues peuvent ensuite être utilisées en tant que nouvelles variables du modèle. Par exemple, la régression sur composantes principales [10] (ou PCR) est une

méthode de modélisation dont la première étape est une analyse en composantes principales, suivie d'une régression linéaire multiple (dont le principe est présenté dans le paragraphe II.1).

I.2.2 - La méthode de régression des moindres carrés partiels

La régression des ***moindres carrés partiels*** [11, 12] (MCP, ou PLS) est également une méthode statistique utilisée pour construire des modèles prédictifs lorsque le nombre de variables est élevé et que celles-ci sont fortement corrélées. Cette méthode utilise à la fois des principes de l'analyse en composantes principales et de la régression multilinéaire. Elle consiste à remplacer l'espace initial des variables par un espace de plus faible dimension, sous-tendu par un petit nombre de variables appelées « variable latentes », construites de façon itérative. Les variables retenues sont orthogonales (non corrélées), et sont des combinaisons linéaires des variables initiales. Les variables latentes sont obtenues à partir des variables initiales, mais en tenant compte de leur corrélation avec la variable modélisée, contrairement aux variables résultant de l'analyse en composantes principales. Elles doivent ainsi expliquer le mieux possible la covariance entre les entrées et la sortie. Elles sont alors les nouvelles variables explicatives d'un modèle de régression classique, telles que la régression linéaire multiple.

I.2.3 - Sélection des variables pertinentes

L'analyse en composantes principales a pour but de réduire les corrélations entre les variables, mais cette étape de réduction est indépendante de la grandeur modélisée. Or, les variables calculées n'ont pas nécessairement une influence sur cette grandeur. Il est nécessaire d'éliminer celles dont l'influence est inférieure à celle du bruit, et de sélectionner uniquement les plus pertinentes d'entre elles. Il est possible, à partir de p descripteurs, de former 2^p sous-ensembles de variables, donc 2^p modèles, dont il faudrait comparer les performances pour déterminer le meilleur jeu de descripteurs. Cette méthode permet d'envisager toutes les combinaisons possibles de descripteurs, mais sa mise en œuvre est très lourde. Des approches alternatives ont donc été développées.

La ***sélection progressive*** consiste à incorporer les variables au modèle une à une, en sélectionnant, à chaque étape, la variable dont la corrélation partielle avec la grandeur modélisée est la plus élevée. À l'inverse, lors de l'***élimination progressive***, on débute la modélisation avec l'ensemble des descripteurs, en les éliminant un par un jusqu'à obtenir le meilleur jeu de composantes.

La ***sélection pas à pas*** est une combinaison des deux méthodes évoquées précédemment. Les variables sont incorporées une à une dans le modèle, par sélection progressive. Cependant, à chaque étape, on vérifie que les corrélations partielles des variables précédemment introduites sont encore significatives.

Une méthode plus efficace est celle du ***descripteur sonde*** [13, 14]. Cette sélection est généralement effectuée à partir d'un modèle linéaire en ses paramètres, par exemple un modèle polynomial. En effet, la pertinence d'une variable est indépendante du modèle

considéré, et les variables ainsi sélectionnées peuvent être utilisées comme variables de modèles plus complexes. La méthode consiste à ajouter un descripteur aléatoire, appelé « descripteur sonde », à la liste des variables candidates. Ces variables sont alors classées selon leur pertinence par une méthode d'orthogonalisation. On se fixe le risque que l'on est prêt à accepter de conserver une variable candidate alors qu'elle est moins bien classée que le descripteur sonde, et l'on en déduit le rang, dans le classement, au-delà duquel les variables sont éliminées.

II - Modélisation par optimisation sans contrainte

La modélisation par apprentissage consiste à trouver le jeu de paramètres qui conduit à la meilleure approximation possible de la fonction de régression, à partir des couples entrées / sorties constituant l'***ensemble d'apprentissage***. Le plus souvent, ces couples sont constitués d'un ensemble de vecteurs de variables (descripteurs dans le cas de molécules) $\left\{\mathbf{x}^i, i=1...N\right\}$, et d'un ensemble de mesures de la grandeur à modéliser $\left\{y\left(\mathbf{x}^i\right), i=1...N\right\}$. La détermination des valeurs de ces paramètres nécessite la mise en œuvre de méthodes d'optimisation qui diffèrent selon le type de modèle choisi. Nous allons tout d'abord présenter les principaux types de modèles faisant appel à l'optimisation paramétrique, sans contrainte, qui consiste à déterminer les paramètres optimaux par minimisation directe d'une fonction de coût par rapport aux paramètres du modèle.

II.1 - Régression linéaire multiple (MLR)

La régression linéaire multiple est la méthode la plus simple de modélisation. Elle consiste à rechercher une équation linéaire par rapport à ses paramètres, reliant la variable à modéliser y au vecteur d'entrées $\mathbf{x}=\left\{x_k, k=1...q\right\}$. Ces entrées peuvent être soit les variables primaires, soit des fonctions non paramétrées, ou à paramètres fixés, de ces variables (par exemple des monômes). L'équation linéaire recherchée est de la forme :

$$g\left(\mathbf{x},\boldsymbol{\theta}\right)=\sum_{k=1}^{q}\theta_k x_k=\mathbf{X}\boldsymbol{\theta}$$

où $\boldsymbol{\theta}=\left\{\theta_k, k=1...q\right\}$ est le vecteur des paramètres ; $\mathbf{X}$, matrice des observations de taille (N, q), est définie comme la matrice dont les éléments de la colonne k prennent pour valeurs les N mesures de la variable k. Pour chaque élément i de la base d'apprentissage, le résidu est défini comme la différence entre la valeur de la grandeur à modéliser pour cet élément i et l'estimation du modèle :

$$R_i = y^i - g(\mathrm{x}^i,\boldsymbol{\theta}) \qquad (1)$$

L'apprentissage est réalisé par minimisation de la fonction de coût des moindres carrés, qui mesure l'ajustement du modèle *g* aux données d'apprentissage :

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{N} \left(R_i\right)^2 = \sum_{i=1}^{N} \left[y^i - g\left(\mathbf{x}^i, \boldsymbol{\theta}\right) \right]^2 = \left\| \mathbf{y} - \mathbf{X}\boldsymbol{\theta} \right\|^2 \quad (2)$$

La fonction $J(\boldsymbol{\theta})$ est une fonction positive quadratique en $\boldsymbol{\theta}$: son minimum est unique. Il est donné par :

$$\boldsymbol{\theta}_{mc} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \quad (3)$$

Les paramètres θ_k obtenus sont appelés coefficients de régression partielle ; chacun d'eux mesure l'effet de la variable explicative x_k concernée sur la propriété modélisée lorsque les autres variables explicatives sont maintenues constantes.
La régression linéaire est facile à mettre en œuvre, et les coefficients θ_k obtenus peuvent être interprétés : ils mesurent l'influence de chacune des variables sur la grandeur étudiée. Cependant, il est souvent nécessaire d'avoir recours à des modèles de plus grande complexité.

II.2 - Réseaux de neurones

Les réseaux de neurones formels [15] étaient, à l'origine, une tentative de modélisation mathématique des systèmes nerveux, initiée dès 1943 par McCulloch et Pitts [16].
Un ***neurone formel*** est une fonction non linéaire paramétrée, à valeurs bornées, de variables réelles. Le plus souvent, les neurones formels réalisent une combinaison linéaire des entrées reçues, puis appliquent à cette valeur une « fonction d'activation » *f*, généralement non linéaire. La valeur obtenue *y* est la sortie du neurone. Un neurone formel est ainsi représenté sur la Figure 1.

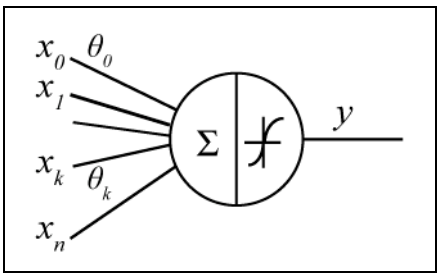


Figure 1 : Représentation d'un neurone formel

Les $\left\{x_k\right\}_{k=1\ldots n}$ sont les variables, ou ***entrées*** du neurone, et les $\left\{\theta_k\right\}_{k=0\ldots n}$ sont les ***paramètres***, également appelés synapses ou poids. Le paramètre θ_0 est le paramètre associé à une entrée fixée à 1, appelée biais. L'équation du neurone est donc :

$$y = f\left(\theta_0 + \sum_{k=1}^{n} \theta_k x_k\right)$$

Les fonctions d'activation les plus couramment utilisées sont la fonction tangente hyperbolique, la fonction sigmoïde et la fonction identité.

Les neurones seuls réalisent des fonctions assez simples, et c'est leurs compositions qui permettent de construire des fonctions aux propriétés particulièrement intéressantes. On appelle ainsi ***réseau de neurones*** une composition de fonctions « neurones » définies ci-dessus.
La Figure 2 représente un réseau de neurones non bouclé, organisé en couches (perceptron multicouche), qui comporte N_e variables, une couche de N_c neurones cachés, et N_s neurones de sortie.

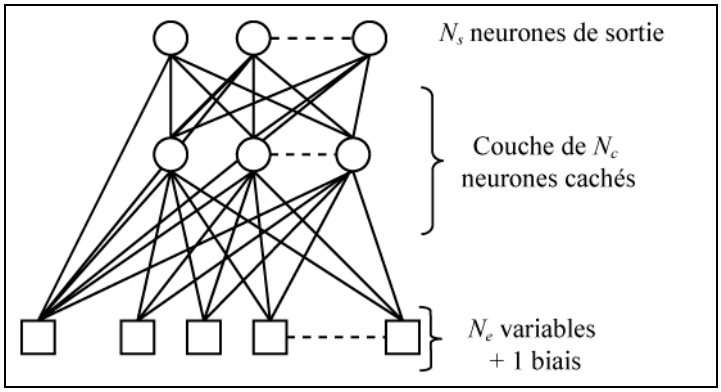


Figure 2 : Représentation d'un réseau de neurones

À chaque connexion est associé un paramètre. Les sorties du réseau sont donc des fonctions non-linéaires de ses variables et de ses paramètres. Le nombre de degrés de liberté, c'est-à-dire de paramètres ajustables, dépend du nombre de neurones de la couche cachée ; il est donc possible de faire varier la complexité du réseau en augmentant ou en diminuant le nombre de neurones cachés.

II.2.1 - Propriétés des réseaux de neurones

Les réseaux de neurones ont pour but de modéliser des processus, à partir d'exemples de couples entrées / sorties. Ils ont la propriété d'***approximation universelle*** : un réseau de neurones comportant un nombre fini de neurones cachés, de même fonction d'activation, et un neurone de sortie linéaire, est capable d'approcher uniformément, avec une précision arbitraire, toute fonction bornée suffisamment régulière, sur un domaine fini de l'espace de ses variables. De plus, il s'agit d'***approximateurs parcimonieux*** : une approximation par un réseau de neurones nécessite en général moins de paramètres que les approximateurs usuels. Le nombre de paramètres nécessaires pour obtenir une précision donnée augmente en effet linéairement avec le nombre de variables pour un réseau de neurones, alors qu'il croît exponentiellement pour un modèle linéaire par rapport aux paramètres. Cette propriété est très importante, car les réseaux de neurones demandent de ce fait moins d'exemples que d'autres approximateurs pour l'apprentissage.

II.2.2 - Apprentissage des réseaux de neurones

Considérons un ensemble d'apprentissage, constitué de *N* couples entrées / sorties, c'est-à-dire d'un ensemble de variables $\left\{\mathbf{x}^i, i=1...N\right\}$ et d'un ensemble de mesures de la grandeur à modéliser $\left\{y\left(\mathbf{x}^i\right), i=1...N\right\}$. Pour une complexité donnée (le choix de cette complexité est étudié dans la section II.3 de ce chapitre), l'apprentissage s'effectue par minimisation de la fonction de coût des moindres carrés, définie par :

$$J\left(\boldsymbol{\theta}\right)=\frac{1}{2}\sum_{i=1}^{N}\left[y\left(\mathbf{x}^i\right)-g\left(\mathbf{x}^i,\boldsymbol{\theta}\right)\right]^2 \tag{4}$$

La minimisation de cette fonction s'effectue par une descente de gradient. Cet algorithme a pour but de converger, de manière itérative, vers un minimum de la fonction de coût, à partir de valeurs initiales des poids aléatoires. À chaque étape, le gradient de la fonction est calculé, à l'aide de l'algorithme de ***rétropropagation***. Puis les paramètres sont modifiés en fonction de ce gradient, dans la direction de la plus forte pente, vers un minimum local de *J*. Cette descente peut être effectuée suivant plusieurs méthodes : gradient simple ou méthodes du second ordre, dérivées de la méthode de Newton. Les méthodes du second ordre, généralement plus efficaces, sont les plus utilisées. La procédure de minimisation est arrêtée lorsqu'un critère est satisfait : le nombre maximal d'itérations est atteint, la variation du module du vecteur des paramètres ou du gradient de la fonction de coût est trop faible…

II.2.3 - Des réseaux de neurones particuliers : réseaux de fonctions radiales de base

Les réseaux de neurones de fonctions radiales de base (souvent notés RBF), sont des réseaux dont la couche cachée est composée de neurones à fonction d'activation gaussienne radiale. La fonction d'activation d'un neurone *j* est par exemple :

$$f\left(\mathbf{x},\sigma_j,\boldsymbol{\theta}_j\right)=\exp\left(-\frac{\left\|\mathbf{x}-\boldsymbol{\theta}_j\right\|^2}{2\sigma_j^2}\right)$$

où σ_j est l'écart-type de la gaussienne et $\boldsymbol{\theta}_j$ est le vecteur des coordonnées de son centre. Les neurones de sortie sont à fonction d'activation linéaire, et sont reliés aux neurones de la couche cachés par des poids β_j ajustables. Une sortie *y* d'un réseau à N_c neurones cachés est ainsi déterminée par :

$$y=\sum_{j=1}^{N_c}\beta_j f\left(\mathbf{x},\sigma_j,\boldsymbol{\theta}_j\right)$$

La sortie est une fonction linéaire des poids β_j, et non-linéaire des paramètres des gaussiennes. Si les paramètres des gaussiennes sont fixés, la sortie devient une fonction linéaire des poids β_j, et les réseaux perdent leur propriété de parcimonie. De plus, la réussite de l'apprentissage

dépend fortement de l'initialisation. Enfin, la configuration d'un réseau de neurones RBF optimal est difficile. Lors du processus d'apprentissage du réseau, deux stratégies sont possibles. La première consiste à modifier simultanément tous les paramètres du réseau (les coordonnées des centres des fonctions radiales, leur écart-type et les poids β_j), par descente de gradient. Cependant, les dynamiques de convergence des fonctions radiales et des poids β_j sont différentes, et les poids convergent plus rapidement que les autres paramètres. L'apprentissage conduit très souvent à un minimum local. Une autre méthode consiste à optimiser séparément les paramètres de la couche cachée, par apprentissage non-supervisé, et les poids entre la couche cachée et la couche de sortie, par descente de gradient.

II.3 - **Sélection du modèle**

La modélisation vise à fournir un modèle qui soit non seulement ajusté aux données d'apprentissage, mais aussi capable de prédire la valeur de la sortie sur de nouveaux exemples, c'est-à-dire de généraliser. Soit R_i l'erreur commise par le modèle considéré sur l'exemple i (également appelé résidu) :

$$R_i = y\left(\mathbf{x}^i\right) - g\left(\mathbf{x}^i, \boldsymbol{\theta}\right)$$

où $y\left(\mathbf{x}^i\right)$ est la valeur mesurée de la grandeur à modéliser pour l'exemple i, et $g\left(\mathbf{x}^i, \boldsymbol{\theta}\right)$ est l'estimation du modèle pour ce même exemple. L'erreur du modèle sur les données d'apprentissage, E_A, peut être évaluée par l'erreur quadratique moyenne en apprentissage, appelée également EQMA :

$$E_A = \sqrt{\frac{1}{N_A}\sum_{i=1}^{N_A}\left(R_i\right)^2} \tag{5}$$

où N_A est le nombre d'exemples servant à établir le modèle. La qualité du modèle en apprentissage est souvent visualisée sur un *diagramme de dispersion*, sur lequel sont portées les valeurs de la grandeur d'intérêt estimées par le modèle, en fonctions des valeurs mesurées de cette grandeur. La qualité de la modélisation est d'autant meilleure que les points de ce graphique sont proches de la première bissectrice. L'ajustement des points à cette droite peut être évalué par le ***coefficient de détermination*** :

$$R^2 = \frac{\sum_{i=1}^{N_A}\left(y^i - \bar{\mathbf{y}}\right)^2}{\sum_{i=1}^{N_A}\left(g\left(\mathbf{x}^i, \boldsymbol{\theta}\right) - \bar{\mathbf{y}}\right)^2} \tag{6}$$

où $\bar{\mathbf{y}}$ est la moyenne des valeurs mesurées. Ce coefficient est égal au rapport de la variance expliquée à la variance totale de la sortie. Plus il est proche de 1, plus la corrélation entre les valeurs mesurées et prédites est forte. Les valeurs intermédiaires renseignent sur le degré de dépendance linéaire entre les deux variables.

L'erreur sur la base d'apprentissage seule n'est pas un bon indicateur de la qualité du modèle ; un bon modèle doit être capable de rendre compte de la relation déterministe entre les variables et la grandeur modélisée, sans s'ajuster au bruit des données d'apprentissage. Il convient donc de trouver comment estimer cette capacité.

II.3.1 - Méthode générale de sélection

Lors de la modélisation, nous recherchons à la fois la complexité la mieux adaptée au problème étudié, et les paramètres du modèle correspondant. Le modèle doit en effet être de complexité suffisante pour rendre compte de la relation entre les variables explicatives et la grandeur modélisée, mais il ne doit pas être trop complexe, afin de ne pas être trop sensible au bruit présent dans les données. En effet, lorsque la complexité du modèle envisagé augmente, on observe généralement une diminution du coût d'apprentissage, car le modèle s'ajuste de plus en plus précisément aux données d'apprentissage. En revanche, l'erreur de généralisation diminue pour atteindre un minimum, puis augmente avec la complexité du modèle, car celui-ci est alors surajusté aux données d'apprentissage, et au bruit présent dans ces données. Ce compromis est également connu sous le nom de dilemme biais-variance : le biais caractérise l'écart entre les estimations et les mesures, et la variance reflète l'influence du choix de la base d'apprentissage sur le modèle [17]. De plus, lorsque les modèles envisagés ne sont pas linéaires par rapport à leurs paramètres, la sélection du modèle optimal ne consiste pas uniquement à choisir la meilleure famille de fonctions, comme c'est le cas pour des modèles linéaires (modèles polynomiaux ou combinaisons linéaires de fonctions non paramétrées). La sélection s'effectue donc souvent en plusieurs étapes.

- Pour les modèles non linéaires en leurs paramètres, la première sélection s'effectue au sein d'une famille de fonctions donnée. En effet, la fonction de coût n'étant pas quadratique en les paramètres du modèle, elle ne possède pas un minimum unique. L'optimisation de cette fonction peut donc conduire à des minima différents, donc à différents modèles de même complexité. Il est donc nécessaire de réaliser plusieurs apprentissages, pour une complexité donnée, et de choisir celui qui est susceptible de posséder les meilleures propriétés de généralisation.
- Il faut ensuite sélectionner, parmi les meilleurs modèles de complexités différentes, celui qui présente les meilleures capacités de généralisation.

Nous allons maintenant détailler les critères de sélection les plus fréquemment utilisés, en étudiant dans un premier temps comment l'erreur de généralisation peut être évaluée.

II.3.2 - Sélection du modèle par estimation de l'erreur de généralisation

L'erreur de généralisation ne peut pas être évaluée exactement ; la base de données disponible est en effet de taille limitée, et la distribution de probabilité des données, dont dépend cette erreur, est généralement inconnue. Il est donc nécessaire de trouver une approximation de cette erreur de généralisation. Nous allons ainsi présenter les méthodes les plus utilisées pour effectuer ces sélections.

La méthode la plus commune, dite méthode de validation simple ou hold-out, consiste à construire, à partir de l'ensemble des données, deux ensembles : les paramètres du modèle sont ajustés sur la ***base d'apprentissage***, et l'erreur de généralisation est évaluée sur la ***base de validation***.

L'erreur en validation est définie de façon analogue au coût d'apprentissage :

$$E_V = \sqrt{\frac{1}{N_V}\sum_{i=1}^{N_V}\left(R_i\right)^2} \tag{7}$$

où N_V est le nombre d'exemples dans la base de validation.

Pour des modèles de complexité donnée, la méthode de sélection consiste à effectuer plusieurs apprentissages à partir de différentes initialisations des paramètres, lorsque l'on a affaire à un modèle non linéaire en ses paramètres. Il faut alors choisir, parmi les modèles obtenus, celui qui fournit la plus faible erreur en validation. Il s'agit d'une méthode très rapide, mais d'efficacité limitée, surtout lorsque le nombre d'exemples disponible est faible. En effet, si le nombre d'exemples est petit, la variance de l'estimation de l'erreur de généralisation est grande ; le coût de validation dépend alors fortement du choix de la base de validation [18]. De plus, les exemples de la base de validation ne sont pas utilisés pour l'estimation des paramètres du modèle, et peuvent faire défaut : le modèle n'étant pas ajusté à certains types d'exemples, ses capacités de généralisation sont limitées. Par conséquent, l'erreur de généralisation est plus souvent estimée à l'aide d'autres méthodes, telles que la validation croisée ou le leave-one-out.

II.3.2.1 - La validation croisée

L'ensemble des N exemples disponibles est cette fois-ci divisé en K sous-ensembles disjoints. La valeur $K = 10$ est souvent retenue, mais il n'y a pas de méthode de choix optimal de K. On construit alors une base d'apprentissage à partir de $K-1$ de ces sous-ensembles. Le modèle est ajusté sur cette base, puis l'erreur de prédiction R_i est calculée pour chaque exemple du sous-ensemble restant. Cette étape est réalisée K fois, en choisissant un sous-ensemble de validation différent à chaque itération. Ainsi, chaque exemple se trouve une fois et une seule dans une base de validation. Cette procédure est illustrée sur la Figure 3, où $K = 5$. Les sous-ensembles $E_1 \ldots E_5$ constituent successivement la base de validation.

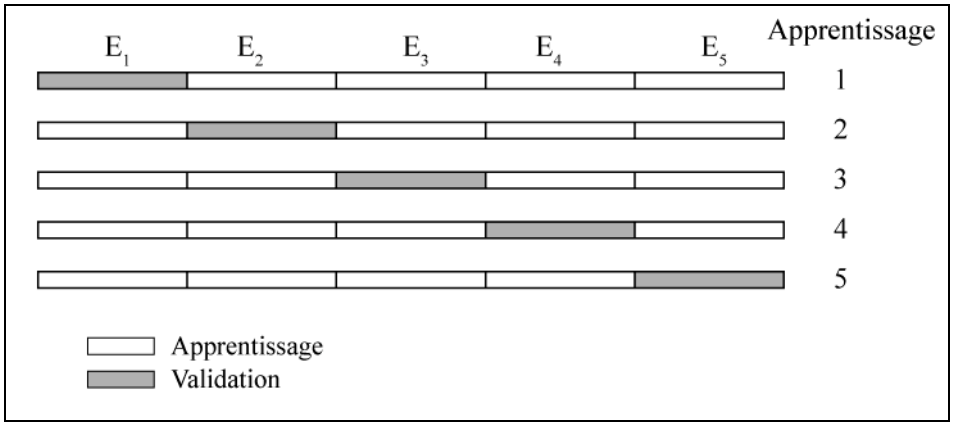


Figure 3 : Principe de la validation croisée

La performance en généralisation de la famille de modèles est ensuite estimée en calculant le score de validation croisée :

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(R_i\right)^2} \tag{8}$$

Cette technique a l'avantage d'être moins sensible au choix des bases de validation (la variance de l'erreur est plus faible), et permet d'utiliser toutes les données en apprentissage. Elle nécessite cependant d'effectuer K apprentissages, qui produisent ainsi K modèles différents. Une fois la meilleure complexité déterminée, on effectue un apprentissage avec l'ensemble des données disponibles.

II.3.2.2 - Méthode du leave-one-out

Le cas particulier où $K = N$ (N est le nombre d'exemples disponibles) est appelé ***leave-one-out*** : à chaque itération, un exemple i est extrait de l'ensemble d'apprentissage. Une série d'apprentissage est réalisée à l'aide des $N-1$ exemples restants, et l'erreur de prédiction sur l'exemple i est calculée pour chacun des modèles obtenus. On retient l'erreur la plus faible, notée R_i^{-i}. Le score de leave-one-out est alors défini par :

$$E_t = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(R_i^{-i}\right)^2} \tag{9}$$

Ce score E_t est un estimateur non-biaisé de l'erreur de généralisation [19]. Cette méthode présente ainsi l'avantage de tirer pleinement profit des données disponibles, et d'être indépendante du choix de bases de validation. Cependant, le temps de calcul peut devenir très grand, car il est nécessaire de procéder à N séries d'apprentissage.

II.3.3 - Le leave-one-out virtuel

La méthode du leave-one-out virtuel [20] consiste à estimer les erreurs R_i^{-i} à partir d'un seul apprentissage réalisé sur l'ensemble des N exemples, ce qui permet d'estimer le score de leave-one-out sans avoir à réaliser N apprentissages. Cette estimation est appelée ***score de leave-one-out virtuel***, et repose sur le calcul de la matrice jacobienne du modèle $g\left(\mathbf{x},\boldsymbol{\theta}^*\right)$, obtenu à partir des N exemples. Un développement de ce modèle au premier ordre par rapport aux paramètres, au voisinage de $\boldsymbol{\theta}^*$, est :

$$g\left(\mathbf{x},\boldsymbol{\theta}\right) \cong g\left(\mathbf{x},\boldsymbol{\theta}^*\right) + \mathbf{Z}\left(\boldsymbol{\theta}-\boldsymbol{\theta}^*\right)$$

Ce développement fait apparaître $\mathbf{Z}$, matrice jacobienne du modèle. Cette matrice est de taille (N, q), où q est le nombre de paramètres du modèle. Les N éléments de la colonne j sont égaux aux dérivées partielles de la sortie par rapport au paramètre j. Les éléments de $\mathbf{Z}$ s'écrivent donc :

$$\left(\mathbf{Z}\right)_{ij} = \left(\frac{\partial g\left(x_i, \boldsymbol{\theta}_{mc}\right)}{\partial \theta_j} \right)_{\boldsymbol{\theta}=\boldsymbol{\theta}_{mc}} \tag{10}$$

Les modèles dont la jacobienne n'est pas de rang plein, c'est-à-dire de rang inférieur à q, doivent être rejetés. En effet, cela signifie qu'au moins deux paramètres ont des effets qui ne sont pas indépendants sur la sortie. Ces modèles comportent donc trop de paramètres, et ont de grandes chances d'être surajustés.

Considérons la matrice de projection orthogonale sur le sous-espace défini par les colonnes de la matrice $\mathbf{Z}$: $\mathbf{H} = \mathbf{Z}\left(\mathbf{Z}^T\mathbf{Z}\right)^{-1}\mathbf{Z}^T$, qui est une matrice carrée (N, N).

Les termes diagonaux de cette matrice sont les ***leviers de plan tangent*** (appelés simplement leviers dans la suite du mémoire) des exemples. Le levier d'un exemple i est ainsi égal à :

$$h_{ii} = \mathbf{z}^{iT}(\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{z}^i \tag{11}$$

où $\mathbf{z}^i$ est le vecteur dont les composantes sont celles de la i-ième ligne de **Z.**

Le calcul de ces leviers permet d'estimer l'erreur de prédiction sur un exemple i lorsqu'il est retiré de la base d'apprentissage (c'est-à-dire R_i^{-i}), à partir de l'erreur commise sur cet exemple lorsqu'il est dans cette base (c'est-à-dire R_i) :

$$R_i^{-i} \cong \frac{R_i}{1-h_{ii}} \tag{12}$$

Ce résultat est d'ailleurs exact si le modèle est linéaire ; il est alors appelé PRESS (Predicted REsidual Sum of Squares).

Dès lors, il est possible de calculer le score de leave-one-out virtuel, qui constitue une très bonne approximation de l'erreur de généralisation si le développement limité au premier ordre sur lequel elle est fondée est suffisamment précis :

$$E_p = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{R_i}{1-h_{ii}}\right)^2} \tag{13}$$

Ce calcul permet non seulement d'estimer rapidement l'erreur de généralisation, mais également de révéler d'éventuels surajustements du modèle à des exemples particuliers. Il peut ainsi constituer la base d'une méthode de planification expérimentale : s'il est possible d'enrichir la base de données de mesures supplémentaires, celles-ci doivent être effectuées au voisinage des points qui ont des leviers importants.

II.3.4 - Sélection de modèle à l'aide des leviers

II.3.4.1 - Interprétation des leviers

Nous avons vu que les leviers sont les termes diagonaux de la matrice de projection orthogonale sur le sous-espace des colonnes de la matrice jacobienne. Lorsque la matrice **Z** est de rang plein, les leviers vérifient les propriétés suivantes :

$$\begin{cases} \sum_{i=1}^{N} h_{ii} = q \\ 0 < h_{ii} < 1 \quad \forall i \in [\![1, N]\!] \end{cases}$$

où q est le nombre de paramètres du modèle. Le levier h_{ii} relatif à un exemple i peut ainsi être interprété comme la proportion des paramètres utilisée par le modèle pour s'ajuster à l'exemple i. Par conséquent, si tous les exemples ont la même influence sur le modèle, les leviers sont tous égaux à $\frac{q}{N}$. Si le levier d'un exemple est proche de 1, le modèle est particulièrement ajusté sur cet exemple, et il est presque parfaitement appris. Par contre, l'erreur en prédiction sur cet exemple lorsqu'il n'appartient pas à la base d'apprentissage, qui peut être estimée par $R_i^{-i} \cong \frac{R_i}{1-h_{ii}}$, est très grande. En revanche, un exemple dont le levier est proche de 0 n'a aucune influence sur le modèle. Si un modèle consacre une grande partie de ses degrés de liberté à un ou plusieurs exemples, il doit être rejeté, car il est susceptible d'être ajusté au bruit présent dans ces mesures [21]. Les leviers constituent donc un outil supplémentaire de sélection.

II.3.4.2 - Calcul des intervalles de confiance

Les leviers permettent également le calcul des intervalles de confiance sur les prédictions du modèle. Un intervalle de confiance au seuil de confiance $(1-\alpha)$ est un intervalle tel qu'on peut dire qu'il contient la vraie valeur de la grandeur prédite avec une probabilité $(1-\alpha)$. L'étendue de l'intervalle de confiance caractérise la confiance que l'on peut avoir dans la prédiction faite par le modèle : plus l'étendue est faible, plus cette confiance est grande. Lorsque le modèle est non-linéaire, un intervalle de confiance approché pour l'espérance mathématique de la sortie Y_p peut être calculé à l'aide des leviers. Pour un exemple i de la base d'apprentissage, son expression est :

$$E\left(Y_P \mid \mathbf{x}^i\right) \in g\left(\mathbf{x}^i, \boldsymbol{\theta}\right) \pm t_\alpha^{N-q} s \sqrt{\mathbf{z}^{iT}\left(\mathbf{Z}^T\mathbf{Z}\right)^{-1}\mathbf{z}^i} = g\left(\mathbf{x}^i, \boldsymbol{\theta}\right) \pm t_\alpha^{N-q} s \sqrt{h_{ii}} \qquad (14)$$

où $g\left(\mathbf{x}^i, \boldsymbol{\theta}\right)$ est la prédiction du modèle pour cet exemple, t_α^{N-q} est la valeur d'une variable de Student à N-q degrés de liberté et un niveau de confiance $(1-\alpha)$, et s une estimation de la variance de l'erreur de prédiction de modèle.

L'intervalle de confiance peut également être estimé pour un exemple qui n'appartient pas à la base d'apprentissage :

$$E^{(-i)}\left(Y_P \mid \mathbf{x}^i\right) \in g\left(\mathbf{x}^i, \boldsymbol{\theta}\right) \pm t_\alpha^{N-q-1} s^{-i} \sqrt{\frac{h_{ii}}{1-h_{ii}}} \qquad (15)$$

Cet intervalle est donc d'autant plus large que h_{ii} est proche de 1 : dans ce cas, la prédiction sur cet exemple est très peu fiable.
Les leviers se révèlent donc être un outil efficace pour détecter le surajustement de modèles à des exemples particuliers, et pour repérer ces exemples. Ils permettent la sélection d'un modèle parmi des modèles de complexités différentes, lorsque leurs performances en généralisation sont comparables : le meilleur est généralement celui pour lequel la distribution des leviers des exemples autour de $\frac{q}{N}$ est la plus étroite.
La sélection de modèle peut donc se dérouler de la façon suivante :

- Pour des modèles de complexité donnée, on réalise plusieurs apprentissages, à partir d'initialisations différentes.
- Les modèles dont la matrice jacobienne n'est pas de rang plein sont écartés ; on calcule alors les scores d'apprentissage et de leave-one-out virtuel des modèles obtenus. Le modèle présentant les meilleures capacités de généralisation est retenu.
- Cette sélection est effectuée pour des modèles de complexité croissante. On compare alors les modèles retenus pour chaque complexité, c'est-à-dire leurs scores d'apprentissage et de leave-one-out virtuel, ainsi que la répartition des leviers des exemples. Ces critères permettent de sélectionner le meilleur modèle.

Les performances de ce modèle peuvent alors être mesurées sur une base de test : elle est composée d'exemples n'ayant pas servi à établir ni à choisir le modèle. Le score de test permet ainsi d'évaluer les performances du modèle en généralisation.

III - Optimisation sous contraintes

Nous introduisons maintenant les méthodes d'optimisation sous contraintes, et montrons comment les méthodes à noyaux, telles que les machines à vecteurs supports, peuvent être utilisées pour modéliser des propriétés ou des activités moléculaires. Le développement de la théorie statistique de l'apprentissage, initié en 1979 par V. Vapnik [22], a permis l'élaboration des méthodes à noyaux, nouvelle classe d'algorithmes d'apprentissage. Ces algorithmes abordent sous un angle différent le dilemme biais-variance, car ils intègrent à l'estimation de la fonction le contrôle de sa complexité. Leur mise en œuvre diffère de celle des méthodes abordées dans la section II. Elle demande en effet la résolution de problèmes d'optimisation s'exprimant sous forme d'une fonction de coût à minimiser, soumise à des contraintes : il s'agit de minimiser l'erreur de modélisation sur la base d'apprentissage tout en garantissant de bonnes capacités de généralisation. Les méthodes à noyaux ont été rapidement adoptées pour leur capacité à traiter des données de grandes dimensions, le fait qu'elles soient bien fondées théoriquement, et leurs bons résultats en pratique.
Nous rappellerons tout d'abord les principaux concepts de la théorie statistique de l'apprentissage, sur laquelle sont fondées ces méthodes, et introduirons le principe de minimisation du risque structurel. Nous présenterons ensuite les machines à vecteurs

supports, et décrirons leur apprentissage en classification ainsi qu'en régression. Nous décrirons enfin comment ces méthodes sont utilisées pour la modélisation de propriétés et d'activités de molécules.

III.1 - Théorie statistique de l'apprentissage

III.1.1 - Espaces de Hilbert à noyau reproduisant

Considérons :

- H, un espace de Hilbert de fonctions ;
- X le domaine de définition des fonctions ;
- $\langle f,g \rangle$ le produit scalaire de f et g dans H.

Une fonction K est dite ***noyau reproduisant*** de l'espace hilbertien H si et seulement si :

- $\forall \mathbf{x} \in X,\ K_{\mathbf{x}} = K(.,\mathbf{x}) \in H$
- $\forall\ (f,\mathbf{x}) \in H \times X, f(\mathbf{x}) = \langle f, K_{\mathbf{x}} \rangle_H$

Si une telle fonction K existe, elle est unique. H est alors un ***espace de Hilbert à noyau reproduisant*** (noté RKHS, pour Reproducing Kernel Hilbert Space).

Propriétés :

- Tout noyau reproduisant est un ***noyau défini positif*** :

$$\forall n > 0,\ \forall (\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_n) \in X^n,\ \forall (c_1,c_2,...,c_n) \in \mathbb{R}^n, \sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j K(\mathbf{x}_i,\mathbf{x}_j) \geq 0$$

- Un noyau reproduisant vérifie la ***propriété de reproduction*** :

$$\langle K_{\mathbf{x}}, K_{\mathbf{y}} \rangle = K(\mathbf{x},\mathbf{y})$$

- Soit $K_{\mathbf{x}}$ un noyau semi-défini positif. Il existe une fonction $\phi(\mathbf{x})$ à valeurs dans un espace F muni du produit scalaire, telle que :

$$K(\mathbf{x},\mathbf{x'}) = \langle \phi(\mathbf{x}), \phi(\mathbf{x'}) \rangle_F$$

Cette fonction réalise une transformation de l'espace des données dans un espace des caractéristiques, aussi appelé *"feature space"*. Toute fonction positive définie peut en effet être exprimée comme un produit interne dans un espace de description (théorème de Mercer). Le noyau $K(\mathbf{x},\mathbf{x'})$ exprime donc une similitude entre les images de **x** et de **x'** par la transformation ϕ.

III.1.2 – Risque moyen et risque empirique

Soient deux variables aléatoires $\mathbf{x} \in X$ et $y \in Y$, où X et Y sont deux ensembles munis d'une loi de probabilité conjointe $P(X,Y)$. On suppose que l'on dispose d'un ensemble de N observations, réalisations de ces variables :

$$D_N = \{(\mathbf{x}_i, y_i) \in X \times Y\}_{i=1,...N}$$

Le problème de l'apprentissage consiste à trouver un bon estimateur, c'est-à-dire une fonction $f : X \to Y$, choisie dans une classe de fonctions $\mathcal{F}$, qui réalise une bonne approximation $f(\mathbf{x})$ de la réponse y, pour tout $\mathbf{x}$, et notamment pour les valeurs qui ne figurent pas dans la base d'apprentissage. Il s'agit donc de minimiser l'erreur commise en approchant y par $f(\mathbf{x})$, erreur qui peut être estimée par une fonction de perte, notée $V\left(y, f\left(\mathbf{x}\right)\right)$. On souhaite ainsi minimiser le ***risque moyen***, espérance de cette erreur, défini par :

$$I\left[f\right] = \int_{X,Y} V\left(y, f\left(\mathbf{x}\right)\right) P\left(\mathbf{x}, y\right) d\mathbf{x} dy$$

La solution f_0 est donnée par : $f_0 = \arg\min_{\mathcal{F}} I\left[f\right]$

Étant donné que la distribution de probabilité $P\left(\mathbf{x}, y\right)$ est inconnue, f ne peut pas être trouvée à partir de cette équation. Il est néanmoins possible de réaliser une approximation du risque moyen, à l'aide par exemple du ***risque empirique***, calculé sur les données D_N :

$$I_{emp}\left[f, N\right] = \frac{1}{N} \sum_{i=1}^{N} V\left(y_i, f\left(\mathbf{x}_i\right)\right)$$

Cependant, ce risque ne tient pas compte de la distribution de probabilité sous-jacente, ce qui pose le problème du surajustement. Il est en effet possible de choisir une fonction f telle que le risque empirique soit nul mais que l'erreur de généralisation soit élevée. De plus, la minimisation de ce risque n'admet pas une solution unique.

III.1.3 - Minimisation du risque structurel

Dans leur théorie de l'apprentissage statistique, Vapnik et Chervonenkis ont prouvé qu'il est possible de majorer l'écart entre le risque empirique mesuré et le risque réel visé en fonction de la famille de fonctions utilisée pour le modèle. On définit pour cela la dimension de Vapnik-Chervonenkis (VC-dimension) [23], qui représente le nombre maximal de points pouvant être séparés de toutes les façons possibles par une fonction de cet ensemble. Notons h la VC-dimension d'un ensemble de fonctions $\left\{V\left(y, f\left(\mathbf{x}\right)\right), f \in \mathcal{F}\right\}$; l'écart entre le risque empirique mesuré et le risque réel vérifie, avec une probabilité 1-η, la relation suivante :

$$I_{emp}\left[f, N\right] - \left(B - A\right)\sqrt{\frac{h \ln\left(\frac{2eN}{h}\right) - \ln\left(\frac{\eta}{4}\right)}{N}} \le I\left[f\right] \le I_{emp}\left[f, N\right] + \left(B - A\right)\sqrt{\frac{h \ln\left(\frac{2eN}{h}\right) - \ln\left(\frac{\eta}{4}\right)}{N}} \quad (16)$$

où A et B sont des bornes de la fonction V sur $\mathcal{F}$.

Ainsi, la seule minimisation du risque empirique ne garantit pas une faible valeur du risque moyen. Il faut également minimiser le second terme des inégalités, fonction croissante de $\frac{h}{N}$.

On introduit alors la notion de ***minimisation du risque structurel*** : il s'agit de trouver une fonction dont le risque empirique s'approche du risque moyen, tout en minimisant ce risque empirique. On cherche donc à minimiser conjointement les deux sources d'erreurs : le risque empirique et l'intervalle de confiance, c'est-à-dire la différence entre le risque empirique et le risque moyen.

Définissons une séquence d'espaces hypothèses imbriqués, de la forme : $H_n = \left\{ f \in RKHS : \|f\|_H \leq A_n \right\}$, avec $A_1 \leq A_2 \leq \ldots \leq A_m$, où $\|f\|_H$ désigne la norme de f sur l'espace de Hilbert. On montre que les VC-dimensions h_n de ces espaces sont des fonctions croissantes de A_n. On cherche alors la fonction f qui minimise le risque empirique sur l'espace pour lequel l'ensemble du terme de droite de l'expression (16) est minimale.

Le problème d'apprentissage consiste alors à rechercher :

$$\min_f \left[\sum_{i=1}^{N} V\left(y_i, f\left(x_i\right)\right) + \lambda \|f\|_H \right]$$

où λ est un paramètre de régularisation, à valeur réelle positive. Le premier terme de la fonction à minimiser pénalise le risque empirique, le second permet d'assurer de bonnes capacités de généralisation.

III.2 - Les machines à vecteurs supports

Les machines à vecteurs supports (ou SVM, pour Support Vector Machines) ont été développées dans les années 90 [24] sur la base de la théorie de l'apprentissage statistique que nous venons d'aborder. À l'origine conçues pour les tâches de classification ou de reconnaissance de formes, elles permettent également de traiter les problèmes de régression non linéaire. Le principe théorique des SVM comporte deux points fondamentaux : la transformation non linéaire de l'espace de représentation des données d'entrées en un espace de plus grande dimension muni d'un produit scalaire (espace de Hilbert), et la détermination d'un hyperplan permettant une séparation linéaire optimale dans cet espace.

III.2.1 - Problèmes de classification

Les principes précédemment abordés peuvent être appliqués lors de la recherche d'un hyperplan séparateur optimal, capable de séparer les données et de maximiser la distance entre ces deux classes, dans une tâche de classification.

III.2.1.1 - Cas linéairement séparable

Soit un ensemble d'apprentissage composé de N exemples $S = \left\{\left(\mathbf{x}_i, y_i\right) \in X \times Y\right\}_{i=1\ldots N}$, linéairement séparables, tel que $Y = \left\{-1;1\right\}$. Un hyperplan séparateur est un hyperplan qui vérifie :

$$\begin{cases} \boldsymbol{\theta} \cdot \mathbf{x}_i + b \geq +1 \text{ si } y_i = +1 \\ \boldsymbol{\theta} \cdot \mathbf{x}_i + b \leq -1 \text{ si } y_i = -1 \end{cases}$$

avec $\boldsymbol{\theta} \in X$, $b \in \mathbb{R}$, conditions qui peuvent être reformulées ainsi :

$$y_i \cdot (\boldsymbol{\theta} \cdot \mathbf{x}_i + b) \geq 1 \quad \forall i \in \left\{1,...,N\right\} \tag{17}$$

L'hyperplan optimal, qui est unique, est celui qui se situe à la distance maximale des vecteurs $\mathbf{x}$ les plus proches. On dit que cet hyperplan maximise la marge. La distance d'un point $\mathbf{x}_i$ de l'espace à l'hyperplan d'équation $\boldsymbol{\theta} \cdot \mathbf{x} + b = 0$ est donnée par :

$$\delta_i = \frac{\left|\boldsymbol{\theta} \cdot \mathbf{x}_i + b\right|}{\left\|\boldsymbol{\theta}\right\|^2}$$

Il a été montré que maximiser la marge réduit la VC-dimension de la famille de fonctions.

Ainsi l'hyperplan à marge maximale recherché maximise $\frac{1}{\left\|\boldsymbol{\theta}\right\|^2}$ ou minimise $\left\|\boldsymbol{\theta}\right\|^2$ sous les contraintes (17).

Pour trouver le meilleur classifieur, en termes de minimisation du risque structurel, il faut donc résoudre le problème d'optimisation suivant :

$$\begin{cases} \text{minimiser } \frac{1}{2}\left\|\boldsymbol{\theta}\right\|^2 \\ \text{sous } y_i \cdot \left(\boldsymbol{\theta}.\mathbf{x}_i + b\right) - 1 \geq 0, \ \forall i \in [\![1, N]\!] \end{cases} \tag{18}$$

Il s'agit d'un problème d'optimisation avec contraintes, qui peut être résolu par la méthode de Lagrange. On exprime le lagrangien L_p comme la somme de la fonction à minimiser et de l'opposé de chaque contrainte i multipliée par une constante α_i, appelée "multiplicateur de Lagrange" :

$$L_p = \frac{1}{2}\left\|\boldsymbol{\theta}\right\|^2 - \sum_{i=1}^{N} \alpha_i \left[y_i \left(\boldsymbol{\theta} \cdot \mathbf{x}_i + b\right) - 1 \right]$$

Formulation duale :

L'existence d'un point selle impose que le minimum par rapport à $\boldsymbol{\theta}$ et b et le maximum par rapport aux variables α_i coïncident. En annulant les dérivées partielles du lagrangien, selon les conditions de Karush-Kuhn-Tucker (KKT) [25], le problème devient alors le suivant :

$$\begin{cases} \max\limits_{\alpha} \sum\limits_{i=1}^{N} \alpha_i - \dfrac{1}{2} \sum\limits_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j \right) \\ \alpha_i \geq 0, \ \forall i \in [\![1, N]\!] \\ \sum\limits_{i=1}^{N} \alpha_i y_i = 0 \end{cases} \tag{19}$$

L'hyperplan solution s'écrit alors :

$$h(\mathbf{x}) = \boldsymbol{\theta}^* \cdot \mathbf{x} + b = \sum_{i=1}^{N} \alpha_i^* y_i \left(\mathbf{x}_i \cdot \mathbf{x} \right) + b^*$$

Une condition complémentaire de KKT indique que pour tout exemple *i* :

$$\alpha_i^* \left[y_i \cdot \left(\boldsymbol{\theta}^* . \mathbf{x}_i + b^* \right) - 1 \right] = 0$$

c'est-à-dire $\alpha_i^* = 0$ ou $y_i \cdot \left(\boldsymbol{\theta}^* . \mathbf{x}_i + b^* \right) - 1 = 0$.

Ce résultat nous conduit à définir les ***vecteurs supports***, qui sont les points pour lesquels les contraintes sont saturées, situés exactement sur la marge. Ce sont les seuls points pour lesquels les coefficients α_i peuvent être non nuls. Ainsi, seuls les vecteurs supports participent à la définition de l'hyperplan optimal ; cette représentation est donc parcimonieuse lorsque le nombre de ces vecteurs est faible.

III.2.1.2 - Cas non-linéairement séparable

Considérons maintenant le cas d'un problème où les données ne sont pas linéairement séparables. Ce cas peut être traité par l'utilisation de deux nouvelles techniques, qui consistent à introduire une marge souple, et à projeter les données dans un espace de dimension plus grande, dans lequel elles seraient linéairement séparables.

Le principe de marge souple consiste à tolérer des erreurs de classification. On introduit des variables ξ_i, qui permettent de pénaliser les erreurs de classification réalisées sur l'ensemble d'apprentissage.

Le problème d'optimisation (18) devient alors :

$$\begin{cases} \text{minimiser } \phi(\boldsymbol{\theta}, \boldsymbol{\xi}) = \dfrac{1}{2} \|\boldsymbol{\theta}\|^2 + C \sum\limits_{i=1}^{N} \xi_i \\ \text{sous } \xi_i \geq 0, \ y_i \cdot \left(\boldsymbol{\theta} . \mathbf{x}_i + b \right) \geq 1 - \xi_i \quad \forall i \in [\![1, N]\!] \end{cases} \tag{20}$$

La constante *C* est un paramètre de l'algorithme, et définit la tolérance aux erreurs. Sa valeur optimale peut être obtenue par validation croisée. Si la marge est supérieure à 1 ($\xi_i = 0$), le coût supplémentaire est nul ; dans le cas contraire ce coût vaut $C\xi_i$. Le problème peut alors se réécrire sous la même forme que le problème (19), avec une contrainte supplémentaire sur les variables α_i :

$$0 \leq \alpha_i \leq C, \ \forall i \in [\![1, N]\!]$$

D'autre part, nous pouvons remarquer que la détermination de l'hyperplan optimal ne fait intervenir que les produits scalaires entre vecteurs. Il est ainsi possible de projeter les vecteurs des variables dans un espace de plus grande dimension, appelé espace des caractéristiques, dans lequel les données seraient linéairement séparables. L'hyperplan optimal peut alors être construit dans cet espace.
Soit ϕ une transformation de l'espace des données dans un espace des caractéristiques *H* muni du produit scalaire $\left\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \right\rangle_H = K(\mathbf{x}_i, \mathbf{x}_j)$, où *K* est une fonction noyau. Les variables sont tout d'abord projetées dans un espace de caractéristiques *H* de dimension supérieure, par la transformation $\phi : X \to H$. Cette fonction est celle associée au noyau *K*, de telle sorte qu'il n'est pas nécessaire de connaître ϕ de manière explicite. On peut ensuite appliquer l'algorithme développé pour des fonctions linéaires dans l'espace *H*, en remplaçant les produits scalaires des vecteurs d'entrée $(\mathbf{x}_i \cdot \mathbf{x}_j)$ par $K(\mathbf{x}_i, \mathbf{x}_j)$.

L'hyperplan optimal est obtenu par résolution du système d'équations :

$$\begin{cases} \max\limits_{\alpha} \sum\limits_{i=1}^{N} \alpha_i - \dfrac{1}{2} \sum\limits_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \alpha_i \geq 0, \ \forall i \in [\![1, N]\!] \\ \sum\limits_{i=1}^{N} \alpha_i y_i = 0 \end{cases}$$

L'équation de l'hyperplan séparateur est alors :

$$h(x) = \boldsymbol{\theta}^* \cdot \varphi(\mathbf{x}) + b = \sum_k y_k \alpha_k^* K(\mathbf{x}_k, \mathbf{x}) + b \text{ avec } \alpha_k^* \geq 0$$

III.2.2 - Les machines à vecteurs supports pour la régression

Nous allons maintenant décrire comment la technique des SVMs peut être utilisée en régression. Considérons un ensemble d'apprentissage $S = \left\{ (\mathbf{x}_i, y_i) \in X \times Y \right\}_{i=1\ldots N}$, où $X \subset \mathbb{R}^d$ et $Y \subset \mathbb{R}$. Nous cherchons à obtenir, à partir de ces données, une fonction *f* capable de traduire la relation entre le vecteur de variables **x** et la grandeur modélisée *y*.

III.2.2.1 - Modèle linéaire

Réalisons tout d'abord un modèle linéaire : l'espace *H* est donc l'ensemble des fonctions linéaires, de la forme :

$$f(x) = \boldsymbol{\theta} \cdot \mathbf{x} + b \ \text{ où } \ \boldsymbol{\theta} \in \mathbb{R}^d, \ b \in \mathbb{R}$$

Cette fonction doit présenter une déviation maximale de ε par rapport aux données d'apprentissage. On définit ainsi la fonction de coût ε-insensible :

$$V\left(y,f\left(\mathbf{x}\right)\right)=\left|y-f\left(\mathbf{x}\right)\right|_{\varepsilon}=\begin{cases}0 \text{ si } \left|\mathbf{x}\right|<\varepsilon\\ \left|\mathbf{x}\right|-\varepsilon \text{ sinon}\end{cases}$$

Par conséquent, seules les déviations supérieures à ε entre les valeurs mesurées et estimées de la sortie *y* sont pénalisantes. Le risque empirique régularisé est alors défini par :

$$F\left[\boldsymbol{\theta},C\right]=C\sum_{i=1}^{N}\left|V\left(y_i,f\left(\mathbf{x}_i,\boldsymbol{\theta}\right)\right)\right|_{\varepsilon}+\frac{1}{2}\left\|f\right\|_{H}^{2}$$

C est un paramètre de régularisation, qui permet de contrôler la complexité du modèle.
Le problème de minimisation de ce risque peut être reformulé comme un problème de minimisation quadratique sous contraintes, qui s'écrit ainsi :

$$\begin{aligned}&\min_{\boldsymbol{\theta}}\frac{1}{2}\left\|\boldsymbol{\theta}\right\|^2+C\sum_{i=1}^{N}\left(\xi_i+\xi_i^*\right)\\ &\text{sous}\begin{cases}y_i-\boldsymbol{\theta}.\mathbf{x}_i-b\le\varepsilon+\xi_i\\ \boldsymbol{\theta}.\mathbf{x}_i+b-y_i\le\varepsilon+\xi_i^*\\ \xi_i,\xi_i^*\ge 0\ \forall i\in\left[\!\left[1,N\right]\!\right]\end{cases}\end{aligned}\qquad(21)$$

En utilisant le lagrangien associé à la fonction objectif, et en introduisant les multiplicateurs de Lagrange α_i et α_i^*, on peut écrire le problème dual équivalent :

$$\begin{aligned}&\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*}\ -\frac{1}{2}\sum_{i,j=1}^{N}\left(\alpha_i-\alpha_i^*\right)\left(\alpha_j-\alpha_j^*\right)\mathbf{x}_i.\mathbf{x}_j-\varepsilon\sum_{i=1}^{N}\left(\alpha_i+\alpha_i^*\right)+\sum_{i=1}^{N}\left(\alpha_i-\alpha_i^*\right)\\ &\text{sous}\begin{cases}\sum_{i,j=1}^{N}\left(\alpha_i-\alpha_i^*\right)=0\\ \alpha_i,\alpha_i^*\in\left[0,C\right]\end{cases}\end{aligned}\qquad(22)$$

La fonction *f* optimale est alors donnée par :

$$f\left(\mathbf{x}\right)=\sum_{i=1}^{N}\left(\alpha_i-\alpha_i^*\right)\mathbf{x}_i\cdot\mathbf{x}+b$$

Les coefficients α_i et α_i^* sont obtenus par résolution du problème d'optimisation (22) et le biais *b* à l'aide des conditions de KKT. Ces conditions permettent également d'établir que seuls les coefficients α_i associés aux exemples pour lesquels les contraintes du problème (21) sont saturées sont non nuls. Les points correspondants sont les ***vecteurs supports***.

III.2.2.2 - Modèle non-linéaire

Il est possible de résoudre le problème de modélisation lorsque la fonction recherchée est non-linéaire, en utilisant une méthode similaire à celle mise en œuvre pour la classification de données non-linéairement séparables.

La fonction est recherchée dans un RHKS H muni d'un noyau K. Les variables d'entrées peuvent être projetées dans cet espace, grâce à une fonction ϕ associée au noyau K. Il devient possible d'utiliser les résultats obtenus dans le cas linéaire. Le problème dual devient ainsi :

$$\begin{aligned} &\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \; -\frac{1}{2}\sum_{i,j=1}^{N}\left(\alpha_i-\alpha_i^*\right)\left(\alpha_j-\alpha_j^*\right)K\left(\mathbf{x}_i,\mathbf{x}_j\right)-\varepsilon\sum_{i=1}^{N}\left(\alpha_i+\alpha_i^*\right)+\sum_{i=1}^{N}\left(\alpha_i-\alpha_i^*\right) \\ &\text{sous} \begin{cases} \sum_{i,j=1}^{N}\left(\alpha_i-\alpha_i^*\right)=0 \\ \alpha_i,\alpha_i^* \in \left[0,C\right] \end{cases} \end{aligned} \tag{23}$$

La solution du problème de régression est :

$$f\left(\mathbf{x}\right)=\sum_{i=1}^{N}\left(\alpha_i-\alpha_i^*\right)K\left(\mathbf{x}_i,\mathbf{x}\right)+b \tag{24}$$

De même qu'en classification, seuls les vecteurs supports participent à la définition de la fonction f ; cette représentation est donc parcimonieuse lorsque le nombre de ces vecteurs est faible.

III.2.3 - Choix de la fonction noyau

Une condition nécessaire et suffisante d'existence d'une fonction noyau est donnée par le théorème de Mercer. Pour être admissible, une fonction noyau doit vérifier les conditions de Mercer, c'est-à-dire être définie positive. En pratique, la fonction noyau est souvent choisie parmi des noyaux classiques, de manière empirique. L'exemple le plus simple de noyau est le noyau linéaire, qui peut être défini dans l'espace de Hilbert des formes linéaires par :

$$\forall \mathbf{x} \in X, K_{\mathbf{x}} : \mathbf{x}' \to K_{\mathbf{x}}\left(\mathbf{x}'\right) = \mathbf{x} \cdot \mathbf{x}'$$

Le noyau $K_{\mathbf{x}}$ exprime la distance entre les vecteurs **x** et **x'**.

Les fonctions noyaux non-linéaires usuelles sont :

- les noyaux polynomiaux : $K\left(\mathbf{x},\mathbf{y}\right) = \left\langle \mathbf{x},\mathbf{y} \right\rangle^d$ ou $K\left(\mathbf{x},\mathbf{y}\right) = \left(\left\langle \mathbf{x},\mathbf{y} \right\rangle + c\right)^d$, avec d entier et $c \geq 0$;
- les noyaux gaussiens : $K\left(\mathbf{x},\mathbf{y}\right) = \exp\left(-\frac{\left\|\mathbf{x}-\mathbf{y}\right\|^2}{2\sigma^2}\right)$;
- les noyaux sigmoïdes : $K\left(\mathbf{x},\mathbf{y}\right) = \tanh\left(\alpha + \beta\left\langle \mathbf{x},\mathbf{y} \right\rangle\right)$ où α et β sont des réels.

Il est également possible d'effectuer une combinaison linéaire de noyaux pour en obtenir un nouveau.

III.2.4 - Choix des hyperparamètres

La résolution des problèmes d'optimisation précédemment décrits nécessite de définir la valeur d'un ensemble d'hyperparamètres :

- C, qui représente le compromis entre la complexité du modèle et l'erreur sur les données d'apprentissage ;
- ε, la largeur du tube d'insensibilité ;
- les éventuels paramètres de la fonction noyau K.

Le choix de ces hyperparamètres peut être effectué par des méthodes de sélection de modèle telles que la validation croisée. Il est également possible de rechercher une première solution à partir d'un jeu d'hyperparamètres particulier, puis de chercher par descente du gradient le minimum d'un indice de performance tel que l'erreur de validation croisée. Ce gradient peut être calculé grâce au théorème qui suit [26].

Considérons la fonction : $L\left(\theta\right) = \max_{\mathbf{x} \in F}\left(\mathbf{x}^T \mathbf{v}_\theta - \frac{1}{2}\mathbf{x}^T \mathbf{P}_\theta \mathbf{x}\right)$

où : $F = \left\{\mathbf{x} : \mathbf{b}^T \mathbf{x} = c, \mathbf{x} \geq 0\right\}$;

$\mathbf{v}_\theta$ est un vecteur de dimension $\left(n,1\right)$;

$\mathbf{P}_\theta$ est une matrice (n,n) dépendant de façon continue du paramètre θ.

Soit $\mathbf{x}_0$ le vecteur pour lequel le maximum est atteint. S'il est unique, on montre que :

$$\frac{\partial L(\theta)}{\partial \theta} = \mathbf{x}_0^T \frac{\partial \mathbf{v}_\theta}{\partial \theta} - \frac{1}{2} \mathbf{x}_0^T \frac{\partial \mathbf{P}_\theta}{\partial \theta} \mathbf{x}_0$$

Il est donc possible de différentier L par rapport à θ comme si $\mathbf{x}_0$ ne dépendait pas de θ.

Les machines à vecteurs supports possèdent ainsi de nombreux atouts :

- l'apprentissage correspond à un problème d'optimisation convexe : la solution est unique ;
- le contrôle de la capacité de la fonction et la minimisation du risque sont simultanés ;
- l'utilisation des noyaux donne une grande souplesse à cette méthode, et permet de l'utiliser dans des domaines d'applications très divers.

III.3 - Les méthodes à noyaux pour la modélisation QSPR/QSAR

Depuis leur développement, et plus particulièrement depuis 2001 [27], les machines à vecteurs supports ont été utilisées avec succès pour la modélisation de propriétés et d'activités moléculaires [28].

III.3.1 - Modélisation par les SVM à partir de descripteurs traditionnels

Considérons tout d'abord l'approche qui consiste à représenter les molécules par un vecteur de descripteurs traditionnels, tels que les caractéristiques évoquées dans la section I.1. Les SVM constituent dans ce cas une méthode performante pour la recherche de la fonction de régression f reliant ces vecteurs de descripteurs aux grandeurs modélisées. Cette approche a par exemple montré son efficacité pour la classification de molécules selon leur activité pharmacologique [29, 30], mais aussi en régression. Les modélisations d'activités telles que la toxicité de phénols [31] ou l'activité inhibitrice de pyrimidines [27], ainsi que de propriétés telles que la solubilité aqueuse [32], ont en effet conduit à de très bons résultats, souvent supérieurs à ceux obtenus par d'autres méthodes sur les mêmes bases.

III.3.2 - Les fonctions noyaux pour molécules

Parallèlement à cette approche se sont développées les ***fonctions noyaux pour molécules*** (Graph Kernels for Structures) [33-35]. Cette méthode consiste à comparer des graphes représentant les liaisons covalentes des molécules, ou des ensembles d'atomes dans l'espace, à l'aide de fonctions noyaux (nous détaillons le principe de la représentation de molécules par des graphes dans le paragraphe I du Chapitre 3). À chaque molécule est associé un vecteur qui recense certains sous-éléments prédéfinis (par exemple des séquences d'atomes) du graphe correspondant. La distance entre deux graphes peut ensuite être évaluée

grâce à une fonction noyau, qui compare les vecteurs qui leur sont associés. L'utilisation de fonctions noyaux est particulièrement appropriée en bioinformatique, domaine dans lequel les données sont souvent de grande dimension.
Les vecteurs associés aux molécules peuvent être considérés comme des vecteurs de descripteurs, calculés cette fois à partir des graphes moléculaires, par opposition aux descripteurs traditionnels précédemment introduits. Cette représentation pose le problème du choix des descripteurs, mais de façon différente : les graphes associés aux molécules peuvent avoir des structures variées, et le choix des sous-éléments caractéristiques dépend de ces structures. On distingue par exemple :

- les noyaux de Fisher [36] et de spectres [37], notamment utilisés pour la classification de séquences biologiques, telles que les séquences d'acides aminés [38, 39] ;
- les noyaux de convolution [40], obtenus par convolution de plusieurs noyaux. Il est ainsi possible de comparer des séquences de longueurs différentes, à l'aide de différents noyaux, et d'obtenir un noyau global par convolution ;
- les noyaux marginalisés [34, 41, 42], permettant de comparer des données structurées plus variées, et pouvant être adaptés à la modélisation de propriétés de molécules, ou au traitement d'images.

La notion de distance permet de modéliser la propriété étudiée par le biais d'outils classiques de modélisation (réseaux de neurones, machines à vecteurs supports…). Cette approche, décrite plus en détail dans l'article [43] (reproduit dans l' annexe 2), présente l'avantage d'éviter le calcul et la sélection des descripteurs, mais demande en revanche de définir les sous-éléments caractéristiques des graphes. De plus, les modèles les plus performants demandent des temps de calcul qui peuvent devenir très grands.

IV - Autres méthodes de QSPR/QSAR

La modélisation d'une propriété ou d'une activité moléculaire nécessite de disposer d'informations caractérisant les molécules, informations à partir desquelles la grandeur en question est prédite. Il peut s'agir de descripteurs, mais il existe des méthodes alternatives de caractérisation des molécules. Nous présenterons dans un premier temps la méthode de contribution de groupes, qui, bien que datant des débuts de la modélisation QSAR, est toujours utilisée pour des applications particulières. Nous décrirons ensuite comment les techniques de calcul et de comparaison de champs moléculaires se révèlent particulièrement efficaces pour la modélisation d'activités biologiques. Nous introduirons finalement des méthodes récentes qui, de façon analogue aux *graph machines*, considèrent que les structures des molécules contiennent des informations dont il est possible de tirer directement profit pour la modélisation.

IV.1 - **Méthode de contribution de groupes**

Les méthodes de contribution de groupes consistent à évaluer une propriété en décomposant la molécule en un ensemble de groupes fonctionnels, et en sommant les contributions relatives à des fragments de molécules [44, 45]. Ces contributions sont déterminées à partir d'une base d'exemples de molécules, dont les valeurs de la propriété sont connues. Plusieurs types de groupes fonctionnels peuvent être définis. Ils sont généralement organisés en un système hiérarchique :

- Les ***groupes d'ordre 0*** sont des atomes, et le calcul d'une propriété est effectué en sommant les contributions de chacun des atomes de la molécule considérée.
- La décomposition en ***groupes d'ordre 1*** consiste à découper la molécule en groupes d'atomes (tels que $-CH_2-$, $-CH_3$ ou -OH). Leurs contributions à une propriété donnée sont sommées sans que l'environnement de chacun des groupes dans la molécule ne soit pris en considération. Ainsi, le groupe $-CH_2-$ a une contribution fixe, qu'il soit relié à un carbone ou à un groupe oxygéné. Ces groupes sont assez souvent employés, car ils permettent d'estimer rapidement la valeur d'une propriété, avec une précision parfois suffisante (par exemple pour l'enthalpie de formation). Cependant, les résultats obtenus, pour la température d'ébullition par exemple, ne sont pas toujours satisfaisants. De plus, certains isomères peuvent conduire à la même décomposition : il est alors impossible de les distinguer par cette méthode.
- ***Les groupes d'ordre 2*** [46, 47] sont constitués des atomes centraux de la molécule (autres que H), accompagnés de leurs plus proches voisins, c'est-à-dire de tous les atomes auxquels ils sont reliés. Contrairement aux groupes d'ordre 1, ceux d'ordre 2 tiennent compte de l'environnement des atomes.

Le Tableau 1 présente une comparaison des décompositions des molécules de butan-2-ol et 2-méthylpropan-1-ol, représentées sur la Figure 4.

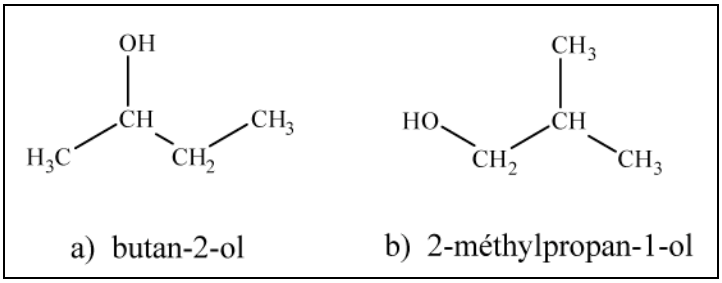


Figure 4 : Exemple de deux molécules, isomères de constitution

Méthode	Groupe	Nombre de groupes a)	Nombre de groupes b)
Ordre 0	C	4	4
	H	10	10
	O	1	1
Ordre 1	CH	1	1
	$-CH_2-$	1	1
	$-CH_3$	2	2
	-OH	1	1
Ordre 2	$C-(C)(H)_3$	2	2
	$C-(C)_2(H)_2$	1	0
	$C-(C)(O)(H)_2$	0	1
	$C-(C)_3(H)$	0	1
	$C-(C)_2(O)(H)$	1	0

Tableau 1 : Décomposition en groupes de deux isomères de constitution

On observe que seule la décomposition en groupes d'ordre 2 permet de distinguer ces deux molécules.

De nombreuses méthodes s'appuient donc sur des groupes des trois ordres pour améliorer la précision des prédictions et différencier les isomères. Elles sont principalement utilisées pour prédire des propriétés thermodynamiques, par exemples des propriétés critiques (température ou pression critique) et de nombreuses grandeurs énergétiques. Elles présentent également l'avantage de permettre l'estimation de propriétés de mélanges, par addition des contributions des composants du mélange.

IV.2 - Analyse comparative de champs moléculaires (CoMFA)

La méthode CoMFA (pour Comparative Molecular Field Analysis) a été développée à partir de 1988 par Cramer [48]. Elle est en particulier utilisée pour la modélisation d'activités biologiques, pour lesquelles les méthodes classiques se révèlent parfois peu performantes. L'activité biologique d'une molécule dépend généralement de son interaction avec un récepteur donné. La modélisation de cette activité peut donc être réalisée en calculant les interactions de chaque molécule (ligand) avec ce récepteur, et en établissant une relation entre ces interactions et l'activité étudiée. Cette modélisation, qui repose sur le calcul de potentiels d'interactions moléculaires, s'effectue en plusieurs étapes.

1) Recherche des conformations moléculaires les plus stables :

Des programmes tels que Concord ou Corina [49] permettent d'accéder à une conformation de basse énergie, à partir de laquelle sont créées plusieurs conformations bioactives.

2) Alignement des ligands :

Les potentiels d'interactions moléculaires dépendent des orientations des structures utilisées pour leur calcul. La comparaison des potentiels de différentes molécules nécessite donc un alignement préalable de ces structures. Il s'agit d'une étape assez délicate, en particulier lorsque les composés ont des structures diverses. Cette étape est même susceptible de limiter les possibilités d'application de la méthode. L'alignement est réalisé par rapport aux groupes fonctionnels identifiés comme potentiellement pharmacophores. Il existe plusieurs méthodes pour réaliser cet alignement. La plus simple est la méthode de superposition par sous-structures, qui consiste à superposer les molécules qui partagent un squelette commun, selon ce squelette. L'approche par superposition de pharmacophores ne nécessite pas cette supposition, mais part du fait que les pharmacophores de chaque molécule sont identifiés. Il s'agit ensuite de maximiser la superposition de ces groupements entre les molécules. Il est également possible d'aligner les molécules par rapport à leurs moments dipolaires ou à leurs champs électrostatiques.

3) Calcul des champs électrostatiques et stériques :

Les champs électrostatiques et stériques sont ensuite évalués. Pour cela, on définit autour de chaque molécule, préalablement alignée, une grille 3D. Puis on calcule, en chaque point de la grille, l'énergie d'interaction de la molécule avec un atome sonde.

4) Corrélation entre les champs et les activités biologiques :

Les valeurs calculées des champs peuvent alors être utilisées comme variables du modèle. Le nombre de ces variables peut être très grand (quelques milliers), lorsque la résolution de la grille est fine par rapport à la taille des molécules. Il n'est alors pas possible de corréler directement ces valeurs de champs avec l'activité étudiée. La modélisation est généralement effectuée par la méthode des moindres carrés partiels, et le nombre optimum de variables final déterminé par validation croisée (voir section II.3.2.1 de ce chapitre).

5) Une **visualisation graphique** des résultats permet enfin de situer autour d'une molécule les régions pour lesquelles une substitution augmente ou diminue l'affinité envers le récepteur.

Cette méthode a montré son efficacité en QSAR, en particulier pour la modélisation d'interactions ligand-protéine. Elle permet en effet de décrire efficacement les interactions ligand-récepteur, car les propriétés des ligands sont calculées à partir de leurs conformations bioactives. Les problèmes sont néanmoins multiples :

– Les résultats de modélisation dépendent de la conformation bioactive choisie. Or, la recherche de cette dernière est parfois difficile, en particulier lorsque la molécule est flexible : il existe alors un nombre important de conformations possibles près du minimum d'énergie.

– Il n'existe pas de règle générale pour l'alignement des molécules. Il s'agit d'un défaut majeur, car le modèle est très sensible à cet alignement.

– De plus, le temps de calcul est généralement assez long (30 à 60 min pour une vingtaine de molécules possédant de 30 à 50 atomes).

Puisque l'alignement des molécules est une limitation de la méthode CoMFA, de nouvelles techniques ne nécessitant pas cette étape ont été développées. La **méthode CoMSA** (pour Comparative Molecular Surface Analysis), ne repose pas sur la comparaison de champs calculés en une série donnée de points, mais celle du potentiel électrostatique moyen de régions définies de la surface de la molécule. Ces régions sont nombreuses, et la méthode des cartes auto-organisatrices de Kohonen [50] permet de réduire la dimension des données tout en préservant leur topologie. Cette méthode est particulièrement adaptée pour transformer la surface tridimensionnelle de la molécule en une carte bidimensionnelle du potentiel électrostatique. La transformation de Kohonen permet en effet à la fois de diminuer la taille des données, et de retrouver les données 3D à partir de leur représentation 2D. Les vecteurs obtenus sont alors analysés et corrélés à l'activité étudiée, par la méthode des moindres carrés partiels.

IV.3 - Modélisation à partir des structures des molécules

Ces dernières années ont vu l'apparition de nouvelles techniques de modélisation qui ne demandent pas la conversion des molécules en un vecteur de descripteurs, mais établissent une relation directe entre les structures des molécules et leur propriétés. Ces techniques, tout comme la méthode que nous avons adoptée, reposent sur la représentation des molécules par des graphes (cette notion est détaillée dans la section I.1 du chapitre III).
La méthode des ***réseaux de neurones récursifs*** [51], ou RNNs, consiste à représenter les structures acycliques par des graphes, et à coder ces graphes de manière récursive par des réseaux de neurones. Cette technique, décrite plus précisément dans la section III du chapitre 2, et dans nos récents articles [43, 52, 53], est cependant limitée aux graphes acycliques, et ne permet pas de modéliser les propriétés de molécules cycliques.
Les techniques alternatives que nous venons de décrire répondent donc généralement à des besoins précis, mais elles ont pour cette raison des domaines d'application limités. Nous montrerons dans la suite de ce mémoire que la méthode des *graph machines*, tout en s'affranchissant des inconvénients liés à l'utilisation de descripteurs, ne présentent pas cette limitation.

CHAPITRE 2
Les Graph Machines

Les outils classiques de modélisation que nous avons présentés, tels que les réseaux de neurones ou les machines à vecteurs supports, réalisent pour la plupart une association entre un vecteur de nombres réels, qui constituent les variables du modèle, et un vecteur de sorties également réelles. Cependant, dans de nombreux problèmes de modélisation, les données se présentent sous la forme de structures dont il faut extraire un vecteur de réels si l'on veut avoir recours à des modèles classiques. Cette étape nécessite de sélectionner les données pertinentes relatives au problème et de les calculer à partir des structures ; elle se traduit par une perte d'information. Il semblerait donc plus pertinent de tirer directement profit de la structure des données, par l'intermédiaire d'un autre type de modèle, capable d'établir de façon directe une association entre ces structures et un vecteur de sorties. Les données structurées se représentent aisément sous la forme de graphes. Nous commencerons par présenter les graphes en tant qu'objets mathématiques, puis en tant que représentations de données structurées. Nous introduirons ensuite les mémoires récursives auto-associatives, premiers modèles à réaliser un codage de données structurées par apprentissage artificiel, et nous présenterons le développement qui nous a permis d'arriver aux *graph machines*, capables de réaliser une modélisation à partir de graphes quelconques.

I - Définition et caractéristiques d'un graphe

La théorie des graphes date du 18ème siècle, et s'est considérablement développée au 20ème siècle, car elle permet de résoudre de nombreux types de problèmes [54]. Nous allons présenter les bases de cette théorie, et les définitions essentielles relatives aux graphes, puis nous illustrerons leur aptitude à représenter différents types de données, telles que des images ou des textes.

I.1 - Graphes simples

Un ***graphe simple*** G est un groupe $\{S, A\}$ où :

- S est ensemble d'objets $\{s_1, s_2, ..., s_n\}$, appelés sommets du graphe ;
- A est un sous-ensemble de $S \times S$, dont les éléments $\{a_1, a_2, ..., a_m\}$ sont les arêtes du graphe.

Un graphe est ***connexe*** s'il est possible, à partir de n'importe quel sommet, de rejoindre tous les autres.

L'arête $a_k = (s_i, s_j)$ est incidente aux nœuds s_i et s_j, qui sont dits adjacents. Le degré d'un nœud est défini comme le nombre d'arêtes qui lui sont incidentes. Un ***cycle*** est une chaîne $\langle s_1, s_2, \ldots, s_k \rangle$ dont le premier et le dernier sommet sont identiques et tous les autres sommets distincts. Si le graphe possède *m* arêtes $\{a_1, a_2, ..., a_m\}$, on peut faire correspondre à tout cycle μ un vecteur $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_m)$ tel que :

$$\mu_i = \begin{cases} 1 \text{ si } a_i \in \boldsymbol{\mu} \\ 0 \text{ si } a_i \notin \boldsymbol{\mu} \end{cases}$$

On dit que *p* cycles $\boxed{\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \ldots, \boldsymbol{\mu}^p}$ sont dépendants s'il existe entre leurs vecteurs associés une relation vectorielle de la forme :

$$\sum_{i=1}^{p} \lambda_i \boldsymbol{\mu}^i = 0$$

telle que $(\lambda_1, \lambda_2, ..., \lambda_p) \in \mathbb{R}^p$ et $(\lambda_1, \lambda_2, \ldots, \lambda_p) \neq (0,0,\ldots,0)$.

On appelle ***distance*** entre deux sommets la longueur de la plus courte chaîne reliant ces sommets ; le ***diamètre*** d'un graphe est alors la plus grande distance entre les sommets d'un graphe.

I.2 - Graphes orientés

Un ***graphe orienté*** est un groupe $G = \{S, A\}$, où A est un ensemble d'arcs de la forme (s_i, s_j), pour lequel l'arc part de s_i et arrive en s_j. Le degré entrant d^- d'un sommet est le nombre d'arcs qui arrivent à ce sommet, et le degré sortant d^+ le nombre d'arcs qui en partent.

Un ***arbre*** est un graphe connexe sans cycle. Dans un arbre orienté, si les sommets s_i et s_j sont reliés par l'arc (s_i, s_j), s_i est ***parent*** du nœud s_j, qui est un ***enfant*** du nœud s_i. Les nœuds qui possèdent à la fois des nœuds parents et enfants sont appelés des ***branches***. Un ***nœud racine*** est un nœud sans parent, tandis que les ***feuilles*** sont les nœuds sans enfant. Un arbre est un arbre *n*-aire si tous les nœuds de l'arbre ont au plus *n* successeurs.

Une ***arborescence*** $\{S, A, r\}$ de racine r est un graphe $\{S, A\}$ où r est un élément de S tel que, pour tout sommet s, il existe un unique chemin d'origine r et d'extrémité s. On montre facilement que, dans une arborescence, la racine r n'admet pas de prédécesseur, et que tout sommet différent de r admet un seul prédécesseur.

La Figure 5, qui représente une arborescence, illustre les concepts que nous venons de définir.

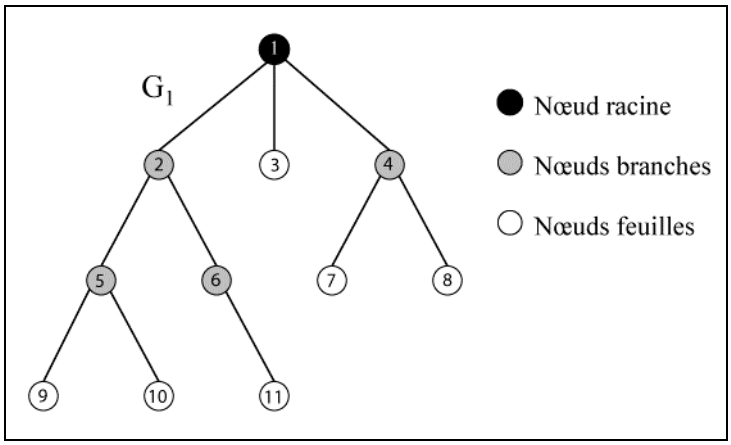


Figure 5 : Exemple d'arborescence

I.3 - Graphes étiquetés

Il est possible d'affecter aux nœuds et aux arcs d'un graphe orienté un ensemble d'étiquettes : le graphe est alors ***étiqueté***. Étant donné un ensemble fini d'étiquettes de sommets L_V, et un ensemble fini d'étiquettes d'arcs L_A, un graphe étiqueté est défini par un triplet $\langle S, r_S, r_A \rangle$ où :

- S est l'ensemble des sommets ;
- $r_S \subseteq S \times L_S$ est l'ensemble des couples (s_i, l_s) tels que le sommet s_i a pour étiquette l_s ;
- $r_A \subseteq S \times S \times L_A$ est l'ensemble des triplets (s_i, s_j, l_a) tels que l'arc (s_i, s_j) a pour étiquette l_a.

I.4 - Matrices d'adjacence

Plusieurs représentations sont possibles pour les graphes. Elles ne sont pas équivalentes, et le choix d'une représentation adaptée au problème à résoudre permet d'obtenir une meilleure efficacité des algorithmes utilisés. On distingue ainsi la représentation par matrice d'adjacence, par matrice d'incidence sommets-arcs (ou sommets-arêtes), et par liste d'adjacence. Nous utilisons la matrice d'adjacence, car cette représentation permet de calculer simplement certaines caractéristiques des graphes, telles que la distance entre les sommets.

La matrice d'adjacence d'un graphe $G = \{S, A\}$ est la matrice M(G) dont les coefficients $m_{i,j}$ sont définis par :

$$m_{i,j} = \begin{cases} 1 \text{ si } (s_i, s_j) \in A, \text{ c'est-à-dire si les noeuds } s_i \text{ et } s_j \text{ sont adjacents} \\ 0 \text{ si } (s_i, s_j) \notin A \end{cases}$$

La Figure 6 représente un graphe ainsi que la matrice d'adjacence qui lui correspond.

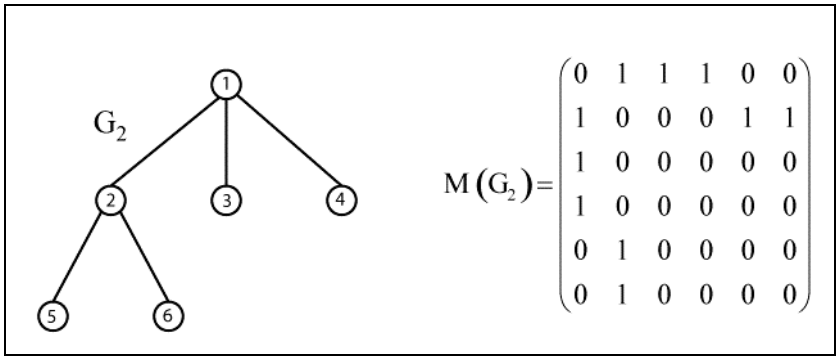


Figure 6 : Représentation d'un graphe par sa matrice d'adjacence

Si n est le nombre de nœuds du graphe, cette matrice est de taille (n, n), et chaque ligne ainsi que chaque colonne correspond à un nœud particulier. Puisque la matrice dépend de la numérotation des nœuds du graphe, la représentation n'est pas unique (il existe $n!$ possibilités). Il est cependant possible de passer d'une représentation à une autre par permutation de lignes et de colonnes.

La matrice d'adjacence permet d'énumérer les chemins du graphe, et d'en déduire la connexité, la cyclicité ainsi que les distances entre les nœuds. On utilise pour cela le produit matriciel. Par exemple, la matrice $\mathbf{M}^2$ a pour coefficients :

$$(\mathbf{M}^2)_{i,j} = \sum_{k=1}^{n} m_{i,k} m_{k,j}$$

Chaque composant de la somme est donc non nul ssi il existe une arête de s_i à s_k et de s_k à s_j, c'est-à-dire s'il existe un chemin de longueur 2 reliant s_i à s_j et passant par s_k. Par extension, on peut démontrer que si **M** est la matrice d'adjacence d'un graphe G dont les sommets sont numérotés de 1 à n, le nombre de chemins de longueur exactement l allant de s_i à s_j est le coefficient (i, j) de la matrice $\mathbf{M}^l$.

Un algorithme permet alors de calculer la distance entre deux nœuds s_i et s_j :

$$L_{i,j} = \min\ l,\ \left(\mathbf{M}^l\right)_{i,j} \neq 0 \tag{25}$$

On en déduit le diamètre du graphe :

$$D = \max_{(s_i, s_j) \in S^2} L_{i,j} \tag{26}$$

Il est par ailleurs possible de déterminer la connexité d'un graphe, c'est-à-dire le nombre de composantes connexes, et d'en déduire le nombre de cycles indépendants que comporte le graphe :

$$N_{cycles} = N_{arêtes} - N_{nœuds} + connexité \tag{27}$$

Cette représentation des graphes permet ainsi d'accéder aisément aux principales caractéristiques d'un graphe, ainsi que d'effectuer des opérations telles que la suppression d'une arête ou d'un cycle.

II - Représentation de données structurées par des graphes

Les graphes permettent de modéliser de nombreux types de données, et les représentent de façon à conserver l'information contenue dans leur structure. Ils sont par exemple bien adaptés à l'analyse d'images et à la reconnaissance de formes. L'utilisation des données sous la forme de graphes permet en effet de tenir compte des propriétés et des relations causales, hiérarchiques et topologiques entre les différentes parties des objets, et permet d'éviter la perte de l'information liée à la structure des données.

Les images, qui peuvent être considérées comme des données structurées, donc être représentées par des graphes, se prêtent à ce type d'approche [55]. La première étape consiste à détecter les régions qui constituent des éléments simples de l'image, à l'aide d'un algorithme de segmentation par approche contour ou frontière. Chaque région ainsi distinguée est alors assimilée à un nœud d'un graphe, qui peut être étiqueté pour tenir compte des spécificités de la région, par exemple sa forme, sa couleur ou sa taille. Les nœuds sont alors reliés par des arêtes ou des arcs qui traduisent des relations de hiérarchie, de contigüité ou d'inclusion des régions correspondantes. Considérons par exemple l'image représentée sur la Figure 7. La segmentation de l'image permet de distinguer les contours a-k. Le graphe correspondant peut alors être construit en affectant à chaque contour un nœud, et en reliant ces nœuds par des arcs en faisant correspondre à la relation d'inclusion une relation parent-enfant : le nœud racine correspond au contour externe de l'image s'il existe, ou à l'image entière dans le cas contraire, puis tout contour inclus dans un autre contour est désigné comme enfant de ce contour. Par exemple, le nœud qui correspond à b dans l'image est enfant de celui correspondant à a, qui est le nœud racine. Le graphe orienté résultant de cet algorithme est représenté sur la Figure 8.

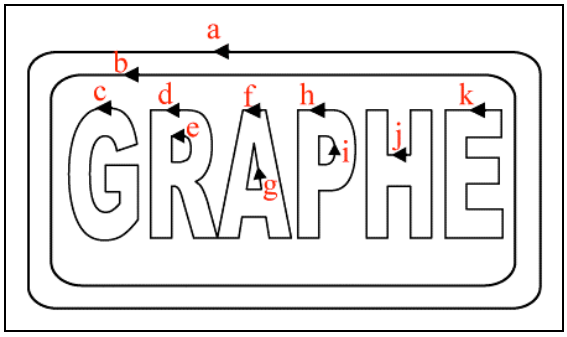


Figure 7 : Segmentation d'une image par l'approche contour

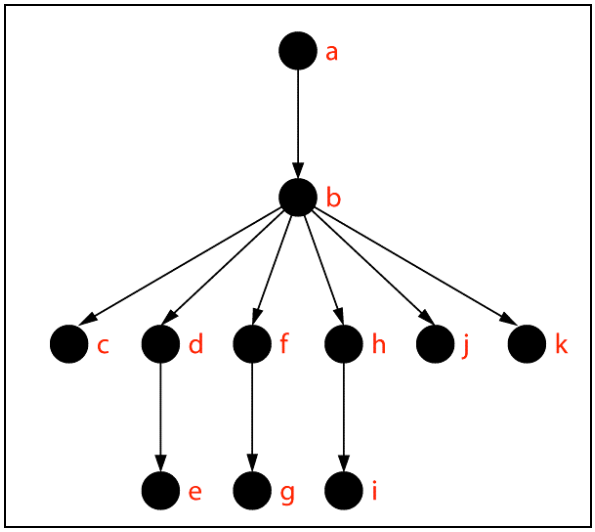


Figure 8 : Graphe associé à l'image de la Figure 7

Il est à noter que cet algorithme produit le même arbre quelle que soit la transformation (symétrie, translation…) appliquée à l'image. Des améliorations de cet algorithme sont possibles ; elles permettent de représenter efficacement des images bruitées.
Lorsque la segmentation vise à distinguer non plus des contours mais des régions de l'image, ce sont à ces régions que sont associés les nœuds d'un graphe. Ils peuvent être caractérisés par un vecteur d'étiquettes contenant des caractéristiques locales de la région (aire, couleur…) ou des relations géométriques avec les autres régions (positions ou orientations relatives…). Les nœuds associés à des régions adjacentes sont alors reliés, et le graphe résultant peut être orienté suivant une méthode choisie. La Figure 9 représente une image (a) et le graphe non-orienté (b), puis orienté (c) qu'il est possible de lui associer [56].

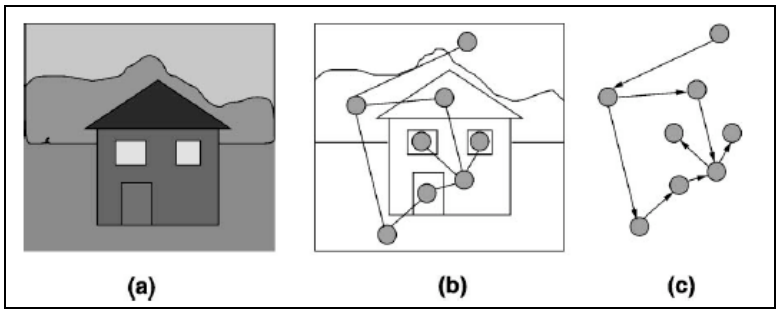


Figure 9 : Représentation par un graphe d'une image segmentée en régions

Pour des tâches de modélisation telles que la classification de textes ou la résolution d'ambigüités, les documents, qui sont des suites de caractères, doivent être transformés en une représentation exploitable par les outils de modélisation utilisés. Une méthode usuelle, dite en « sac de mots », consiste à construire à partir du document un vecteur de grande dimension, pour lequel chaque valeur correspond à la présence ou à l'absence d'un mot [57]. L'inconvénient majeur de cette méthode est la perte de l'information contenue dans l'ordre des mots, car seule leur fréquence est conservée. Il est alors avantageux d'utiliser des graphes pour représenter ces textes [58, 59], car ils permettent de tenir compte de la structure interne de ceux-ci et des relations entre les mots. Chaque mot ou groupe de mots peut être assimilé à un nœud, auquel est affectée une « étiquette de partie du discours » (ou POS-tag, pour Part of Speech tag), qui renseigne sur la nature du mot (adjectif, abréviation, participe passé...). Selon la nature du problème à résoudre, un ou plusieurs graphes peuvent alors être obtenus pour chaque phrase en liant ces nœuds par des arcs, à l'aide des étiquettes des groupes de mots. Supposons par exemple que l'on souhaite lever une ambigüité (désambigüisation sémantique) liée à la phrase :

« La syntaxe de cette phrase qui est ambigüe doit être éclaircie »

L'ambigüité est ici due au groupe de mots « qui est ambigüe », qui peut être relatif à « la syntaxe » ou à « cette phrase ». Il est ainsi possible, après étiquetage des groupes de mots, de construire deux graphes différents. Un modèle préalablement établi, après apprentissage sur

des phrases correctement transformées en graphes, permet alors d'affecter à chacun de ces deux graphes une probabilité de correspondre à la réelle interprétation de la phrase.
Par ailleurs, lorsqu'un document est composé d'informations de natures différentes (titres, corpus, liens par exemple), il peut être représenté par un graphe acyclique orienté, qui tient compte à la fois de sa structure et de son contenu. La structure du graphe est déterminée par celle du document, et les étiquettes associées aux noeuds permettent de tenir compte de son contenu [60, 61]. Cette représentation est avantageuse pour des documents HTML ou XML par exemple.
Les graphes peuvent ainsi représenter de nombreux types de données structurées, et sont employés pour résoudre les problèmes associés, tels que des problèmes de planification de transport, de câblage de circuits imprimés ou intégrés, etc.

III - Apprentissage à partir de graphes : RAAMs et LRAAMs

Nous allons maintenant expliquer comment il est possible d'établir une relation entre un ensemble de données structurées, représentées par des graphes, et un ensemble de nombres réels, réalisant ainsi l'association structures-données réelles évoquée. Les premiers essais de modélisation de données structurées remontent aux mémoires associatives dynamiques. Ces essais ont par la suite conduit aux Mémoires Auto-Associatives Récursives (RAAMs), qui permettent de représenter de façon compacte les arbres, et aux RAAMs étiquetés (LRAAMs).

III.1 - Les Mémoires Auto-Associatives Récursives

Afin de montrer la capacité des réseaux de neurones à manipuler des données structurées, Pollack a proposé un modèle, fondé sur l'idée d'une mémoire associative entre une structure et un vecteur de taille fixe [62]. Les RAAMs (pour *Recursive AutoAssociative Memory*) et leurs dérivées sont des modèles établis à partir de réseaux de neurones, qui apprennent un codage d'une structure en un vecteur, puis sont à même de décoder la structure d'origine à partir de ce vecteur avec une perte d'information minimale.
L'élément de base des RAAMs est un codeur-décodeur constitué par un réseau de neurones possédant autant de variables que de sorties ($N_e = N_s$), et dont le nombre de neurones cachés est inférieur au nombre de variables ($N_c < N_e$). Un exemple d'un tel codeur-décodeur est représenté sur la Figure 10. À l'issue de l'apprentissage, le système réalise une auto-association, car les sorties du modèle sont identiques aux variables : la base d'apprentissage est du type $\left(\left\{\mathbf{x}_k, \mathbf{x}_k\right\}, 1 \le k \le n\right)$.
Lorsque le problème admet une solution, les neurones cachés réalisent un codage du vecteur de variables **x** sous forme compacte **f**(**x**) (car $N_c < N_e$), où **f**(**x**) désigne le vecteur des sorties des neurones cachés. Ce vecteur **f**(**x**) peut ensuite être décodé, afin de retrouver en sorties les données d'entrée.

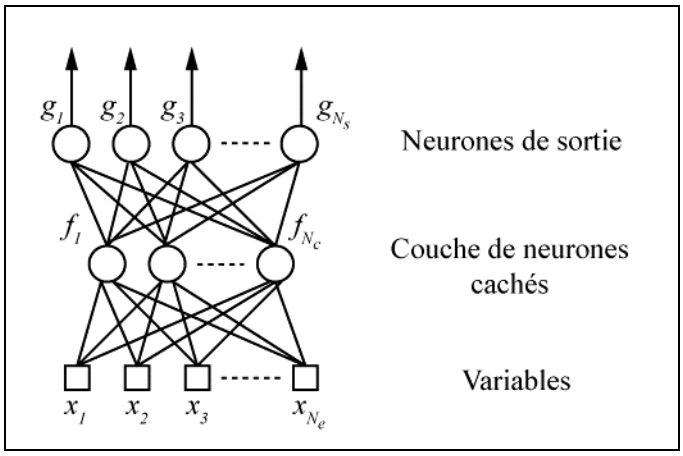


Figure 10 : Codeur-décodeur constitué de neurones formels

Cette unité codeur-décodeur peut être utilisée pour effectuer le codage d'un arbre, de différentes manières. Dans l'article [63], le codage s'effectue de la façon suivante : les feuilles de l'arbre, c'est-à-dire les nœuds sans enfant, sont tout d'abord codées, puis leurs parents et les nœuds suivants, de façon récursive, jusqu'au nœud racine. Le vecteur obtenu en sortie du codeur correspondant au nœud racine est alors une représentation compacte de l'ensemble de l'arbre. Le décodage s'effectue de façon symétrique et de manière récursive, fournissant successivement le décodage du nœud racine jusqu'aux feuilles de l'arbre. Il est alors possible de réaliser un apprentissage de l'ensemble codeur-décodeur obtenu, de façon à retrouver en sortie un vecteur égal au vecteur d'entrées. Considérons par exemple l'arbre binaire $\left(\left(\mathbf{A},\mathbf{B}\right),\left(\mathbf{C},\mathbf{D}\right)\right)$ de la Figure 11 où **A**, **B**, **C** et **D** sont des vecteurs de taille identique.

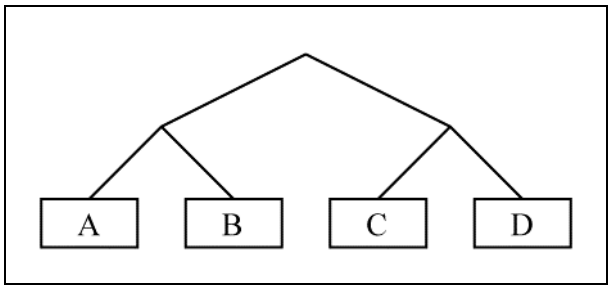


Figure 11 : Arbre binaire

A et **B** sont tout d'abord codés en une représentation R_1, **C** et **D** en une représentation R_2, puis R_1 et R_2 sont codés à leur tour en R_3, qui est une forme compacte de l'arbre. Pour retrouver l'information initiale, il faut alors décoder R_3 en $\left(R_1^{'},R_2^{'}\right)$, à leur tour respectivement décodés en $\left(\mathbf{A}',\mathbf{B}'\right)et\left(\mathbf{C}',\mathbf{D}'\right)$. L'apprentissage est séquentiel : il se fait tout d'abord sur le codeur-décodeur pour auto-associer $\left(\mathbf{A},\mathbf{B}\right)$ à$\left(\mathbf{A},\mathbf{B}\right)$, puis sur la paire$\left(\mathbf{C},\mathbf{D}\right)$, et enfin sur la paire$\left(R_1,R_2\right)$. Ce type de codage pose alors le problème de la « cible mobile» : les représentations des vecteurs non-terminaux (dans le cas de l'exemple évoqué, R_1 et R_2), donc

une partie de la base d'apprentissage, changent au cours de l'apprentissage. La convergence de celui-ci n'est alors plus garantie.

Pour nous affranchir de ce problème, nous proposons une méthode de codage légèrement différente. Le principe de base consiste à créer, à partir de plusieurs codeurs-décodeurs tels que ceux précédemment décrits, un modèle dont la structure est isomorphe à celle des arbres à coder, à partir de réseaux de neurones à poids partagés. Ce modèle effectue alors, avec le même jeu de paramètres, le codage des paires $(\mathbf{A},\mathbf{B})$, $(\mathbf{C},\mathbf{D})$ et (R_1,R_2).

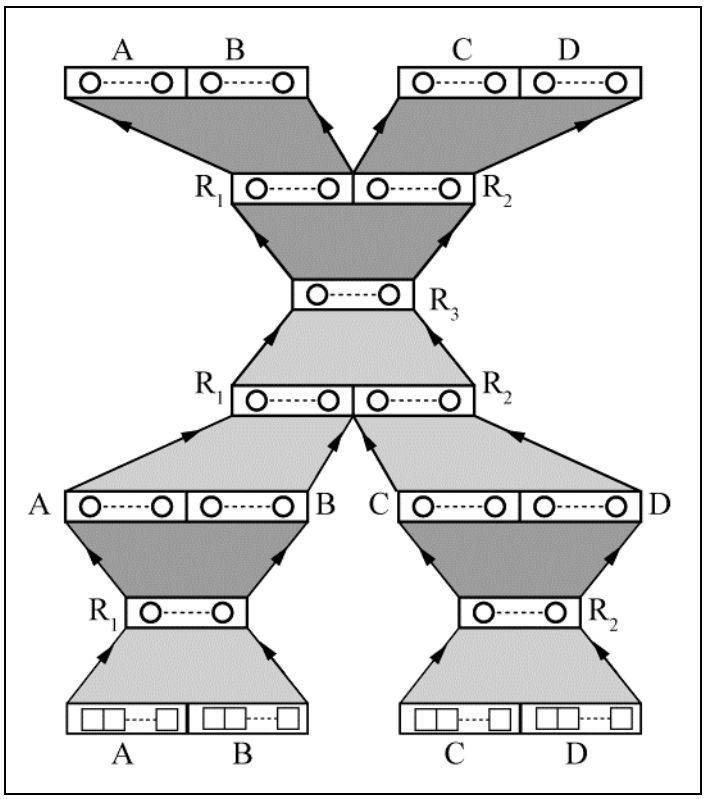


Figure 12 : Modèle associé au graphe de la Figure 11

La Figure 12 représente l'ensemble « codeurs-décodeurs » correspondant au modèle associé à l'arbre de la Figure 11. Les connexions ne sont pas détaillées, mais représentées par des zones grisées : les zones grisées de même teinte correspondent à des connexions de même vecteur de paramètres. Sur ce schéma, les carrés n'effectuent aucun calcul. On peut remarquer que la structure de ce réseau est effectivement identique à celle de l'arbre qui est codé.

Les zones grisées peuvent représenter non pas une seule couche de paramètres, mais deux couches de paramètres et une couche de neurones cachés. Supposons que les zones grisées représentent une seule couche de paramètres. Le nombre de neurones cachés doit alors être égal au nombre de neurones nécessaires pour coder une feuille de l'arbre. Or, la complexité de la relation établie entre les entrées et les sorties, qui est liée au nombre de neurones cachés, n'est en général pas liée au nombre de neurones nécessaire au codage des feuilles. Il est possible de se libérer de cette contrainte en intercalant une couche cachée supplémentaire dans le codeur et/ou le décodeur. Dans ce cas, les zones grisées ne représentent plus une seule

couche de paramètres, mais deux, ainsi qu'une couche cachée. Les zones grisées de même teinte correspondent alors à des paramètres et à des neurones identiques.

Les RAAMs permettent également de coder des séquences (elles sont alors appelés SRAAMs, pour Sequential RAAMs). Considérons par exemple la séquence $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ représentée sur la Figure 13. Le codage est récursif : si l'on note R_x la représentation codée d'un vecteur $\mathbf{x}$, les éléments d'entrée des codeurs successifs sont $(0\ \mathbf{x}_1)$, $(R_{x_1}\ \mathbf{x}_2)$ et $(R_{x_1x_2}\ \mathbf{x}_3)$.

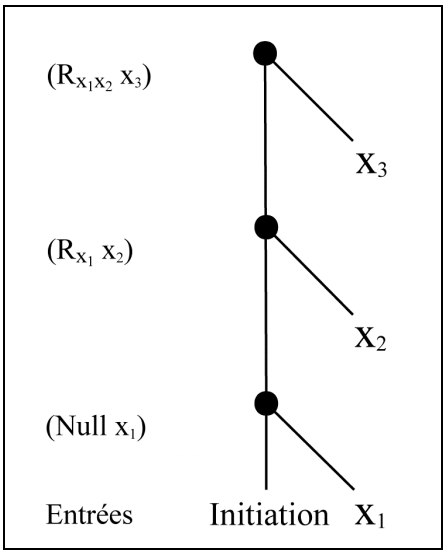


Figure 13 : Codage de la séquence (x_1, x_2, x_3) par une SRAAM

La couche d'entrée est ainsi composée de deux sous-éléments, l'un contenant le vecteur de variables correspondant au nœud à coder (c'est-à-dire $\mathbf{x}_1$, $\mathbf{x}_2$ ou $\mathbf{x}_3$), l'autre au vecteur nul, puis, au fur et à mesure du codage, à un pointeur vers le nœud enfant, qui est la représentation de la sous-séquence déjà codée (R_{x_1} ou $R_{x_1x_2}$).

III.2 - Les Mémoires Récursives Auto-Associatives Étiquetées

Les Mémoires Récursives Auto-Associatives Étiquetées (LRAAMs, pour Labeling RAAMs) [63] sont des RAAMs qui permettent de tenir compte des étiquettes d'un graphe. Le principe des LRAAMs consiste à considérer deux types d'entrées pour chaque nœud : une partie des entrées correspond aux pointeurs vers les enfants du nœud dans le graphe, comme précédemment, et une seconde partie correspond aux étiquettes des nœuds.
À chaque nœud du graphe est ainsi associée une entrée, qui comporte un vecteur de variables (qui peuvent être codées de façon binaire) associé à ses étiquettes, et un ensemble de pointeurs vers ses enfants dans le graphe, qui sont des nombres réels, et dont le nombre est égal à son degré sortant. Ces données constituent donc les entrées des codeurs associés à

chaque nœud. Les SRAAMs sont en ce sens des LRAAMs, pour lesquelles il n'y a qu'un seul pointeur.
Ceci soulève une nouvelle fois le problème de la cible mobile, car les pointeurs des nœuds non-terminaux sont calculés lors de l'apprentissage, donc varient. Ce problème peut être évité en utilisant la même idée que celle présentée dans le cadre des RAAMs, c'est-à-dire en associant à l'arbre étiqueté un modèle dont la structure est isomorphe à celle de l'arbre. À chaque nœud sont associés un codeur et un décodeur, et les réseaux de neurones constituant les codeurs-décodeurs des différents nœuds sont à poids partagés : ces réseaux partagent le même jeu de paramètres.

IV - Les Graph Machines

Nous avons présenté, dans la section précédente, des techniques qui permettent de manipuler des données structurées par apprentissage artificiel.
Nous allons maintenant montrer comment il est possible, sur cette base, d'établir une relation entre des données structurées et des vecteurs de réels, de même que les outils de modélisation traditionnels établissent une relation entre des vecteurs de variables et les vecteurs de données à modéliser.

IV.1 - Modélisation à partir de graphes acycliques

Nous avons rappelé comment les RAAMs et les LRAAMs permettaient de trouver une représentation compacte d'un arbre, sous forme vectorielle, en sortie du codeur. Cette représentation vectorielle de l'arbre peut alors être exploitée, non plus par un décodeur pour retrouver l'information d'origine, mais comme vecteur d'entrée d'un classifieur, par exemple un réseau de neurones [64]. De plus, l'apprentissage de la représentation et celui du classifieur peuvent être réalisés simultanément : l'apprentissage du codeur est effectué dans le contexte de la « cible mobile » tandis que celui du classifieur se fait de manière conventionnelle. Nous pouvons cependant éviter le problème de la cible mobile de la manière décrite dans la section III.1 : à chaque graphe est associé un réseau de structure identique, et les codeurs correspondant à chacun des nœuds sont à poids partagés.

IV.2 - Structure mathématique des *graph machines*

La première étape, qui constitue l'idée principale sous-jacente aux *graph machines*, consiste à construire, pour chacun des graphes, une fonction mathématique réalisée en combinant des fonctions élémentaires, selon une combinaison décrite par le graphe qui lui est associé.
Considérons un ensemble de graphes acycliques $\mathcal{G} = \{G_i\}$. Pour chaque graphe G_i, on construit une fonction g^i de la façon suivante : à chaque nœud de G_i est associée la fonction

paramétrée dite « fonction de nœud » $f_{\boldsymbol{\theta}}$, où $\boldsymbol{\theta}$ est le vecteur des paramètres, qui est identique pour toutes les fonctions. La fonction associée au nœud racine peut être différente des fonctions relatives aux autres nœuds : nous la noterons $F_{\boldsymbol{\Theta}}$. Les fonctions $f_{\boldsymbol{\theta}}$ sont alors composées de façon à refléter la structure du graphe : si s_j et s_k sont deux sommets du graphe G_i, tels qu'un arc part de s_j et arrive en s_k, alors le résultat de la fonction associée au nœud s_j est argument de celle associée au nœud s_k.
La fonction de nœud d'un nœud s_k est ainsi de la forme :

$$f_{\boldsymbol{\theta}}(\mathbf{z}_k) = f_{\boldsymbol{\theta}}(z_0, \mathbf{v}_k, \mathbf{x}_k).$$

Les arguments de la fonction sont de plusieurs types :

- z_0 est une constante égale à 1.
- $\mathbf{v}_k$ est un vecteur dont les composantes sont égales aux valeurs prises par les fonctions associées aux nœuds enfants du nœud s_k. Puisque la fonction $f_{\boldsymbol{\theta}}$ est la même pour tous les nœuds, ce vecteur doit avoir un nombre fixe de composantes, que nous définissons ainsi : $M_i = \underset{k}{\operatorname{argmax}}\, d_k^+$, où d_k^+ est le degré sortant du nœud s_k.

  Pour un nœud s_k tel que $d_k^+ < M_i$, les composantes superflues de $\mathbf{v}_k$ sont égales à 0.
- $\mathbf{x}_k$ est un vecteur optionnel qui apporte de l'information sur le nœud : ce sont les étiquettes du nœud.

$\mathbf{z}_k$ est ainsi un vecteur de taille D_i tel que :

$$D_i = 1 + M_i + \left|\mathbf{x}_k\right|$$

Sa première composante z_0 vaut 1, les composantes 2 à $d_k^+ + 1$ sont les valeurs prises par les fonctions f_θ des nœuds enfants. Si $d_k^+ < M_i$, c'est-à-dire si le nombre de nœuds enfants du nœud s_k est inférieur au maximum M_i possible, les composantes $d_k^+ + 2$ à $M_i + 1$ sont nulles. Enfin, les composantes $M_i + 1$ à D_i sont celles du vecteur $\mathbf{x}_k$.
La fonction paramétrée, appelée *graph machine*, relative au graphe G_i, est finalement de la forme :

$$g^i_{\boldsymbol{\theta},\boldsymbol{\Theta}} = F_{\boldsymbol{\Theta}}(\mathbf{z}_r)$$

où $\mathbf{z}_r$ est le vecteur des arguments de la fonction associée au nœud racine.
Lorsque de telles fonctions sont construites pour l'ensemble des graphes $\mathcal{G} = \{G_i\}$, les fonctions de nœud $f_{\boldsymbol{\theta}}$ sont identiques pour tous les nœuds d'un même graphe, mais aussi pour tous les graphes. Il est donc nécessaire de s'assurer que les vecteurs $\mathbf{z}$ sont de dimension D fixée quel que soit le graphe considéré, c'est-à-dire $1 + M_i + \left|\mathbf{x}\right| = D$ pour tout *i*.

Il faut donc :

- que le vecteur $\mathbf{x}$ fournisse les mêmes types d'informations pour tous les graphes (et soit ainsi de taille fixe) ;

- que le vecteur **v** soit de dimension M fixe. Il suffit de choisir $M = \max_i M_i$.

IV.3 - Les étiquettes

Le vecteur **x** fournit des informations sur chacun des nœuds du graphe : il correspond aux étiquettes des LRAAMs. Ces informations peuvent être codées de différentes façons. Si la propriété caractérisée par une étiquette est quantitative (si elle mesure par exemple la taille des régions d'une image), il est pertinent d'affecter une seule entrée à cette étiquette, c'est-à-dire une valeur du vecteur **x**, qui varie selon le nœud considéré et la valeur de l'étiquette associée. Au contraire, si cette propriété n'agit pas de façon quantitative sur le problème modélisé, et que l'ensemble des valeurs prises est borné et fini (par exemple les couleurs possibles des régions d'une image), on a affaire à une variable catégorielle : un codage « un parmi *n* » est mieux adapté.

V - L'apprentissage des graph machines

V.1 - Propriété d'approximation universelle

Les réseaux de neurones sont des approximateurs universels, ce qui signifie que toute fonction bornée, suffisamment régulière, peut être approchée dans un domaine fini de l'espace de ses variables, avec une précision arbitraire, par un réseau de neurones possédant une couche de neurones cachés et un neurone de sortie linéaire.
Il a été montré que cette propriété peut être étendue aux réseaux de neurones récursifs [65-67]. Elle est également valable dans le cas des *graph machines*. Celles-ci se composent en effet de deux parties : un codeur, qui fournit une représentation compacte d'un graphe sous une forme vectorielle, et une fonction (par exemple un réseau de neurones) permettant d'effectuer une classification ou une régression à partir de cette représentation vectorielle. Cette fonction possède la propriété d'approximation universelle ; la capacité d'approximation des *graph machine*s dépend donc de celle du codeur.
Considérons une application $F(G)$ de l'ensemble des arbres *n*-aires, dont les étiquettes sont des réels, vers l'ensemble des réels. Soit $\varepsilon > 0$ une précision arbitrairement choisie et $\delta > 0$ une confiance donnée. Il est alors possible de trouver une machine *M* qui possède la propriété suivante :

$$P\left[G \,|\, \left| M(G) - F(G) > \varepsilon \right| \right] < \delta$$

Ce résultat, prouvé dans [65-67], signifie qu'il est possible de trouver une *graph machine M* capable d'approcher la fonction *F*, pour tout arbre *n*-aire, avec une précision arbitraire.
Les *graph machines* sont donc, en théorie, bien adaptées pour établir une relation entre des données structurées et des sorties réelles. Nous allons maintenant montrer comment leur apprentissage est réalisé.

V.2 - Utilisation des algorithmes traditionnels

Nous avons présenté dans le chapitre 1 l'apprentissage traditionnel, pour lequel la base d'exemples permettant d'estimer les paramètres du modèle $g_{\boldsymbol{\theta}}$ est un ensemble de *N* couples entrées / sortie $\left(\left\{\mathbf{x}^i, y^i\right\}, i = 1,...,N\right)$. Le modèle est le même pour tous les exemples, et, lors de l'apprentissage, la fonction de coût minimisée est :

$$J\left(\boldsymbol{\theta}\right) = \sum_{i=1}^{N}\left(y^i - g\left(\mathbf{x}^i, \boldsymbol{\theta}\right)\right)^2 \tag{28}$$

Lors de l'apprentissage des *graph machines*, la base d'apprentissage est constituée de *N* couples structure - sortie $\left(\left\{\mathrm{G}_i, y^i\right\},\ i = 1,...,N\right)$. Il n'y a plus un modèle unique pour tous les exemples : à chaque exemple correspond une fonction particulière $g^i_{\boldsymbol{\theta},\boldsymbol{\Theta}}$, composée à partir des fonctions paramétrées $F_{\boldsymbol{\Theta}}$ (pour le nœud racine) et $f_{\boldsymbol{\theta}}$ (pour les autres nœuds), de façon à refléter la structure de l'exemple *i*. Rappelons que les fonctions $F_{\boldsymbol{\Theta}}$ et $f_{\boldsymbol{\theta}}$ sont *les mêmes* pour tous les exemples.

Il est alors possible de définir une fonction de coût similaire à la fonction de coût des moindres carrés traditionnelle. Cette fonction mesure les écarts entre les observations et les valeurs prédites par le modèle, et peut comporter des termes de régularisation :

$$J(\boldsymbol{\theta},\boldsymbol{\Theta}) = \sum_{i=1}^{N}(y^i - g^i_{\boldsymbol{\theta},\boldsymbol{\Theta}})^2 + \lambda_1\left\|\boldsymbol{\Theta}\right\| + \lambda_2\left\|\boldsymbol{\theta}\right\| \tag{29}$$

où λ_1 et λ_2 sont des constantes de régularisation correctement choisies. La régularisation vise à limiter l'amplitude des paramètres, pour éviter un surajustement du modèle.

La minimisation de la fonction de coût s'effectue de la même manière que lors d'un apprentissage classique, en modifiant les paramètres de façon itérative en fonction de son gradient.

La méthode des poids partagés permet le calcul de celui-ci. Sa *k*-ième composante, c'est-à-dire la dérivée de la fonction de coût par rapport à la composante *k* du vecteur $\boldsymbol{\theta}$, est :

$$\frac{\partial J\left(\boldsymbol{\theta},\boldsymbol{\Theta}\right)}{\partial \theta_k} = \sum_{i=1}^{N}\frac{\partial J^i}{\partial \theta_k} \tag{30}$$

où J^i est la contribution de l'exemple *i* à la fonction de coût.

Le terme $\dfrac{\partial J^i}{\partial \theta_k}$ peut être calculé grâce à la technique des poids partagés. Le jeu de paramètres $\boldsymbol{\theta}$ est en effet le même pour tous les nœuds, donc si le graphe G_i comporte n_i nœuds, le nombre d'occurrences du paramètre θ_k dans le graphe G_i est également n_i. Ce terme peut alors s'exprimer comme la somme des contributions de chacun des nœuds :

$$\frac{\partial J^i}{\partial \theta_k} = \sum_{j=1}^{n_i} \frac{\partial J^i}{\partial \theta_k^j} \tag{31}$$

où θ_k^j désigne le paramètre θ_k qui correspond au nœud j.
Le gradient s'exprime finalement ainsi :

$$\frac{\partial J(\boldsymbol{\theta}, \Theta)}{\partial \theta_k} = \sum_{i=1}^{N} \sum_{j=1}^{n_i} \frac{\partial J^i}{\partial \theta_k^j} \tag{32}$$

Lorsque les fonctions F_{Θ} et $f_{\boldsymbol{\theta}}$ sont des réseaux de neurones, ce gradient peut être calculé par rétropropagation, de la manière usuelle. Dans d'autres cas, il est possible d'avoir recours à des méthodes numériques pour ce calcul.
Les algorithmes traditionnels de descente du gradient, tels que Levenberg-Marquardt, BFGS ou le gradient conjugué, peuvent alors être mis en œuvre pour minimiser la fonction de coût.

V.3 - Sélection de modèle

Les outils que nous avons passés en revue au paragraphe II.3 du chapitre 1 permettent de sélectionner les modèles présentant les meilleures performances de généralisation. Si la validation croisée et le leave-one-out réel peuvent être mis en œuvre de la même manière qu'en modélisation traditionnelle, l'utilisation des leviers et du leave-one-out virtuel n'est pas aussi immédiate : les expressions de la fonction de coût et du gradient ne sont en effet pas les mêmes, et le calcul des leviers n'est pas applicable tel quel. Cette méthode peut cependant être étendue aux *graph machines*, ce que nous allons démontrer dans un premier temps sur un modèle simple à un seul paramètre, puis dans le cas général.

V.3.1 - Les leviers pour les *graph machines*

Considérons donc un modèle avec un seul paramètre θ, pour lequel la fonction associée au nœud racine est identique à la fonction de nœud f_θ. Notons θ_{LS} le paramètre obtenu par apprentissage, y^j la valeur mesurée pour l'exemple j, et $g^j_{\theta_{LS}}$ la prédiction du modèle pour cet exemple. L'erreur de modélisation est donc $R_j = y^j - g^j_{\theta_{LS}}$. De même, si nous notons θ^{-i}_{LS} le paramètre obtenu lorsque la base d'apprentissage ne contient pas l'exemple i, l'erreur de prédiction sur l'exemple j lorsque l'exemple i n'appartient pas à la base d'apprentissage devient $R_j^{-i} = y^j - g^j_{\theta^{-i}_{LS}}$.

Nous cherchons à exprimer l'erreur R_i^{-i} en fonction de l'erreur de modélisation R_i.
Nous obtenons ainsi à partir des deux relations précédentes :

$$R_j^{-i} = R_j + g^j_{\theta_{LS}} - g^j_{\theta^{-i}_{LS}} \tag{33}$$

En supposant que le retrait d'un exemple modifie peu le paramètre θ_{LS}, il est possible d'exprimer $g^j_{\theta^{-j}_{LS}}$ en fonction de $g^j_{\theta_{LS}}$ en effectuant un développement de Taylor du premier ordre de $g^j_{\theta^{-i}_{LS}}$:

$$g^j_{\theta^{-i}_{LS}} = g^j_{\theta_{LS}} + \frac{\partial g^j_{\theta_{LS}}}{\partial \theta}\left(\theta^{-i}_{LS} - \theta_{LS}\right) \tag{34}$$

ce qui, combiné avec la relation (33), donne :

$$R^{-i}_j = R_j - \frac{\partial g^j_{\theta_{LS}}}{\partial \theta}\left(\theta^{-i}_{LS} - \theta_{LS}\right) \tag{35}$$

On peut également effectuer un développement limité de la dérivée du modèle :

$$\frac{\partial g^j_{\theta^{-i}_{LS}}}{\partial \theta} = \frac{\partial g^j_{\theta_{LS}}}{\partial \theta} + \frac{\partial^2 g^j_{\theta_{LS}}}{\partial \theta^2}\left(\theta^{-i}_{LS} - \theta_{LS}\right) \tag{36}$$

La fonction de coût, telle que nous l'avons définie précédemment, s'exprime (sans terme de régularisation), par :

$$J(\theta) = \sum_{j=1}^{N} (y^j - g^j_\theta)^2 \tag{37}$$

Cette fonction est minimum après apprentissage, donc :

$$\frac{\partial J(\theta_{LS})}{\partial \theta} = -2\sum_{j=1}^{N} (y^j - g^j_{\theta_{LS}}) \frac{\partial g^j_{\theta_{LS}}}{\partial \theta} = 0 \tag{38}$$

d'où

$$\sum_{j=1}^{N} R_j \frac{\partial g^j_{\theta_{LS}}}{\partial \theta} = 0 \tag{39}$$

Nous obtenons de manière similaire :

$$\sum_{j \neq i} R^{-i}_j \frac{\partial g^j_{\theta^{-i}_{LS}}}{\partial \theta} = 0\,, \tag{40}$$

ce qui, combiné avec les équations (35) et (36), donne :

$$\sum_j R_j \frac{\partial g^j_{\theta_{LS}}}{\partial \theta} - R_i \frac{\partial g^i_{\theta_{LS}}}{\partial \theta} - \left[\sum_{j \neq i}\left(\frac{\partial g^j_{\theta_{LS}}}{\partial \theta}\right)^2 - \sum_{j \neq i} R_j \frac{\partial^2 g^j_{\theta_{LS}}}{\partial \theta^2}\right]\left(\theta^{-i}_{LS} - \theta_{LS}\right) = 0 \tag{41}$$

Le premier terme correspond à la dérivée de la fonction de coût par rapport à θ, après apprentissage : il est donc nul. Si l'on néglige le terme de second ordre (approximation de Levenberg-Marquardt), on obtient alors :

$$\theta_{LS}^{-i} - \theta_{LS} = -\frac{R_i \dfrac{\partial g^i_{\theta_{LS}}}{\partial \theta}}{\sum_{j \neq i} \left(\dfrac{\partial g^j_{\theta_{LS}}}{\partial \theta} \right)^2} \tag{42}$$

Le développement limité (35) peut alors être appliqué :

$$R_i^{-i} = R_i + \frac{\partial g^i_{\theta_{LS}}}{\partial \theta} \frac{R_i \dfrac{\partial g^i_{\theta_{LS}}}{\partial \theta}}{\sum_{j \neq i} \left(\dfrac{\partial g^j_{\theta_{LS}}}{\partial \theta} \right)^2} \tag{43}$$

Ce qui conduit à : $$R_i^{-i} = \frac{R_i}{1 - h_{ii}} \tag{44}$$

avec : $$h_{ii} = \frac{\left(\dfrac{\partial g^i_{\theta_{LS}}}{\partial \theta} \right)^2}{\sum_{j} \left(\dfrac{\partial g^j_{\theta_{LS}}}{\partial \theta} \right)^2} \tag{45}$$

Cette expression est semblable à celle obtenue dans le cadre de la modélisation classique, et le terme h_{ii} est équivalent aux leviers traditionnels.

Nous considérons maintenant un modèle M établi à partir de fonctions de nœuds $f_{\boldsymbol{\theta}}(\mathbf{z})$. Le vecteur de paramètres $\boldsymbol{\theta}$ est de dimension q, et nous notons comme précédemment $\boldsymbol{\theta}_{\mathrm{LS}}$ le vecteur obtenu par apprentissage sur la base des N exemples. M^{-i} est le modèle de vecteur de paramètres $\boldsymbol{\theta}_{\mathrm{LS}}^{-i}$ obtenu par apprentissage sur la base privée de l'exemple i.

Un développement limité du modèle au second ordre autour d'un vecteur de paramètres $\boldsymbol{\theta}^*$ donne :

$$\mathbf{g}_{\boldsymbol{\theta}} = \mathbf{g}_{\boldsymbol{\theta}^*} + \mathbf{Z}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^*\right) + {}^T\left(\boldsymbol{\theta} - \boldsymbol{\theta}^*\right)\mathbf{S}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^*\right) \tag{46}$$

où $\mathbf{Z}$ est la matrice jacobienne du modèle, de taille (N, q), en $\boldsymbol{\theta}^*$, définie par $\left(\mathbf{Z}\right)_{jk} = \left(\dfrac{\partial g^j_{\boldsymbol{\theta}^*}}{\partial \theta_k} \right)$.

Cette matrice est semblable à la jacobienne intervenant en modélisation classique : son terme (j, k) représente la dérivée première de la fonction associée au graphe j par rapport au k-ième paramètre.

$\mathbf{S}$ est un tenseur d'ordre trois, et s'exprime par $\mathbf{S} = \sum_{i=1}^{N} \mathbf{S}_i(\mathbf{e}^i)$, où $\mathbf{S}_i$ est une matrice (q, q) définie par $\mathbf{S}_i = \left(\left. \frac{\partial^2 g^i_{\boldsymbol{\theta}}}{\partial \theta_k \partial \theta_m} \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} \right)_{\substack{k=1,\ldots,q \\ m=1,\ldots,q}}$ et $\mathbf{e}^i$ est le i-ième vecteur de la base orthonormée de $\mathbb{R}^N$.

En utilisant la relation (46) dans la fonction de coût, on obtient, après différentiation et en négligeant les termes d'ordre supérieur à 1 en $\left(\boldsymbol{\theta} - \boldsymbol{\theta}^*\right)$:

$$\frac{\partial J}{\partial \boldsymbol{\theta}} = \frac{\partial J}{\partial \left(\boldsymbol{\theta} - \boldsymbol{\theta} *\right)} \cong -2\,{}^T\mathbf{Z}\left(\mathbf{y} - \mathbf{g}_{\boldsymbol{\theta}^*}\right) + \left\{ 2\,{}^T\mathbf{Z}\mathbf{Z} - 2\sum_{j=1}^{N}\left(y^i - g^i_{\boldsymbol{\theta}^*}\right)\mathbf{S}_j \right\}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^*\right) \tag{47}$$

Le terme entre accolades, de taille (q, q), est la hessienne de la fonction de coût : $\mathbf{H} = \left(\frac{\partial^2 J_{\boldsymbol{\theta}^*}}{\partial \theta_k \partial \theta_m} \right)_{\substack{k=1,\ldots,q \\ m=1,\ldots,q}}$. Le second terme de la matrice hessienne peut être négligé (approximation de Levenberg-Marquardt), et $\mathbf{H} \cong 2\,{}^T\mathbf{Z}\mathbf{Z}$.

La fonction de coût est minimum en $\boldsymbol{\theta}_{\mathrm{LS}}$, d'où :

$$\boldsymbol{\theta}_{\mathrm{LS}} \cong \boldsymbol{\theta}^* + \left({}^T\mathbf{Z}\mathbf{Z}\right)^{-1}\,{}^T\mathbf{Z}\left(\mathbf{y} - \mathbf{g}_{\boldsymbol{\theta}^*}\right) \tag{48}$$

Dans le cas d'un modèle linéaire, cette relation est exacte.

Si l'on suppose que le retrait d'un exemple modifie peu le vecteur de paramètres $\boldsymbol{\theta}_{\mathrm{LS}}$, il est possible d'obtenir un développement similaire de $\boldsymbol{\theta}^{-i}_{\mathrm{LS}}$ au voisinage de $\boldsymbol{\theta}^*$:

$$\boldsymbol{\theta}^{-i}_{\mathrm{LS}} \cong \boldsymbol{\theta}^* + \left({}^T\mathbf{Z}^{(-i)}\mathbf{Z}^{(-i)}\right)^{-1}\,{}^T\mathbf{Z}^{(-i)}\left(\mathbf{y}^{(-i)} - \mathbf{g}^{-i}_{\boldsymbol{\theta}^*}\right) \tag{49}$$

où $\mathbf{Z}^{(-i)}$ est une matrice de taille (N-1, q), $\mathbf{y}^{(-i)}$ et $\mathbf{g}^{-i}_{\boldsymbol{\theta}^*}$ des vecteurs de taille (N-1).

Les équations (48) et (49) permettent alors d'obtenir :

$$\boldsymbol{\theta}^{-i}_{\mathrm{LS}} \cong \boldsymbol{\theta}_{\mathrm{LS}} - \left({}^T\mathbf{Z}\mathbf{Z}\right)^{-1}\mathbf{z}^i \frac{R_i}{1 - h_{ii}} \tag{50}$$

où $\mathbf{z}^i$ est le i-ième vecteur colonne de ${}^T\mathbf{Z}$ et $h_{ii} = {}^T\mathbf{z}^i\left({}^T\mathbf{Z}\mathbf{Z}\right)^{-1}\mathbf{z}^i$ est le levier de l'exemple i.

L'erreur de modélisation du modèle M pour l'exemple i est $R_i = y^i_p - g^i_{\boldsymbol{\theta}_{\mathrm{LS}}}$, celle du modèle M^{-i} est $R^{-i}_i = y^i_p - g^i_{\boldsymbol{\theta}^{-i}}$, d'où :

$$R^{-i}_i = R_i + g^i_{\boldsymbol{\theta}_{LS}} - g^i_{\boldsymbol{\theta}^{-i}_{LS}} \cong {}^T\mathbf{z}^i\left(\boldsymbol{\theta}_{LS} - \boldsymbol{\theta}^{-i}_{LS}\right) \tag{51}$$

En utilisant la relation (50), on obtient finalement une estimation de l'erreur R^{-i}_i en fonction

de R_i :

$$R_i^{-i} \cong \frac{R_i}{1-h_{ii}} \tag{52}$$

Cette expression est identique à celle obtenue en modélisation classique, et les leviers h_{ii} présentent les mêmes propriétés : ils sont tous compris entre 0 et 1, et vérifient la relation $\sum_{i=1}^{N} h_{ii} = q$. Les leviers mesurent ainsi également l'influence de chaque exemple sur le modèle, et permettent de détecter les exemples de trop forte influence (données erronées ou type d'exemple sous-représenté dans la base d'apprentissage) et ceux qui n'influent quasiment pas sur le modèle, et sont donc inutiles.

Cette relation permet également de remplacer le score de leave-one-out E_t, défini par :

$$E_t = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(R_i^{-i}\right)^2}$$

par le score de leave-one-out virtuel E_p :

$$E_p = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{R_i}{1-h_{ii}}\right)^2}$$

qui est une très bonne estimation de l'erreur de généralisation du modèle.

V.3.2 - Estimation des intervalles de confiance

La donnée des leviers permet le calcul approché des intervalles de confiance pour des modèles non-linéaires. Ce calcul, présenté dans le paragraphe II.3.4.2 du chapitre 1, peut être étendu aux *graph machines*. L'espérance mathématique de la sortie Y_p, pour le graphe G^i, est donc donnée, avec une confiance 1-α, par :

$$E\left(Y_p \mid G^i\right) = g^i_{\Theta^*,\theta^*} \pm t_\alpha^{N-q} s\sqrt{h_{ii}}$$

où g est le modèle considéré, de paramètres Θ^* et θ^*, t_α^{N-q} est la valeur d'une variable de Student à N-q degrés de liberté et un niveau de confiance 1-α et s est une estimation de la variance de l'erreur de prédiction du modèle.

V.3.3 - Illustration de la pertinence des leviers

Nous avons montré que les leviers peuvent être étendus aux *graph machines*. Nous allons maintenant l'illustrer, en présentant d'abord une comparaison entre les erreurs de leave-one-out réel et celles calculées par l'intermédiaire des leviers sur une base de données structurées, puis en montrant sur un exemple comment les leviers permettent d'estimer l'influence des exemples d'apprentissage sur le modèle.

V.3.3.1 - Comparaison leave-one-out réel / virtuel

Cette comparaison est illustrée sur les Figures 14 et 15, pour lesquelles la base d'étude est constituée de 330 molécules, auxquelles sont associées leurs températures d'ébullition. Nous verrons dans le paragraphe I du chapitre 3 comment des molécules peuvent être représentées sous forme de graphes, ce qui permet l'utilisation des *graph machines* pour modéliser et prédire leurs propriétés.

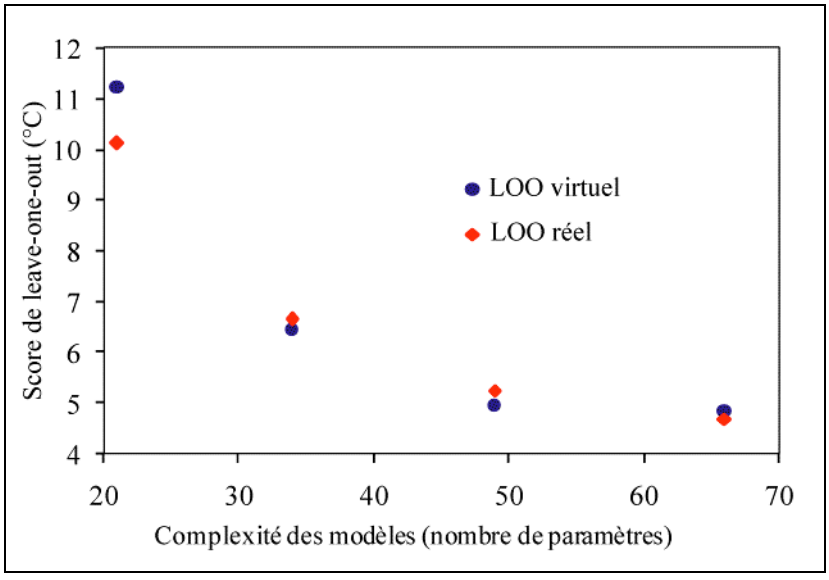


Figure 14 : Comparaison des scores de LOO réel et virtuel avec des modèles de diverses complexités, pour la modélisation de la température d'ébullition de 330 molécules

Les scores de leave-one-out réel et virtuel sont très proches (erreurs respectives de 10 %, 3 %, 6 % et 3 %), et le score de leave-one-out virtuel semble constituer effectivement une très bonne approximation de l'erreur de généralisation. Plus globalement, le calcul du score de leave-one-out virtuel indique sur cet exemple que l'erreur de généralisation est la plus faible pour un modèle à 66 paramètres (ce qui correspond à un réseau de neurones à 5 neurones cachés), comme l'aurait fait celui du score de leave-one-out réel, mais en nécessitant un temps de calcul 330 fois plus court.
La Figure 15 illustre de façon plus précise la comparaison entre les erreurs commises par le modèle sur les exemples lorsqu'ils ne font pas partie de la base d'apprentissage et les estimations de ces erreurs obtenues par l'équation (52).

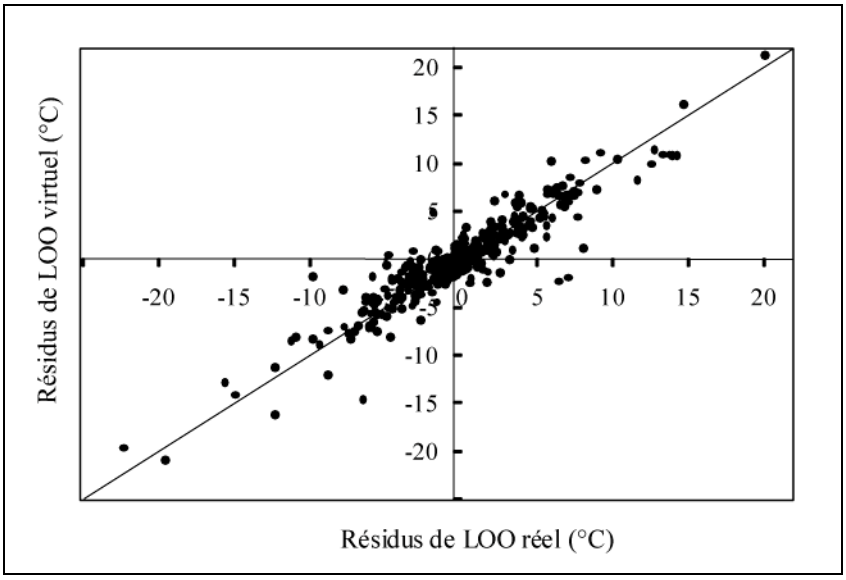


Figure 15 : Comparaison des résidus de LOO réel et virtuel pour la modélisation par des *graph machines* de la température d'ébullition de molécules

On constate que les valeurs des résidus évaluées par la méthode du leave-one-out virtuel sont globalement très proches des valeurs réelles de ces résidus, même lorsque celles-ci sont importantes. Nous avons ainsi, dans la plupart des modélisations effectuées, privilégié la méthode du leave-one-out virtuel par rapport au leave-one-out réel.

VI - Modélisation à partir de graphes cycliques

La démarche de modélisation à partir de graphes acycliques que nous avons présentée peut être étendue aux cas où les graphes comportent un ou plusieurs cycles. Nous avons pour cela choisi une approche qui consiste à transformer les graphes cycliques en graphes acycliques tout en tenant compte de la présence des cycles du graphe initial. Cette méthode nous permet ainsi de modéliser indifféremment des structures cycliques ou acycliques. Nous présenterons également dans cette section des alternatives à cette approche, que nous n'avons pas retenues pour des raisons que nous expliquerons.

VI.1 - **Transformation de graphes quelconques en arborescences**

Pour que la structure d'un graphe connexe quelconque puisse être exploitée par la méthode présentée, il est nécessaire de transformer celui-ci en arborescence, donc de rendre ce graphe acyclique et de choisir un nœud racine, ce qui oriente implicitement l'arborescence obtenue. Nous avons décidé d'effectuer cette transformation en appliquant un algorithme que nous allons décrire. Le choix de cet algorithme est justifié dans la section I.3 du chapitre 3. La première étape consiste à sélectionner, parmi les nœuds du graphe, celui qui sera le nœud racine de l'arborescence, grâce à un premier algorithme qui numérote les nœuds de façon canonique et unique, selon des critères tels que leurs distances aux autres nœuds du graphe et leur degré. Le nœud choisi comme racine du futur arbre est celui qui porte le numéro 1 ; il est un nœud central du graphe. D'autres critères peuvent être pris en considération, en fonction du type de structures étudiées, et des informations fournies par les étiquettes des nœuds. Par exemple, lorsque les graphes représentent des images, et les nœuds des régions de ces images, il peut être nécessaire d'ajouter un critère pour tenir compte de la taille de ces régions, fournie par les étiquettes, lors de la numérotation canonique des nœuds. Les nœuds des graphes associés à des molécules chimiques sont numérotés suivant les critères détaillés dans [68] et dans le paragraphe I.3 du chapitre 3. Nous décrivons plus en détail cet algorithme dans l'annexe 1. Dans une seconde étape, les graphes sont transformés en graphes acycliques si nécessaire. Il faut pour cela supprimer une arête pour chaque cycle du graphe, tout en veillant à ce que celui-ci reste connexe. Un second algorithme effectue le choix de ces arêtes à supprimer, en sélectionnant les plus distantes du nœud racine. L'arborescence alors obtenue est orientée du nœud central vers les extrémités du graphe.

Une particularité importante des *graph machines* est la façon de gérer les cycles : bien que certaines arêtes soient supprimées pour rendre les graphes acycliques, l'information correspondant à leur présence ne l'est pas. Elle est en effet implicitement conservée, par l'intermédiaire des étiquettes, qui fournissent le degré, dans le graphe initial, des deux nœuds reliés par l'arête supprimée. Considérons ainsi les graphes de la Figure 16.

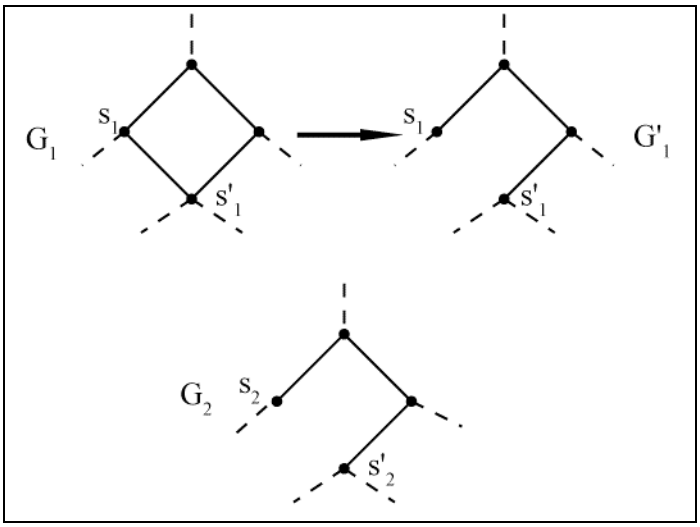


Figure 16 : Graphes cycliques et acycliques différenciés par leurs degrés

Le graphe G_1 est un graphe cyclique, qui après ouverture devient le graphe G'_1. Le graphe G_2 est acyclique, et semble identique à G'_1, mais il diffère de celui-ci par ses étiquettes : tandis que les nœuds s_1 et s'_1 sont respectivement de degrés d_1 et d'_1, les nœuds s_2 et s'_2 sont de degrés $d_1 - 1$ et $d'_1 - 1$. Ainsi, un graphe acyclique après suppression d'un ou plusieurs cycles n'est pas identique à un graphe acyclique de même structure, mais n'ayant pas subi de transformation, grâce à l'information contenue dans le degré des nœuds, et portée par les étiquettes.

La Figure 17 illustre comment le graphe G_3, qui comporte quatre cycles, est transformé en arborescence. Les numéros affectés aux nœuds lors de la numérotation canonique figurent près des nœuds du graphe G_3. Le nœud 1 est le centre du graphe : c'est le nœud pour lequel la distance maximale aux autres nœuds est la plus faible, et de plus fort degré. Il est alors choisi comme nœud racine. Le graphe G_3 comporte 4 cycles : il faut donc supprimer 4 arêtes. Le cycle formé par les nœuds 1, 2 et 4 constitue un exemple simple : l'arête 2-4 étant la plus éloignée du nœud racine, c'est celle qui est supprimée. Les autres cycles sont ouverts de la même manière, et l'arborescence obtenue, orientée du nœud racine aux feuilles, est le graphe G'_3. Les nœuds initialement reliés dans le graphe G_3 conservent leur degré, malgré la suppression des arêtes : ces degrés sont indiqués sur le graphe G'_3 par les chiffres en gras et en italique à gauche des nœuds.

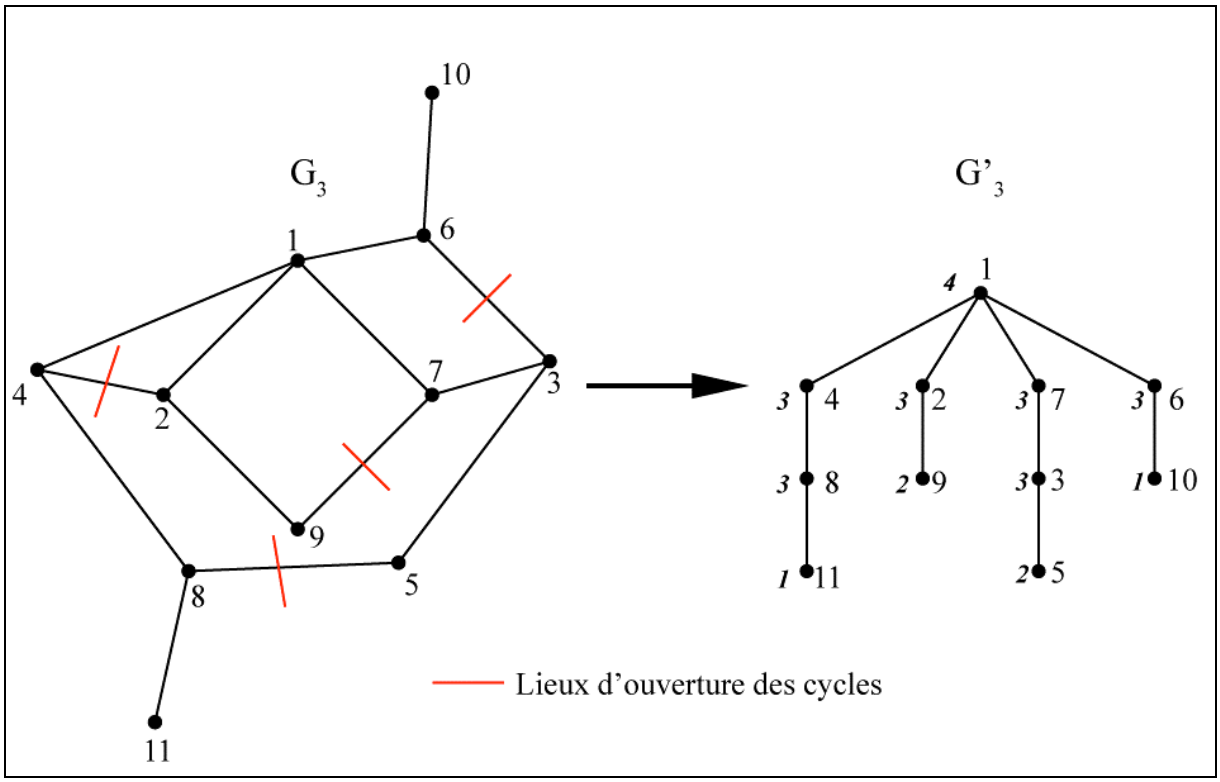


Figure 17 : Transformation d'un graphe comportant plusieurs cycles en graphe acyclique

Les graphes cycliques peuvent alors être codés par les *graph machines* de la même manière que les graphes acycliques.

VI.2 - Méthode alternative de modélisation à partir de graphes cycliques

Il est également possible de modéliser les graphes cycliques sans les rendre acycliques. La construction des *graph machines* associées nécessiterait, de la même façon que pour les graphes acycliques, de choisir un nœud racine et d'orienter le graphe. Considérons par exemple les graphes de la Figure 18, où G_4 est la structure d'origine et G'_4 le graphe cyclique orienté correspondant, dont le nœud 1 est la racine.

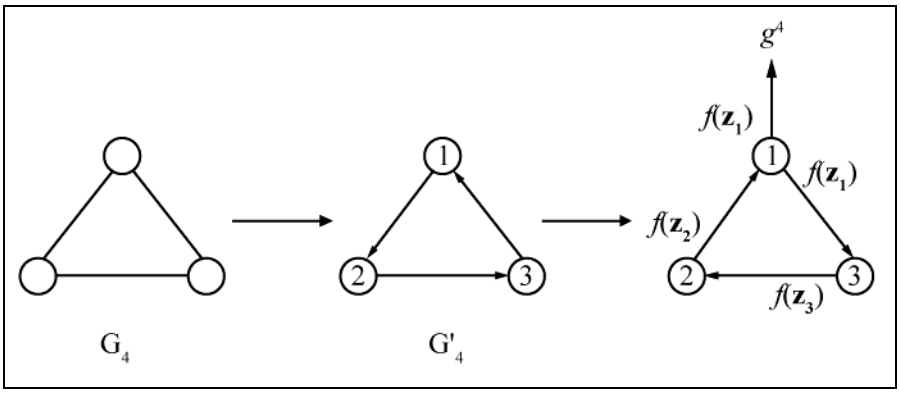


Figure 18 : Alternative pour modéliser un graphe cyclique - graphe cyclique simple

La méthode utilisée dans le cas de graphes acycliques, et appliquée au graphe G'_4, donne alors la relation :

$$g^4 = \underline{f(\mathbf{z}_1)} = f\left(f\left(\mathbf{z}_2\right),\mathbf{x}_1\right) = f\left(f\left(f\left(\mathbf{z}_3\right),\mathbf{x}_2\right),\mathbf{x}_1\right) = f\left(f\left(f\left(\underline{f\left(\mathbf{z}_1\right)},\mathbf{x}_3\right),\mathbf{x}_2\right),\mathbf{x}_1\right) \qquad (53)$$

Cette relation ne peut être vérifiée que pour une certaine forme de fonction f donnée, qu'il est possible de déterminer en résolvant l'équation récursive (53).
Un exemple de structure plus complexe est présenté sur la Figure 19.

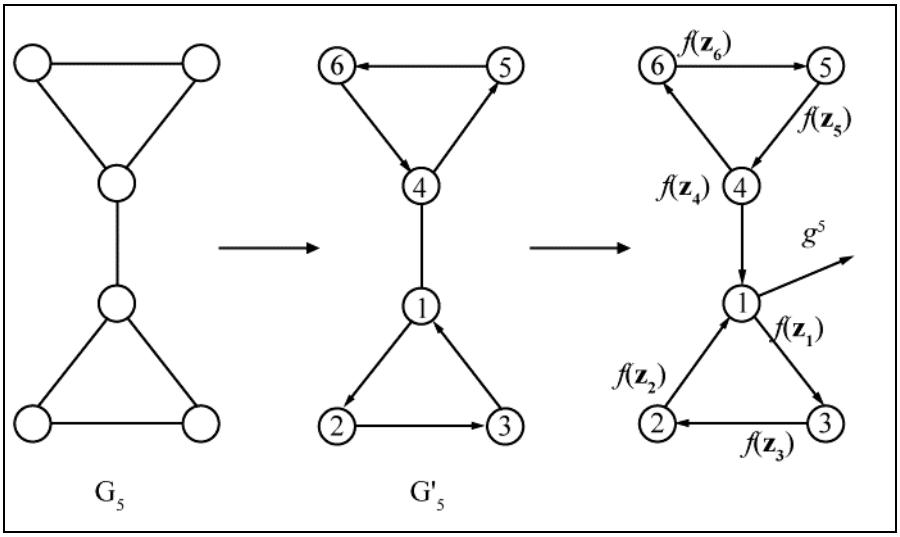


Figure 19 : Alternative pour modéliser un graphe comportant deux cycles

Nous voyons que dans ce cas, f doit satisfaire deux équations récursives différentes. Il n'est donc plus possible d'utiliser une fonction identique pour tous les nœuds.
L'étude d'une base constituée de N structures nécessite ainsi la résolution de N équations récursives, et conduit à N différentes fonctions de nœud racine F^i, $i \in [\![1, N]\!]$. Les inconvénients de cette méthode sont donc multiples :

- La résolution des équations récursives, qui peuvent devenir très complexes pour de grandes structures, augmente le temps de calcul.
- Nous ne pouvons plus utiliser les poids partagés et ne disposons pas d'un outil de calcul nous permettant, par apprentissage, de déterminer les paramètres de différentes fonctions f.
- La fonction de nœud n'étant plus unique, nous ne pouvons plus utiliser les leviers ni tous les outils qui s'y rapportent.

Cette approche semble donc moins efficace que celle retenue pour la modélisation des graphes cycliques.

VII- Exemples didactiques d'utilisation des *graph machines*

Lorsque le problème de modélisation est très simple, il est possible de le résoudre sans avoir recours à l'apprentissage. Nous allons ainsi illustrer le fonctionnement des *graph machines* par deux exemples didactiques : le calcul du nombre de nœuds puis du nombre d'arêtes d'un graphe. Pour ces deux problèmes, la fonction de nœud f, ainsi que les paramètres $\boldsymbol{\theta}$, peuvent être calculés simplement.

VII.1- Détermination du nombre de nœuds d'un graphe

Une analyse simple conduit à choisir une fonction de nœud linéaire, puisque la grandeur à modéliser n'est autre que le nombre de fonctions de nœud. Nous optons donc pour une fonction de nœud du type :

$$f_{\boldsymbol{\theta}}(\mathbf{z}) = \theta_0 + \sum_{i=1}^{M} \theta_i v_i + \sum_{j=1}^{|\mathbf{x}|} \theta_j x_j$$

où **v** et **x** sont les vecteurs définis dans la section IV.2 de ce chapitre. La fonction associée au nœud racine est identique. Nous appellerons la fonction recherchée f^1.

Pour que la fonction $g_{\boldsymbol{\theta}}$ associée à un graphe compte effectivement le nombre de nœuds de celui-ci, il suffit que, lors de la propagation des feuilles à la racine de l'arborescence, chaque nœud incrémente la sortie de 1. La valeur de chaque fonction de nœud doit donc être égale à la somme des sorties des fonctions des nœuds enfants du nœud (c'est-à-dire la somme des v_i) plus 1. De plus, les caractéristiques particulières de chaque nœud ne jouent aucun rôle : il

n'est donc pas nécessaire de prendre en considération leurs éventuelles étiquettes (telles que leurs degrés).
La fonction de nœud est donc :

$$f_{\boldsymbol{\theta}}^{1}(\mathbf{z}) = 1 + \sum_{i=1}^{M} v_i \tag{54}$$

La Figure 20 illustre comment cette fonction de nœud permet effectivement de recenser les nœuds d'un graphe.

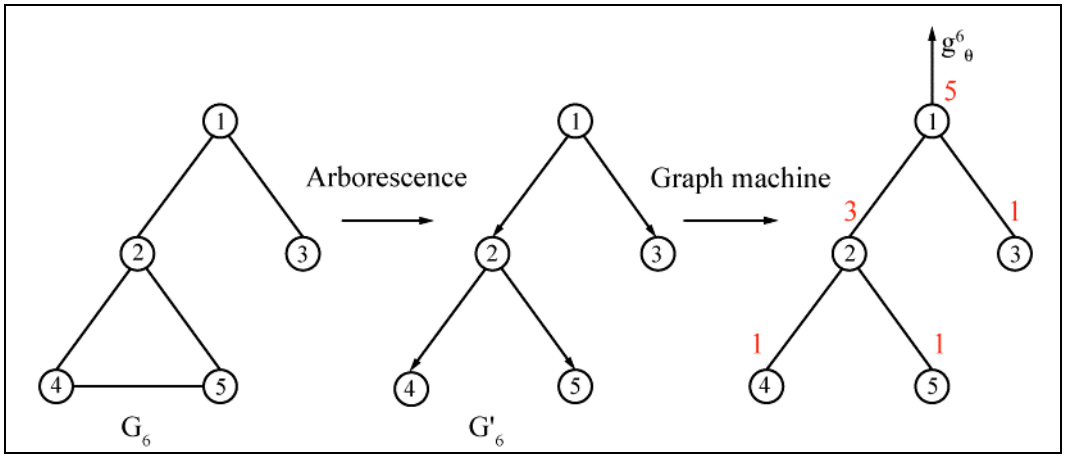


Figure 20 : Exemple de calcul du nombre de nœuds d'un graphe par les *graph machines*

- Les nœuds 3, 4 et 5 n'ont pas d'enfant, donc **v** est le vecteur nul, et $f_{\boldsymbol{\theta}}(\mathbf{z}_3) = f_{\boldsymbol{\theta}}(\mathbf{z}_4) = f_{\boldsymbol{\theta}}(\mathbf{z}_5) = 1$.
- Le nœud 2 a pour enfants les nœuds 4 et 5, donc pour ce nœud $v_1 = v_2 = 1$, et $f_{\boldsymbol{\theta}}(\mathbf{z}_2) = 3$.
- Enfin, les enfants du nœud 1 sont les nœuds 2 et 3, on a donc $v_1 = 3,\ v_2 = 1$ et $g^6_{\boldsymbol{\theta}} = f_{\boldsymbol{\theta}}(\mathbf{z}_1) = 1 + 3 + 1 = 5$. La valeur de la fonction $g^6_{\boldsymbol{\theta}}$ est bien le nombre de nœuds du graphe G_6.

VII.2 - Détermination du nombre d'arêtes et de cycles d'un graphe

Ce problème est plus complexe que le précédent, car certaines arêtes peuvent être supprimées lors de la conversion des graphes d'origine en arborescences. Comme nous l'avons suggéré dans le paragraphe VI.1 de ce chapitre, ce sont les degrés des nœuds, accessibles grâce à leurs étiquettes, qui permettent de tenir compte de toutes les arêtes d'origine. Or, le nombre d'arêtes vérifie la relation $N_{arêtes} = \frac{1}{2} \sum_{k=1}^{Nnoeuds} d_k$, où d_k est le degré du nœud k du graphe. En effet, chaque arête relie deux nœuds, et incrémente chacun de leur degré de 1, donc la somme des degrés de 2.

Ce problème est également linéaire, et nous choisissons à nouveau une fonction de la forme :

$$f_{\boldsymbol{\theta}}(\mathbf{z}) = \theta_0 + \sum_{r=1}^{M} \theta_r v_r + \sum_{s=1}^{|\mathbf{x}|} \theta_s x_s$$

Notons f^2 la fonction cherchée. Lors de la propagation, il faut que chaque fonction de nœud somme les sorties de ses enfants et incrémente cette somme de la moitié de son degré, soit :

$$f^2_{\boldsymbol{\theta}}(\mathbf{v}_k, d_k) = \sum_{r=1}^{M} v_r + \frac{1}{2} d_k \tag{55}$$

Il est alors possible de calculer le nombre d'arêtes de graphes connexes quelconques. Un exemple de ce calcul est donné sur la Figure 21.

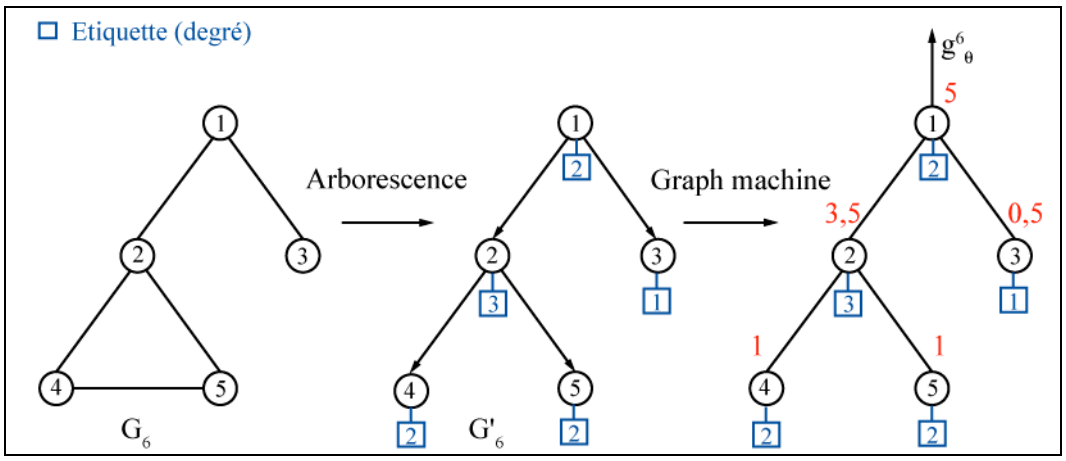


Figure 21 : Exemple de calcul du nombre d'arêtes d'un graphe par les *graph machines*

Ce problème, malgré sa relative simplicité, illustre comment les *graph machines* sont capables de tenir compte des arêtes supprimées lorsque les cycles sont ouverts.
Cette propriété des *graph machines* est davantage mise en évidence lors de la modélisation du nombre de cycles de graphes, problème qu'il est également possible de résoudre sans avoir recours à l'apprentissage. Le nombre de cycles indépendants d'un graphe connexe (également appelé nombre cyclomatique) vérifie en effet la relation :

$$N_{cycles} = N_{arêtes} - N_{noeuds} + 1 \tag{56}$$

Il faut donc que la contribution de chaque nœud soit égale au nombre d'arêtes qu'il engendre diminué de 1 (pour 1 nœud), et que ces contributions se propagent le long de la *graph machine* associée.
Il suffit alors, pour obtenir la fonction de nœud f^3 permettant de calculer le nombre de cycles, d'effectuer une combinaison des fonctions (54) et (55) :

$$f^3_{\boldsymbol{\theta}}(\mathbf{z}) = \underbrace{\sum_{r=1}^{M} v_r}_{\text{Propagation}} + \underbrace{\sum_{r=1}^{M} v_r + \frac{1}{2} d_k}_{\text{Narêtes}} - \underbrace{\left(1 + \sum_{r=1}^{M} v_r\right)}_{\text{Nnoeuds}} = -1 + \sum_{r=1}^{M} v_r + \frac{1}{2} d_k$$

La fonction associée au nœud central doit être différente pour tenir compte du terme (+1) de la relation (56) :

$$F_{\theta}^{3}(\mathbf{z}) = f_{\theta}^{3}(\mathbf{z}) + 1 = \sum_{r=1}^{M} v_r + \frac{1}{2} d_k$$

Un exemple de calcul du nombre de cycles d'un graphe est donné sur la Figure 22.

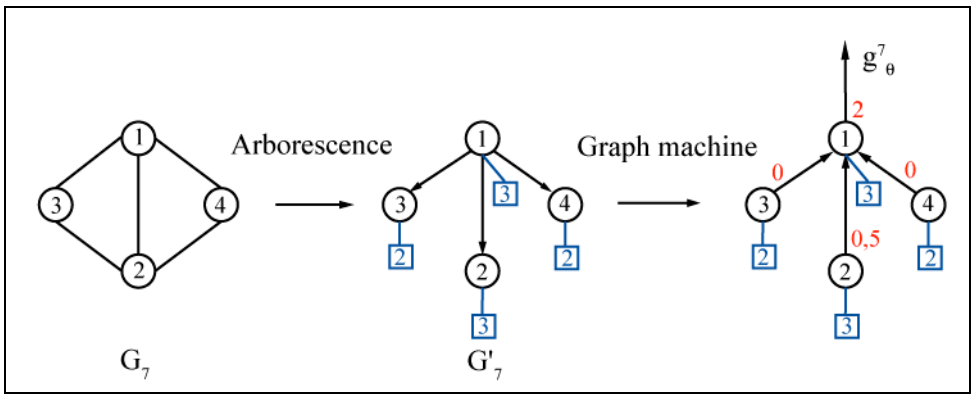


Figure 22 : Exemple de calcul du nombre de cycles d'un graphe par les *graph machines*

VIII - Résumé : méthodologie de conception de modèles prédictifs ou de classifieurs par apprentissage à partir d'une base de données structurées

Les problèmes décrits dans la section précédente sont des exemples didactiques, pour lesquels la fonction de nœud adéquate et ses paramètres peuvent être trouvés pratiquement sans calcul, et surtout sans apprentissage. Lorsque le problème de modélisation est plus complexe, il peut être nécessaire de choisir une fonction de nœud non-linéaire, dont les paramètres doivent être déterminés par apprentissage, sur une base d'exemples de taille suffisante. La démarche de modélisation, étant donnée une base d'apprentissage de N exemples, peut alors être la suivante :

- les structures sont mises sous la forme de matrices d'adjacence, si elles ne le sont pas déjà ; des lignes supplémentaires peuvent être ajoutées aux matrices pour renseigner les étiquettes des nœuds ;
- ces structures sont converties en arborescences par un algorithme convenable ;
- plusieurs fonctions de nœud de complexités diverses (par exemple des réseaux de neurones avec un nombre de neurones cachés variable) sont envisagées, et les *graph machines* correspondantes construites ;
- les paramètres partagés θ et Θ sont estimés par apprentissage à partir des N fonctions $g^i_{\theta,\Theta}$ et des valeurs expérimentales de la propriété étudiée ;

- les prédictions, les leviers de chaque exemple, les intervalles de confiance et les scores de leave-one-out virtuel sont calculés pour chacun des modèles envisagés, et le modèle le plus performant peut être sélectionné.

CHAPITRE 3
Méthodologie en QSPR et QSAR

Les *graph machines*, qui permettent d'effectuer une régression ou une classification directement à partir de structures, sont tout particulièrement adaptées lorsque la conversion des structures en vecteurs de données conduit à une perte d'information, ou est elle-même une opération délicate. Cette conversion nécessite en effet de sélectionner les variables les plus pertinentes pour rendre compte du phénomène à modéliser, et de les calculer, de les mesurer ou même d'en estimer la valeur si le calcul et la mesure ne sont pas envisageables. La qualité du modèle peut donc être affectée de multiples façons. Il est parfois difficile de trouver des variables pertinentes ou de les sélectionner lorsqu'elles sont trop nombreuses. De plus, la mesure ou l'approximation de ces variables (par une modélisation préalable par exemple) introduit une erreur dans le modèle. Enfin un modèle dont les variables sont elles-mêmes des résultats d'expériences est peu intéressant lorsque la modélisation a pour but de remplacer des résultats expérimentaux trop complexes ou trop longs à obtenir.
Ces problèmes apparaissent en particulier lors de la modélisation de propriétés ou d'activités de molécules, et nous allons voir dans cette partie comment la méthode des *graph machines* permet de s'en affranchir.

I - Construction des graph machines associées aux molécules

Les molécules peuvent être considérées comme des objets structurés. Les méthodes relatives aux *graph machines* sont alors applicables aux problèmes de modélisation d'activités ou de propriétés moléculaires (QSAR/QSPR, ou Quantitative Structure-Activity/Property Relationship). Nous étudierons dans un premier temps comment les molécules peuvent être représentées par des graphes, et comment l'algorithme qui transforme ces graphes en arborescences est choisi. Nous montrerons ensuite qu'il est possible, si le problème de modélisation le nécessite, d'ajouter des informations supplémentaires via les étiquettes des nœuds. Ces étiquettes permettent de modéliser des propriétés à partir de la seule donnée des structures des molécules et d'informations simples telles que le degré des nœuds, alors que d'autres méthodes nécessitent le calcul de nombreux descripteurs.

I.1 - Représentation de molécules par des arborescences

De nombreuses représentations existent pour les molécules, et les moyens d'identification sont multiples : elles peuvent par exemple être définies par un nom, un numéro d'identification, ou la donnée des coordonnées de leurs atomes dans l'espace. Le code SMILES (pour Simplified Molecular Input Line Entry Specification) [69] est un langage formel qui décrit les molécules et les réactions chimiques en représentant les atomes et les liaisons entre ceux-ci par des caractères ASCII ; la structure en deux ou en trois dimensions de la molécule peut être retrouvée à partir de ce code. Ce type de représentation présente l'avantage d'être très compact. De plus, il existe une forme canonique de ce code, qui constitue alors un identifiant universel pour une structure chimique [70]. La Figure 23.a illustre trois représentations possibles d'une molécule : son dessin, son nom ainsi que son code SMILES.

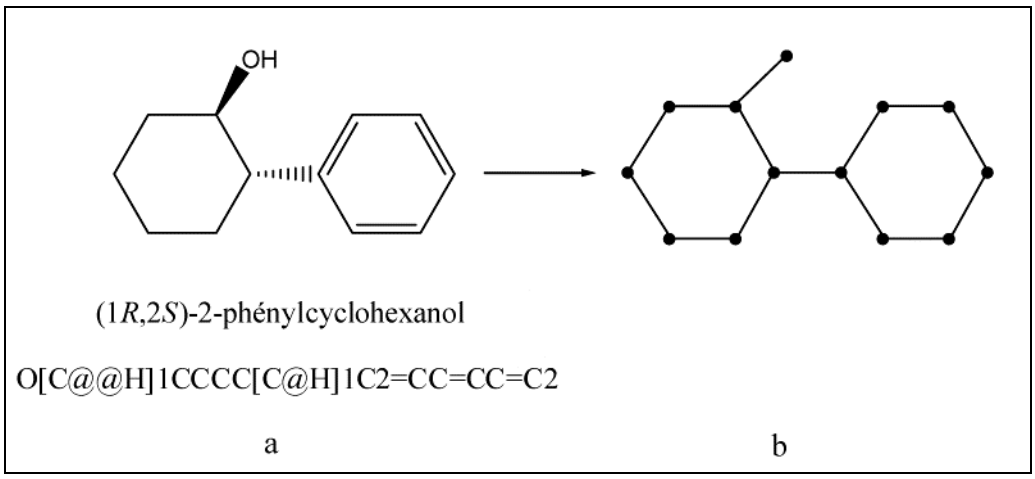


Figure 23 : a) Plusieurs représentations d'une molécule - b) Représentation par un graphe

Les molécules sont constituées d'atomes et de liaisons entre ces atomes, et peuvent ainsi être considérées comme des données structurées, donc représentées par des graphes. Ces représentations peuvent être obtenues de la façon suivante :

- chaque atome non-hydrogène est associé à un nœud
- chaque liaison entre deux atomes, qu'elle soit simple, multiple ou aromatique, est associée à une arête entre les deux nœuds correspondants.

La Figure 23.b illustre ainsi comment la molécule de la Figure 23.a peut être représentée par un graphe. Nous allons voir qu'il est également possible de caractériser les atomes grâce à des étiquettes portées par les nœuds associés.

I.2 - Étiquettes (nature, degré, isomérie, éventuel descripteur)

Les étiquettes, qui portent des informations relatives aux nœuds, permettent de tenir compte des caractéristiques de ces nœuds. Lorsque les graphes représentent des molécules, il s'agit des caractéristiques des atomes, qui peuvent être qualitatives (nature de l'atome, agencement des voisins autour de cet atome) ou quantitatives.

– La **nature chimique** d'un atome (carbone, oxygène, soufre…) a généralement une influence sur les propriétés de la molécule à laquelle il appartient : elle doit donc être prise en considération. La distinction entre les atomes de nature différente peut être réalisée grâce aux étiquettes ; puisqu'il s'agit d'une variable catégorielle, nous avons adopté le codage « 1 parmi *n* », où *n* est le nombre de types d'atomes possibles.

– Le **degré** d'un atome, c'est-à-dire le nombre de doublets liants qu'il partage avec d'autres atomes, est également une information dont il faut tenir compte lorsque des liaisons multiples ou aromatiques, ou des cycles, interviennent. Nous avons montré dans la section VI du chapitre 2 que l'information relative à la présence des cycles, lors de l'ouverture de ceux-ci, est conservée grâce aux degrés des nœuds. Le degré peut également rendre compte des liaisons multiples ou aromatiques. En effet, lors de la conversion des molécules en graphes, les liaisons, quelle que soit leur nature, sont toutes représentées par une arête unique, et les particularités des différentes liaisons disparaissent. En revanche, ces différences peuvent être préservées grâce au degré, qui renseigne sur le nombre et la nature des liaisons formées par un atome dans la molécule d'origine. Par exemple, lorsqu'un atome forme une liaison simple, son degré augmente de 1 ; si c'est une liaison double ou aromatique, il augmente de 2. Le degré est également relié au nombre d'atomes d'hydrogène porté par un atome. Le degré *d* d'un atome (neutre) est égal à :

$$d = V - N_H$$

où V désigne la valence de l'atome, et dépend de celui-ci (elle vaut 4 pour un atome de carbone, ce qui est généralement la valeur maximale pour les molécules considérées ici), et N_H correspond au nombre de liaisons avec un hydrogène.

La Figure 24 montre comment le degré permet de différencier les liaisons simples, multiples ou aromatiques. La molécule représentée par le graphe G_8 comporte une liaison double et un cycle aromatique à six carbones, ce qui signifie que six électrons sont délocalisés sur le cycle. Cette structure est souvent dessinée comme un cycle hexagonal dans lequel trois doubles liaisons alternent avec trois liaisons simples. En revanche, la molécule qui correspond au graphe G'_8 ne contient que des liaisons simples. Les graphes G_8 et G'_8 semblent identiques, car aucune distinction n'est faite entre les liaisons de natures différentes lors de la transformation des molécules en graphes. Les degrés des atomes de carbone permettent en revanche de différencier ces graphes, et de distinguer les propriétés des deux molécules lors de la régression ou de la classification.

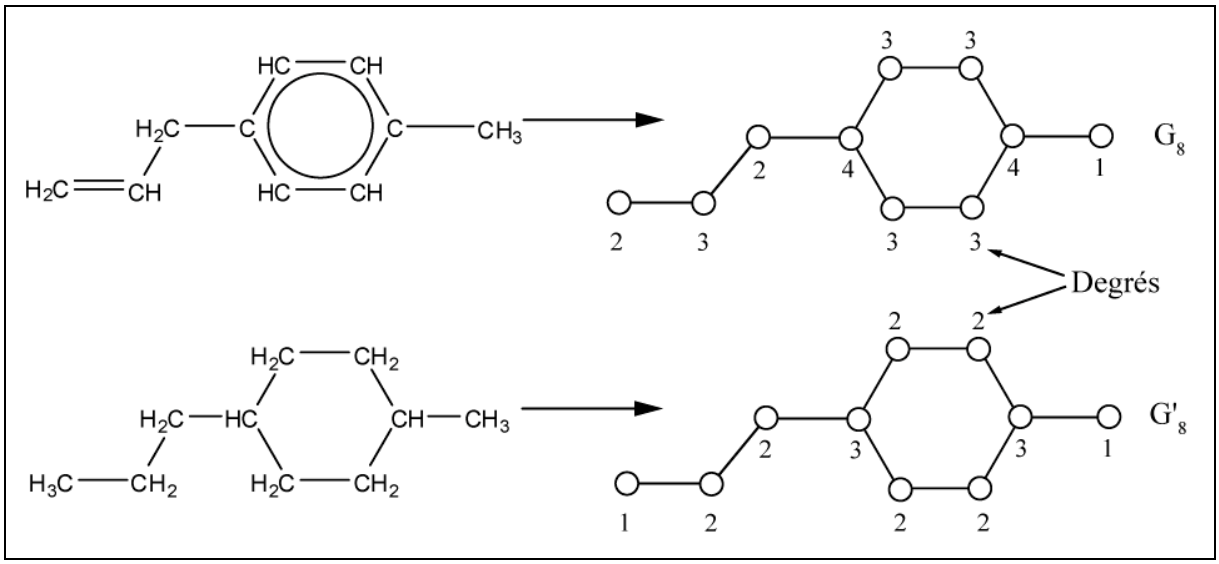


Figure 24 : Rôle du degré pour la représentation en graphes de molécules

Le codage « 1 parmi *n* », où *n* est le degré maximum, est également choisi pour cette étiquette, afin de ne pas imposer de relation quantitative entre les différents degrés. Par exemple, il n'est pas justifié de décider que le degré 3 ait trois fois plus d'influence sur le modèle que le degré 1.

– D'autres informations relatives aux atomes peuvent être nécessaires à la modélisation. Il peut s'agir de la configuration autour de ces atomes, de charges ponctuelles, etc. L'indication de la configuration est particulièrement intéressante, puisqu'elle permet de distinguer les isomères de configuration qui ont bien souvent des propriétés ou des activités différentes, alors que les graphes correspondants semblent identiques.

– Il existe enfin des problèmes pour lesquels il pourrait être nécessaire de fournir des informations globales sur la molécule, ce qui est possible en couplant les *graph machines* aux approches mettant en jeu des descripteurs. Puisque ces descripteurs sont relatifs à la molécule, ils peuvent intervenir grâce à une étiquette placée uniquement sur le nœud racine du graphe correspondant. La fonction associée à une molécule *i* serait alors de la forme : $g^i_{\theta,\Theta} = F_\Theta(\mathbf{z}_r, \mathbf{D})$, où F_Θ est la fonction de nœud associée au nœud racine et **D** un vecteur de descripteurs. Cette expression est à comparer avec la fonction sans descripteur : $g^i_{\theta,\Theta} = F_\Theta(\mathbf{z}_r)$.

Les étiquettes associées aux nœuds permettent ainsi de représenter les molécules par des graphes tout en conservant les informations sur la nature des atomes et des liaisons. Elles laissent également la possibilité d'utiliser des descripteurs, susceptibles d'améliorer les résultats de modélisation de propriétés complexes.

I.3 - Conversion des graphes en arborescences - choix de l'algorithme

Les graphes associés aux molécules doivent être transformés en arborescences pour permettre la construction des *graph machines*. Cette opération s'effectue selon l'algorithme évoqué dans la section VI.1 du chapitre 2. La numérotation canonique des nœuds nécessaire au choix du nœud racine peut s'effectuer selon un algorithme développé spécifiquement pour les graphes de molécules [68], et détaillé dans l'annexe 1. Les principaux critères de numérotation sont : la nature de l'atome associé au nœud, son degré, le type de liaisons qu'il forme avec les autres atomes et sa distance aux autres nœuds. Le nœud portant le numéro 1 est alors choisi comme nœud racine.
Considérons par exemple la molécule de 2,3-diméthylbut-1-ène, représentée sur la Figure 25. La numérotation des atomes est indiquée à côté de chacun d'eux.

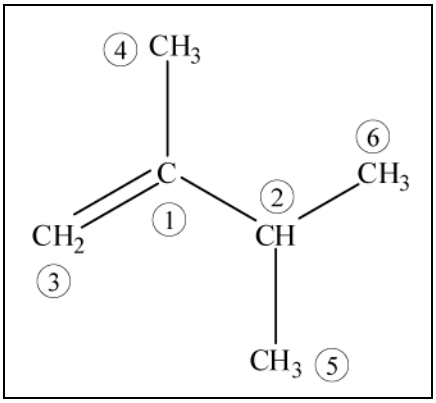


Figure 25 : Exemple de numérotation canonique d'atomes d'une molécule

- Les atomes 1 et 2 sont les deux atomes les plus proches du "centre" de la molécule : dans le graphe correspondant, ce sont les nœuds dont la distance maximale aux autres nœuds du graphe est la plus faible (ces deux nœuds sont séparés des autres nœuds par deux arêtes au maximum). Le degré de l'atome 1 ($d_1 = 4$) est plus élevé que celui de l'atome 2 ($d_2 = 3$) : il est donc choisi comme atome central.
- De manière identique, l'atome 3 est choisi ainsi car son degré est supérieur à celui de l'atome 4.
- Enfin, les atomes 5 et 6 jouent le même rôle, leur numérotation n'a d'ailleurs aucune influence.

Nous avons étudié l'importance du choix du nœud racine et du lieu d'ouverture des cycles en comparant les capacités de généralisation de modèles obtenus selon différentes méthodes

I.3.1 - Influence de la distance nœud racine - centre du graphe pour des alcanes

Le premier problème à étudier est celui du choix du nœud racine, et de l'influence de ce choix sur les performances du modèle. Nous avons pour cela étudié la modélisation de la température d'ébullition de 149 alcanes, représentés par des graphes acycliques comportant de

2 à 10 nœuds de même nature (atomes de carbone). Chaque nœud d'un graphe peut être choisi comme nœud racine ; nous avons donc créé 10 bases de 149 arborescences, en utilisant pour chacune de ces bases une méthode de choix du nœud racine différente. Les N_i nœuds de chaque graphe i sont numérotés de façon canonique, de $Num_i = 1$ à N_i, suivant la méthode décrite dans l'annexe 1 .On choisit alors comme nœud racine, pour la base K, le nœud dont le numéro est donné par :

$$Num_i = E\left(K.\frac{N_i}{10} \right) + 1$$

La base d'indice $K = 0$ correspond au choix du nœud dont le numéro est 1 : il s'agit du nœud central. La base d'indice $K = 9$ correspond au nœud dont le numéro est N_i , donc le plus éloigné du nœud central.
Une onzième base est obtenue en choisissant aléatoirement pour chaque molécule son nœud racine. Nous avons alors réalisé une validation croisée (10-CV) pour chacune de ces 11 bases, et comparé la qualité des modèles en généralisation, pour une complexité donnée (les fonctions de nœud sont des réseaux de neurones à 4 neurones cachés). Les résultats sont présentés sur la Figure 26.

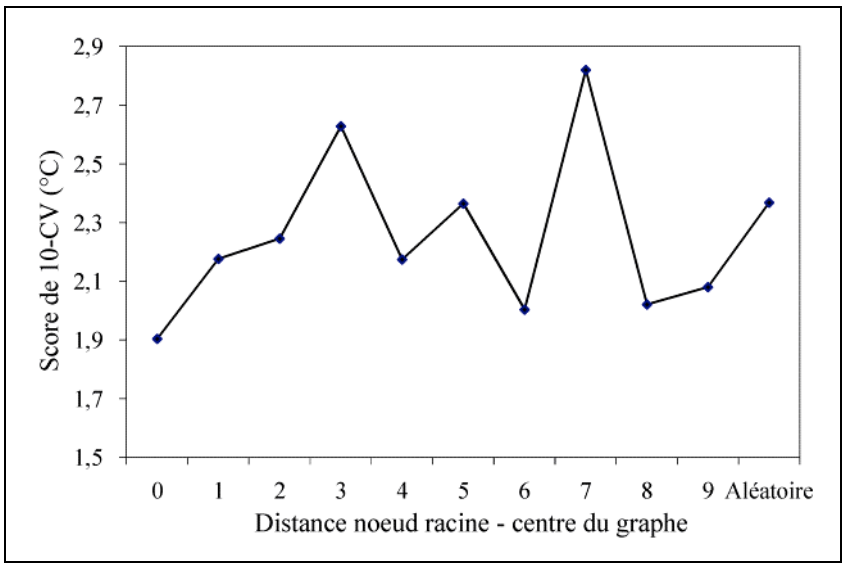


Figure 26 : Comparaison des performances de modèles obtenus par différentes méthodes de décomposition

Il semble légèrement favorable de choisir comme nœud central un nœud proche du centre du graphe pour la prédiction de la température d'ébullition d'un alcane, mais ce choix n'a pas d'influence importante sur les performances du modèle.

Nous avons également comparé ces scores de validation croisée à ceux obtenus lorsque les sous-bases d'apprentissage utilisent un découpage identique (indices de 0 à 9), mais que les exemples de validation sont, eux, découpés de façon aléatoire. Ces résultats sont représentés sur la Figure 27.

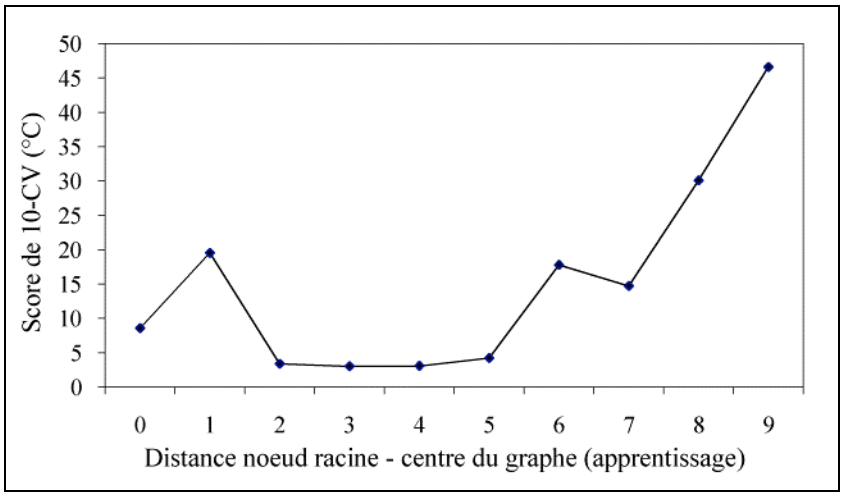


Figure 27 : Comparaison des performances lorsque l'algorithme de choix du nœud racine des exemples de validation est aléatoire

Les scores de validation croisée sont alors moins bons, et peuvent même être décuplés, quel que soit l'algorithme choisi pour les exemples d'apprentissage. Ceci semble indiquer qu'il est important, lors de la prédiction sur de nouvelles molécules, de garder le même algorithme que pour les graphes d'apprentissage.

I.3.2 - Influence de la distance nœud racine-fonction pour des molécules monofonctionnelles

Nous avons vu que lorsque les exemples étudiés sont des hydrocarbures, acycliques et sans groupe fonctionnel, le choix de l'algorithme modifie peu les performances du modèle, mais qu'il faut, en revanche, appliquer le même algorithme en prédiction qu'en apprentissage. Nous allons maintenant étudier, dans le cas de molécules monofonctionnelles, l'influence de la distance entre le nœud racine et le groupe porteur de la fonction (par exemple, pour un alcool, la distance entre le nœud racine et le nœud associé à l'oxygène). La base particulière étudiée est constituée de 107 molécules acycliques monofonctionnelles (alcools, aldéhydes, cétones, acides carboxyliques, amines et nitriles), comportant de 1 à 10 atomes non-hydrogène. Chacune de ces molécules est transformée en arborescence dont les nœuds racines correspondent successivement à chacun des atomes. Ces décompositions sont numérotées de 0 à 9 suivant la distance entre le nœud central et la fonction de la molécule : 1 correspond à un découpage sur l'atome portant la fonction, 9 sur l'atome le plus éloigné, et 0 sur la fonction elle-même. L'apprentissage est réalisé sur l'ensemble de ces décompositions, c'est-à-dire sur une base de 10*107 exemples. Tout d'abord, nous souhaitons déterminer s'il existe un algorithme menant à de meilleurs résultats en prédiction que les autres. Nous avons ainsi réalisé une série d'apprentissages du coefficient de partage eau-octanol (logP) sur l'ensemble des 1070 exemples, puis calculé le score de leave-one-out virtuel pour chacun des sous-ensembles correspondant à un algorithme particulier, avec des réseaux de 2 à 4 neurones cachés (2N à 4N). Les résultats sont résumés sur la Figure 28.

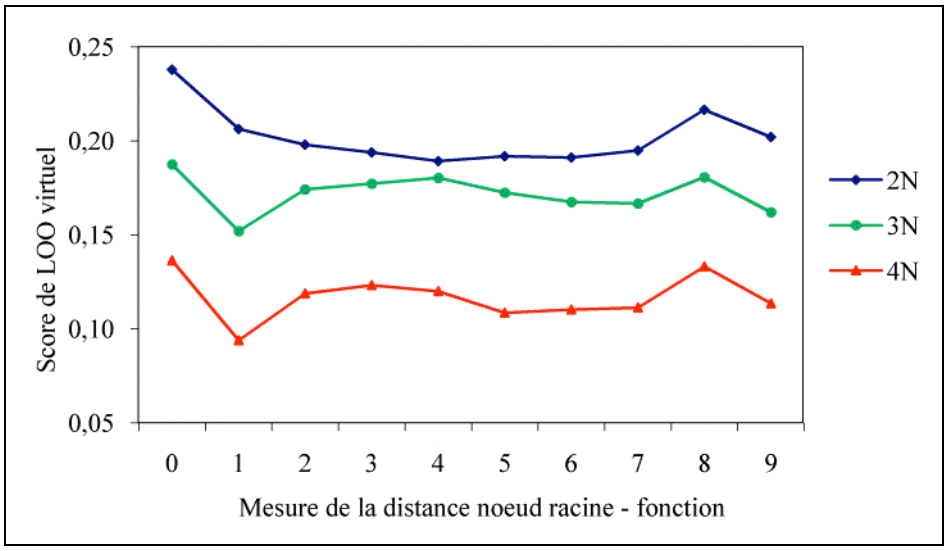


Figure 28 : Évolution des scores de LOO virtuels suivant la décomposition en graphes (apprentissage du logP)

Ces résultats semblent indiquer que le choix de l'algorithme influe peu sur la capacité de généralisation du modèle. Il semble cependant plutôt favorable de choisir comme nœud central le carbone qui porte la fonction (numéro 1), et défavorable de choisir le nœud correspondant à la fonction-même (l'oxygène d'une fonction alcool par exemple), comme l'illustre la Figure 29.

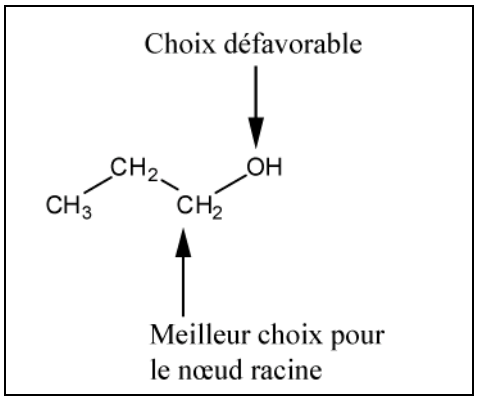


Figure 29 : Exemple de choix de nœud racine pour une molécule monofonctionnelle

Afin d'estimer les performances de chaque méthode de décomposition prise individuellement, nous avons créé 11 bases de 107 exemples, en appliquant pour chacune d'elle une méthode de décomposition particulière (10 bases pour lesquelles la distance entre le nœud central et la fonction varie et une base où le nœud central est choisi de façon aléatoire). Une série d'apprentissages suivis de calculs des scores de leave-one-out virtuel a alors été réalisée sur chacune des bases. Nous avons comparé les performances en généralisation, estimées par les scores de leave-one-out virtuel, obtenues par les différentes méthodes prises séparément. Ces résultats sont présentés sur la Figure 30.

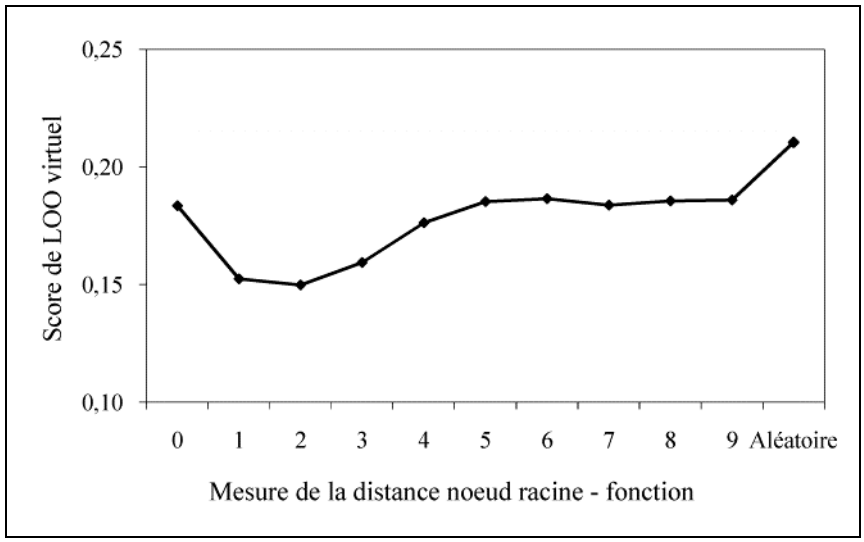


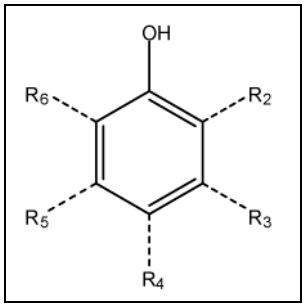
Figure 30 : Scores de LOO virtuel obtenus avec les différentes méthodes de décomposition (apprentissage du logP)

Ces résultats indiquent que le choix de la méthode utilisée pour obtenir des arborescences à partir des structures a peu d'influence sur la qualité des résultats, puisque les différentes méthodes conduisent à 8 % près (écart-type de 0,02) au même score de leave-one-out virtuel. La valeur du score obtenu lorsque le nœud racine est choisi de façon aléatoire, plus élevée que les autres d'environ 20 %, confirme que, si les différents algorithmes semblent équivalents, il est important, en revanche, d'avoir recours à la même méthode de création des arborescences pour tous les exemples.

I.3.3 - Choix de l'algorithme d'ouverture des cycles

Enfin, nous avons étudié l'influence du choix de la liaison à rompre dans un cycle pour former un graphe acyclique, en comparant les performances obtenues par différentes méthodes pour la modélisation de la toxicité de phénols. La base de données est constituée de 153 molécules, dont la toxicité sur un poisson d'eau douce, le *Tetrahymena pyriformis*, est connue [31, 71]. Elle est exprimée comme le logarithme de la concentration (en mmol/L) nécessaire pour inhiber de 50 %, après 40 h, la croissance de la population de cellules. La modélisation de cette activité est par ailleurs détaillée dans un article reproduit en annexe 2 [52].

Les molécules étudiées sont formées à partir d'un cycle benzénique et d'un ou plusieurs substituants, dont le groupement hydroxyle -OH.



Tous les graphes comportent donc au moins un cycle, qu'il est possible d'ouvrir en supprimant une des six arêtes le constituant. En considérant que le nœud central est fixé (nous avons choisi le carbone portant le groupement hydroxyle), il existe au plus six façons possibles de transformer les molécules de la base en graphes acycliques orientés.
L'étude des performances obtenues avec les différents algorithmes s'est déroulée selon la même démarche que précédemment. Nous avons formé trois bases contenant chacune un exemplaire de chaque graphe, pour lequel l'ouverture du cycle a été réalisée à une distance donnée du nœud central (0, 1 ou 2 nœuds suivant la base). Nous avons de plus constitué une quatrième base en supprimant aléatoirement une arête de chaque cycle, afin de déterminer si, comme nous l'avions remarqué pour le choix du nœud central, la méthode de construction des graphes doit être unique. Les résultats, représentés sur la Figure 31, confirment que le site d'ouverture des cycles présente peu d'importance sur les performances en généralisation des modèles.

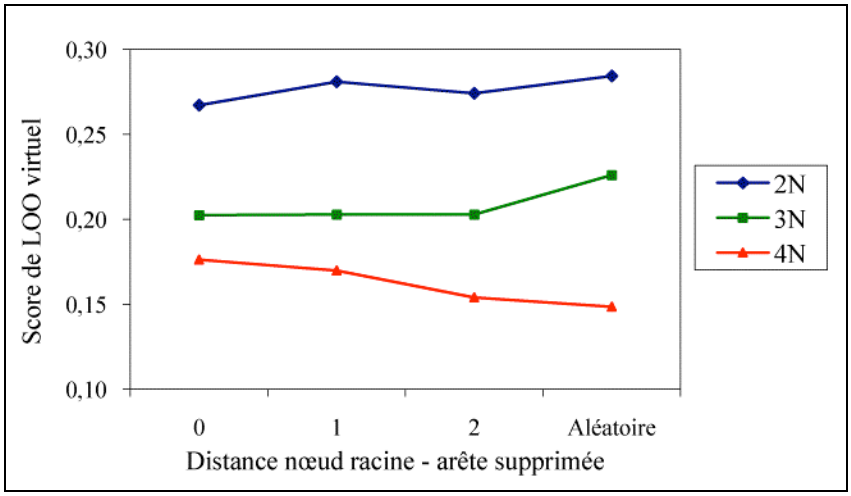


Figure 31 : Comparaison des scores de leave-one-out virtuels obtenus par chaque méthode d'ouverture des cycles.

Il semble par contre que, contrairement à ce que nous avions observé pour le choix du nœud central, il ne soit pas nécessaire d'ouvrir tous les cycles d'une base selon la même méthode - près ou loin du nœud central - pour améliorer les performances en prédiction.

Les essais effectués suggèrent donc qu'il est pertinent d'utiliser systématiquement le même algorithme pour la conversion des graphes en arborescences ; nous avons ainsi choisi celui décrit par Jochum [68], qui propose une forme canonique des graphes et respecte leur symétrie.

II - Sélection des exemples de la base d'apprentissage

Un problème de prédiction ou de classification comporte souvent des contraintes liées à la base d'exemples disponibles. Lorsque le modèle est établi sur un ensemble d'exemples trop restreint, ses capacités de généralisation peuvent être mauvaises. Les exemples peuvent également être nombreux, mais fournir peu d'informations significatives. Enfin, si la base d'apprentissage est déséquilibrée, c'est-à-dire si la répartition des mesures dans le domaine expérimental n'est pas uniforme, le modèle risque d'être très bon pour certains exemples, mais très mauvais sur d'autres. Il est donc nécessaire de choisir soigneusement la base d'apprentissage sur laquelle le modèle est ajusté. Cette sélection peut être faite à l'aide des leviers, qui mesurent l'influence de chaque exemple sur le modèle et qui, comme nous l'avons montré en section V.3.1 du chapitre 2, peuvent être calculés dans le cadre des *graph machines*. Idéalement, pour une base d'apprentissage bien équilibrée, les leviers ont tous une valeur voisine de $\frac{q}{N}$, où q est la dimension du modèle et N le nombre d'exemples.

Nous allons montrer, à l'aide de plusieurs exemples, que les leviers permettent de repérer des déséquilibres dans les données. Le premier exemple consiste à classifier des molécules en deux catégories, les aromatiques (A) et les non-aromatiques (NA). L'apprentissage a été réalisé sur 6 bases de 170 exemples contenant des proportions respectives de 5 %, 10 %, 20 %, 30 %, 40 % et 50 % de molécules non-aromatiques. Les taux de classification obtenus par apprentissage sont de 100 %. Les moyennes des leviers des molécules non-aromatiques sont reportées dans le Tableau 2.

Proportion de non-aromatiques	5 %	10 %	20 %	30 %	40 %	50 %
Levier moyen (NA)	0,85	0,57	0,35	0,24	0,18	0,18

Tableau 2 : Évolution de la valeur moyenne des leviers avec la composition de la base d'apprentissage

Nous constatons que les leviers des molécules non-aromatiques sont d'autant plus élevés que leur proportion est faible. De plus, ils sont supérieurs à ceux des molécules aromatiques, dont la plus forte moyenne est de 0,16. Ces écarts traduisent le déséquilibre des bases d'apprentissage. Puisque les molécules non-aromatiques sont en nombre plus faible, il faut que chacune d'elles ait une forte influence sur le modèle pour être correctement classée.

Les leviers sont également utiles pour mesurer des déséquilibres plus subtils, ce qui peut être mis en évidence sur un exemple d'apprentissage de la masse molaire de dérivés halogénés des alcanes (comportant des atomes de fluor, chlore et brome). On peut considérer que la base est déséquilibrée lorsque les molécules fluorées, chlorées, bromées et iodées n'y sont pas représentées en proportions égales, ou lorsque la distribution des valeurs de la grandeur modélisée - ici la masse molaire - n'est pas uniforme sur cette base. Nous avons choisi d'étudier les leviers en faisant varier la proportion de molécules contenant des atomes de brome dans la base d'apprentissage. L'atome de brome est en effet le plus lourd des trois

halogènes considérés : sa masse molaire vaut 79,9 g/mol, alors que celles du fluor et du chlore valent respectivement 19 g/mol et 35,5 g/mol. Lorsque la proportion des molécules contenant des atomes de brome varie, la distribution des valeurs de la masse molaire sur la base d'apprentissage est modifiée : le nombre d'exemples de grande masse molaire diminue avec la proportion de molécules bromées. Ces molécules devraient alors avoir une influence croissante sur le modèle. De plus, ce sont les seules qui définissent la relation entre l'atome de brome et la masse molaire (les atomes de brome interviennent sur la prédiction du modèle par l'intermédiaire des paramètres liés à l'étiquette "Brome"). Lorsque ces exemples sont peu nombreux, ils ont une influence d'autant plus grande sur ces paramètres, donc sur le modèle.
Les 5 bases d'apprentissage ont une taille fixe de 330 graphes ; la proportion d'exemples faisant intervenir des atomes de brome (de 1 à 4 atomes) varie entre 5 % et 50 %. La Figure 32 compare les leviers moyens obtenus sur ces différentes bases pour les molécules bromées.

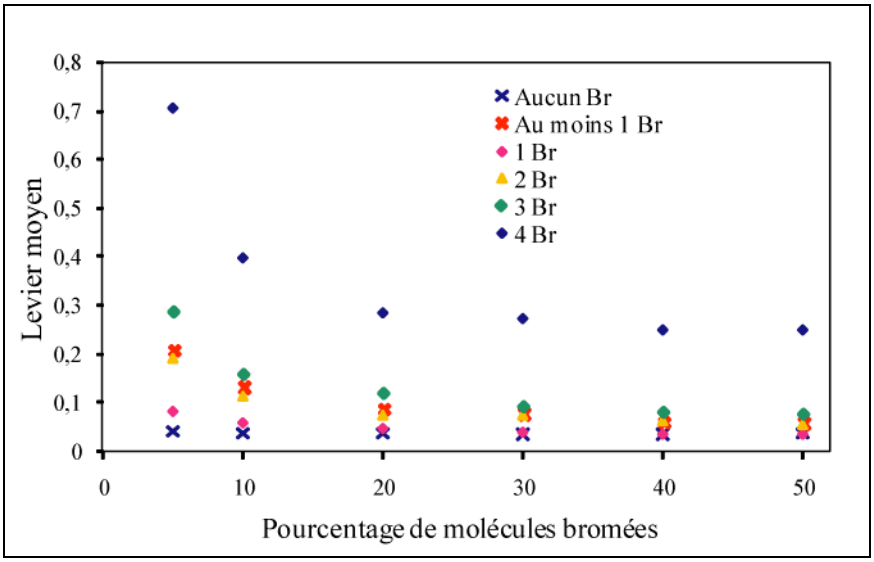


Figure 32 : Évolution du levier moyen des molécules de chaque famille en fonction de leur proportion dans la base d'apprentissage

On constate effectivement une évolution de la valeur des leviers des molécules bromées. Ils sont d'autant plus importants – et supérieurs à la moyenne des leviers – que la proportion des molécules bromées est faible. De plus, pour une base donnée, les leviers augmentent globalement avec le nombre d'atomes de brome : les molécules possédant 4 bromes ont un levier plus important que celles n'en ayant que 3. Le levier permet donc effectivement de mesurer sur cet exemple les éventuels déséquilibres de la base d'apprentissage : des leviers trop importants signifient que les exemples influent fortement sur le modèle, et celui-ci risque d'être surajusté à ces exemples. Dans ce cas, un pourcentage de 15 à 20 % de molécules bromées semble nécessaire pour que ces exemples n'aient pas une trop forte influence.

L'ensemble d'apprentissage peut également contenir des exemples marginaux, soit parce qu'ils présentent des particularités uniques ou rares dans la base, soit parce que les données

associées sont erronées. Nous allons illustrer la détection de ces singularités grâce au calcul des leviers dans chacun de ces deux cas.

Le premier exemple consiste à apprendre l'énergie libre de solvatation sur une base d'apprentissage de 179 composés organiques. Après apprentissage, le calcul des leviers des exemples permet de calculer les leviers moyens de chaque famille de molécules. Ils sont représentés sur la Figure 33 et comparés à la moyenne des leviers sur l'ensemble de la base ($\frac{q}{N} = 0{,}5$).

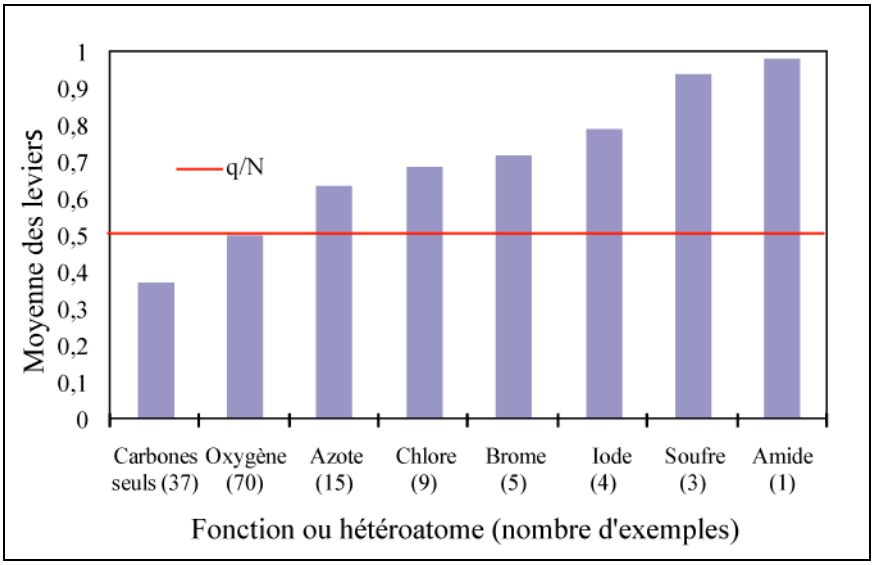


Figure 33 : Leviers de familles de molécules lors de l'apprentissage de l'énergie libre de solvatation

Nous remarquons que les familles de molécules sous-représentées dans la base, par exemple les molécules iodées, qui ne sont que 4, ont un levier supérieur à la moyenne, ce qui traduit leur forte influence sur le modèle. Afin d'équilibrer la base d'apprentissage, il faudrait par exemple l'enrichir de molécules halogénées et d'amides.

Il est ainsi possible, grâce aux leviers, de déceler une mauvaise distribution des structures des exemples, comme c'est le cas pour la modélisation de la masse molaire des alcanes halogénés, ou une mauvaise distribution des sorties, ce que nous avons vu lors de la classification des aromatiques/non-aromatiques, lorsque les deux classes ne comportaient pas le même nombre d'exemples. Les leviers aident également à repérer des exemples isolés ou des données erronées. Il s'agit donc d'un outil efficace pour l'optimisation de la base d'apprentissage.

III - Fonctions de nœud et sélection de la complexité

Lors de la modélisation par les *graph machines*, il est possible d'adapter la complexité du modèle au problème, afin d'obtenir à la fois un coût d'apprentissage et des capacités de généralisation satisfaisants. Cette complexité est déterminée par la fonction de nœud, qui peut être une simple fonction linéaire, un polynôme, un réseau de neurones ou toute autre fonction continue et dérivable, linéaire ou non.

III.1 - Structure de la fonction de nœud

Les réseaux de neurones sont des approximateurs universels, et leur parcimonie est particulièrement intéressante pour la modélisation de propriétés ou d'activités de molécules. Les mesures sont en effet souvent coûteuses, et les bases de données disponibles assez restreintes. Les réseaux de neurones permettent ainsi de tirer le meilleur parti et d'obtenir des modèles plus précis que d'autres approximateurs universels tels que les polynômes. Ce sont également des modèles pour lesquels nous disposons d'un très bon outil de calcul (NeuroOne [72]). Pour toutes ces raisons, les fonctions de nœud sont des réseaux de neurones dans la plupart des problèmes de modélisation abordés dans ce travail. L'architecture type d'un réseau associé à un nœud feuille est détaillée sur la Figure 34.

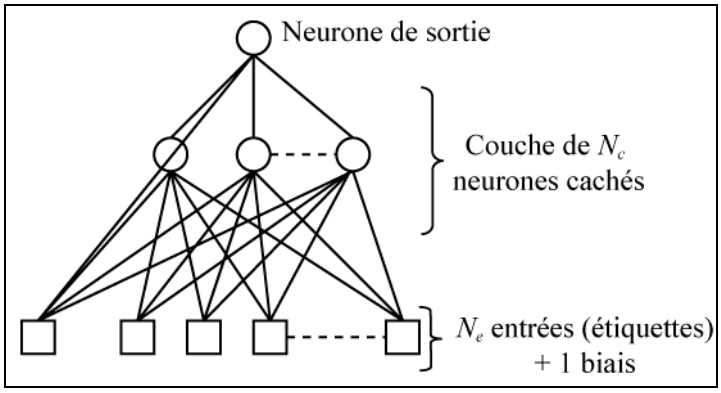


Figure 34 : Schéma d'un réseau de neurones utilisé comme fonction de nœud

Ils sont composés principalement de N_e entrées et d'une couche de N_c neurones cachés.

- Les entrées ne sont pas des entrées classiques de modélisation : il s'agit des étiquettes du nœud associé au réseau. Toutes sont égales à 1, et sont reliées aux neurones cachés de la façon suivante :
  - lorsqu'une étiquette est décrite par le codage « un parmi n », n entrées lui sont allouées ; seule l'entrée correspondant à la valeur réelle de l'étiquette est reliée à la couche cachée.
  - si par contre une étiquette peut prendre des valeurs continues, ou décrit une grandeur quantitative, une seule entrée lui est réservée. Les paramètres entre cette entrée et la couche cachée sont alors fixés, et prennent la valeur de l'étiquette.

Ces entrées ne jouent pas le même rôle que les entrées d'un réseau conventionnel. En effet, dans ce type de réseau, le nombre d'entrées est fixe ; elles sont toutes reliées aux neurones de la couche cachée, et leurs valeurs changent pour chaque exemple de la base d'apprentissage. Le réseau de neurones permet alors d'établir une relation entre ces entrées et la sortie. En revanche, les entrées de chaque fonction de nœud caractérisent le nœud auquel elles sont associées, et diffèrent d'un nœud à un autre.

- Les neurones cachés sont à fonction d'activation sigmoïde pour la plupart des modélisations effectuées (c'est-à-dire les problèmes non-linéaires).

Lorsque le nœud associé au réseau est le nœud racine du graphe, les sorties des neurones cachés sont reliées à un neurone de sortie. Dans le cas contraire (branche ou feuille de l'arbre), ces sorties sont complètement connectées à la couche cachée correspondant au nœud parent. Réciproquement, la couche cachée associée à un nœud branche ou à un nœud racine s_k reçoit des données de ses entrées (étiquettes), comme représenté sur la Figure 34, ainsi que des sorties des couches cachées des nœuds enfants de s_k. Ces deux types de données correspondent aux deux types d'arguments d'entrées de la fonction de nœud décrits dans le paragraphe IV.2 du chapitre 2. Un biais (terme constant) est par ailleurs relié à la couche cachée et au neurone de sortie s'il est présent.

La Figure 35 représente un exemple de *graph machine*, dont la fonction de nœud est un réseau de neurones à 2 neurones cachés, associée au graphe G_9. Celui-ci correspond à la molécule d'éthanol, représentée sur la Figure 36.

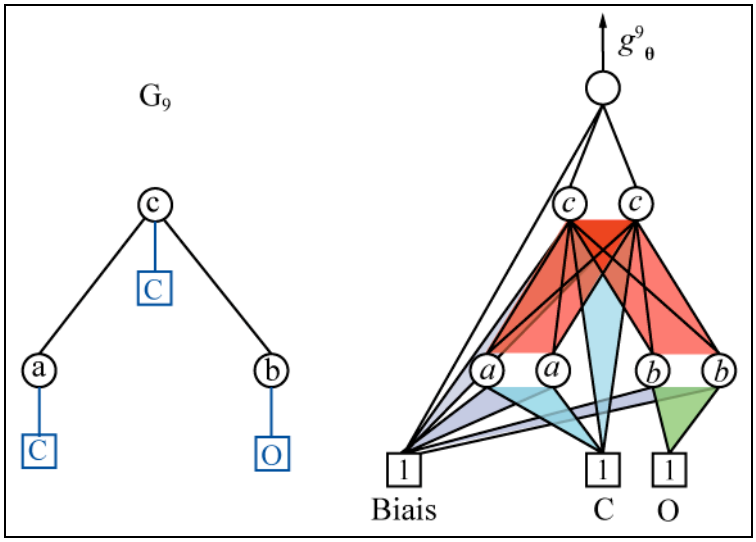


Figure 35 : *Graph machine* construite à partir de réseaux de neurones à 2 neurones cachés

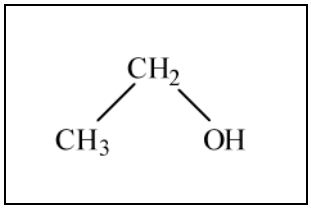


Figure 36 : Molécule d'éthanol

Il n'y a que deux types d'atomes non-hydrogène (carbone et oxygène), donc deux étiquettes possibles, auxquelles sont affectées deux entrées de valeur 1 (notées C et O). À chaque nœud correspond un réseau de neurones, avec une couche cachée de deux neurones. Le réseau *a* est relié au biais et à l'entrée « carbone », le réseau *b* au biais et à l'entrée « oxygène ». Le nœud

c, associé à un atome de carbone, est parent des nœuds a et b : par conséquent le réseau *c* est relié à la fois aux couches cachées des réseaux *a* et *b*, au biais et à l'entrée « carbone ». Puisque les fonctions de nœuds sont les mêmes, les poids des réseaux sont partagés (ils sont identiques d'un réseau à l'autre). Les ensembles de poids identiques sont repérés sur la Figure 35 par des zones de mêmes couleurs.

III.2 - Cas particulier : les *graph machines* pour la classification

La méthode des *graph machines* peut être également appliquée à la classification d'objets structurés, et en particulier à la classification de molécules. Lorsque le problème de classification est à deux classes (la sortie est binaire), il suffit d'associer au nœud racine une fonction à même d'effectuer la classification, telle qu'un réseau de neurones dont le neurone de sortie est une fonction d'activation sigmoïde. Le profil de cette fonction, d'expression $s(x) = \frac{1 + th(x)}{2}$, est représenté sur la Figure 37.

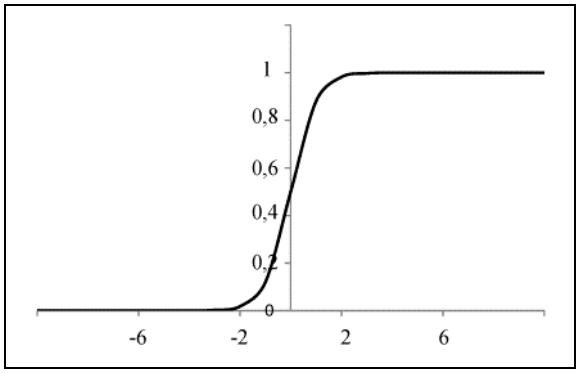


Figure 37 : Profil de la fonction sigmoïde

Cette courbe possède pour asymptotes les droites d'équation $y = 0$ et $y = 1$, et s'approche de la fonction seuil. Le modèle effectue alors une régression logistique, et peut être utilisé pour effectuer une classification.

III.3 - Sélection du modèle

Comme nous l'avons vu, il est possible d'ajuster la complexité des *graph machines* en modifiant la fonction de nœud. La sélection du modèle le plus performant peut être effectuée grâce à des outils traditionnels, tels que les leviers, le leave-one-out (réel ou virtuel) et la validation croisée.

- Pour une complexité donnée (un nombre de neurones cachés donné), plusieurs modélisations sont réalisées sur la base d'apprentissage. Les modèles dont la jacobienne n'est pas de rang plein sont écartés, et ceux qui réalisent les meilleurs scores de leave-one-out sont retenus. Cette sélection est réalisée pour diverses complexités.

- Les scores de leave-one-out des modèles retenus pour chaque complexité sont comparés. Si plusieurs modèles présentent des résultats équivalents, celui dont la distribution des leviers est la plus étroite autour de $\frac{q}{N}$ est choisi, car il minimise le risque de présenter un surajustement.

Les prédictions sur la base de test peuvent dès lors être effectuées à partir du modèle sélectionné. Le calcul de la jacobienne et des leviers associés à chaque exemple de la base de test permet d'accéder à un intervalle de confiance sur ces prédictions.

CHAPITRE 4
Exemples de modélisations de propriétés et d'activités moléculaires par les *graph machines*

Dans ce chapitre nous allons présenter de nombreuses applications des *graph machines* dans le domaine de la prédiction de propriétés et d'activités de molécules. Ces applications nous ont permis non seulement d'améliorer et de tester les *graph machines*, mais aussi de mesurer leur apport dans ce domaine, sur des exemples concrets dans le domaine pharmaceutique ou en toxicologie. Nous présenterons d'abord des exemples de modélisations de propriétés physico-chimiques de molécules. Nous verrons ensuite que les *graph machines* sont un outil de prédiction d'activités moléculaires : elles sont capables d'estimer les risques liés à certains composés (cancérogénicité, toxicité), et d'aider à la découverte de nouveaux médicaments.
Lors de ces modélisations, nous n'avons pas toujours été en mesure de comparer l'erreur de prédiction sur la base de test à l'erreur expérimentale. Celle-ci est en effet absente des bases de données dans la plupart des cas. Nous avons donc estimé les performances des modèles en comparant les erreurs de prédiction aux erreurs obtenues par d'autres types de modèles.

I - Prédiction de propriétés de molécules

I.1 - Prédiction du coefficient de partage eau/octanol

Le transport, le passage à travers les membranes, la bioaccumulation ou encore l'activité pharmacologique d'une molécule peuvent être conditionnés par son partage entre une phase lipidique et une phase aqueuse, c'est-à-dire son caractère hydrophile. Celui-ci peut être quantifié par le coefficient de partage eau-octanol, noté logP, qui mesure la solubilité différentielle d'un soluté dans ces deux solvants non miscibles :

$$\text{logP} = \log\left(\frac{C_{octanol}}{C_{H_2O}}\right)$$

$C_{octanol}$ et C_{H_2O} sont les concentrations du soluté dans l'octanol et l'eau. Le logP est ainsi utilisé dans de nombreux modèles en tant que descripteur, pour la prédiction d'effets toxiques ou biologiques ou d'interactions ligand-récepteur. Cette propriété physico-chimique peut être mesurée, mais ces mesures sont généralement longues et coûteuses. Par conséquent, différentes méthodes de prédiction du logP ont été mises au point, et il existe un nombre

important de logiciels de prédiction de cette propriété. Ceux-ci s'appuient aussi bien sur des méthodes de contribution de groupes (ACD/LogP [73], KowWin [74, 75], cLogP [76]…) que sur des régressions multilinéaires à partir de descripteurs (VLogP [77]) ou sur des réseaux de neurones (AUTOLogP [78]).

Nous avons réalisé un modèle prédictif du coefficient de partage eau-octanol, par apprentissage, à l'aide d'une base de 1050 composés appartenant à diverses familles de molécules. L'erreur expérimentale sur les valeurs du logP est estimée entre 0,2 et 0,3. La modélisation et la sélection du modèle sont réalisées sur un ensemble d'apprentissage de 875 exemples, choisis de façon aléatoire. Le coût d'apprentissage et le score de leave-one-out virtuel, donnés dans le Tableau 3, sont quasiment identiques pour le modèle choisi : celui-ci ne devrait donc pas être surajusté aux données d'apprentissage.

La qualité du modèle obtenu est alors évaluée sur une base de test de 175 exemples. Les résultats d'apprentissage (EQMA) et de test (EQMT) sont comparés dans le Tableau 3 aux résultats obtenus par une méthode de régression par des réseaux de neurones, à partir de descripteurs [79]. Nous y indiquons également le nombre d'exemples dans chacune des bases d'apprentissage (N_{app}) et de test (N_{test}).

Modèle	**GM**	**Réseaux de neurones [79]**
N_{app} / N_{test}	875 / 175	980 / 105
EQMA	**0,29**	0,41
LOO virtuel	0,30	-
EQMT	**0,30**	0,53

Tableau 3 : Comparaison des performances de modélisation du logP des g*raph machines* et de réseaux de neurones

Les prédictions sur la base de test sont également représentées sur la Figure 38.

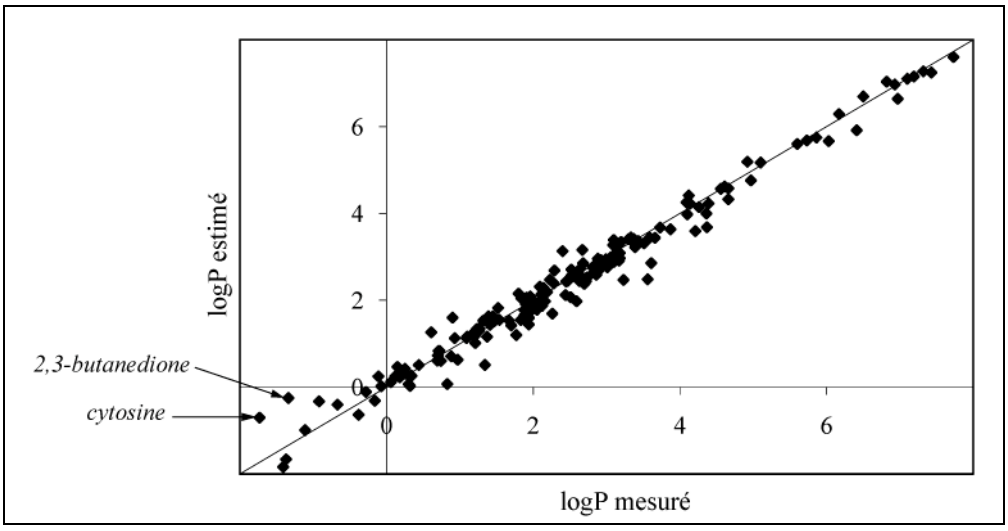


Figure 38 : Prédiction du coefficient de partage eau-octanol sur la base de test

Le coefficient de partage eau-octanol de ces molécules est bien prédit : les erreurs commises par le modèle sur les bases d'apprentissage et de test sont proches de l'erreur expérimentale.

De plus, l'erreur de prédiction sur la base de test est quasiment identique au coût d'apprentissage, ce qui suggère à nouveau que le modèle n'est pas surajusté. Les valeurs du logP les moins bien prédites sont deux valeurs très faibles, qui se situent à la périphérie du domaine d'apprentissage.

I.2 - Prédiction de descripteurs moléculaires

Nous avons montré dans les chapitres 2 et 3 que les *graph machines* réalisent directement des relations entre graphes et nombres, alors que les méthodes traditionnelles de QSAR et QSPR demandent le calcul de descripteurs à partir des structures, puis établissent des relations entre vecteurs et nombres. L'utilisation de cette nouvelle méthode demande que toute l'information nécessaire à la modélisation des propriétés et activités des molécules soit effectivement contenue dans leur structure, et que cette information soit préservée lors de la construction des *graph machines*. Nous avons vérifié cette propriété des *graph machines*, en réalisant la prédiction de plusieurs descripteurs, habituellement calculés à partir des structures moléculaires, et utilisés comme variables explicatives de modèles traditionnels.
Les descripteurs que nous avons étudiés sont des descripteurs importants pour la prédiction du coefficient de partage eau-octanol [80]. Ils ont été calculés après détermination de la géométrie des molécules, par recherche de la conformation de plus basse énergie.
Ces descripteurs sont les suivants :

- la masse molaire (M), qui varie linéairement avec la composition de la molécule ; une fonction de nœud linéaire semble donc bien adaptée à la modélisation de cette propriété ;
- la surface moléculaire (S), paramètre important pour la détermination du logP, à cause de la dépendance entre la surface de la molécule et son énergie libre de solvatation ; la précision du calcul de la surface est contrôlée par la résolution de la grille utilisée pour ce calcul (voir [81]) ;
- l'ovalité de la molécule (O), rapport entre la surface de la molécule et la surface d'une sphère de même volume ;
- la somme des valeurs absolues des charges atomiques portées par les atomes d'oxygène et d'azote (qNO) et par tous les atomes (qT) ;
- la somme des carrés des charges atomiques portées par les atomes d'oxygène (q^2O).

Nous avons prédit chacune de ces propriétés à partir de la base de molécules étudiée dans [80] ; dans cette publication, c'est à partir de ces descripteurs qu'a été réalisée la prédiction du logP. Nous avons pour cela réalisé l'apprentissage et la sélection de modèle sur un ensemble de 257 composés, et mesuré la capacité des graph machines à prédire ces descripteurs sur un ensemble de test de 64 molécules choisies aléatoirement. Nous avons indiqué dans le Tableau 4 les résultats d'apprentissage et de test obtenus pour chacun de ces descripteurs, ainsi que les coefficients de corrélation (R^2) pour la base de test. Ces coefficients permettent en effet d'évaluer la qualité des prédictions. L'écart-type (s) de chaque propriété sur la base de données

est également précisé, car il permet de mieux appréhender la signification des valeurs des erreurs quadratiques d'apprentissage et de test.

Descripteur modélisé	Fonction de nœud	EQMA	Test	
			EQMT	R^2
M *s=60,5*	linéaire	0	0	1
S *s=64*	RN - 3N	1,77	2,08	0,998
O *s=0,155*	RN - 4N	0,010	0,011	0,994
q^2O *s=0,20*	RN - 4N	0,010	0,012	0,996
qNO *s=0,33*	RN - 3N	0,02	0,03	0,992
qT *s=1,1*	RN - 3N	0,07	0,09	0,992

Tableau 4 : Modélisation par les *graph machines* de propriétés utilisées comme descripteurs moléculaires

Nous pouvons ainsi constater que tous ces descripteurs sont prédits avec une grande précision par les *graph machines*. Les erreurs quadratiques moyennes d'apprentissage et de test sont du même ordre de grandeur, ce qui montre les bonnes performances en généralisation des modèles établis. La surface est particulièrement bien prédite : les erreurs en apprentissage et en validation sont de l'ordre de 3 % de l'écart-type des mesures, alors que les erreurs commises sur les charges varient entre 6 % et 9 %. Cet écart peut s'expliquer par la différence de précision des données d'apprentissage. Les valeurs expérimentales de la surface sont en effet très précises, alors que l'erreur expérimentale sur les charges est plus importante. La méthode de calcul de ces charges n'est pas parfaite, et demande plusieurs calculs dont les erreurs s'accumulent.
La capacité des *graph machines* à prédire ces descripteurs explique pourquoi il est possible de modéliser la plupart des propriétés et activités sans avoir à calculer ces descripteurs, directement à partir des structures des molécules : les *graph machines* contiennent en effet toute l'information nécessaire à la modélisation.

I.3 - Énergie libre de solvatation de diverses molécules

La solvatation est le phénomène observé lorsqu'un composé chimique, introduit à l'état solide dans un solvant, réussit à rompre les liaisons qui le maintiennent dans l'état solide, et se dissout dans le solvant. L'énergie de solvatation d'une espèce dépend de la nature du solvant et du soluté : un composé polaire est généralement très bien solvaté dans un solvant

également polaire, tandis qu'un composé apolaire est mieux solvaté dans un solvant apolaire. L'énergie de solvatation est ainsi souvent associée à son énergie de transfert d'un milieu hydrophobe vers un milieu hydrophile, mesurée par le coefficient de partage. Pour l'industrie pharmaceutique, cette propriété est donc un facteur clé dans le choix d'une molécule, car elle détermine son devenir dans le corps humain.

L'énergie libre $\Delta_{solv}G°$ de solvatation dans l'eau (exprimée en kJ/mol) a été modélisée à l'aide de nombreuses méthodes de QSPR, dont la méthode de contribution de groupes [82], et plus récemment celle des réseaux récursifs [83]. La base de données étudiée dans les articles cités est constituée de 179 composés organiques acycliques monofonctionnels. Nous la décomposons en une base d'apprentissage et une base de test, de respectivement 146 et 33 exemples, semblables aux bases étudiées dans l'article [83], afin de comparer les performances des *graph machines* à celles des réseaux récursifs (notés RNNs).

Une première série d'apprentissages vise en premier lieu à déterminer les éventuels exemples isolés ou les erreurs dans la base. Les leviers des exemples sont calculés, et trois exemples retirés : le méthane (qui ne comporte qu'un atome de carbone et par conséquent est un exemple isolé), l'acétamide et le fluorométhane (qui sont les seuls représentants de leurs classes de molécules, respectivement amide et fluoroalcane). Le thiobisméthane, qui contient un atome de soufre, est par ailleurs déplacé de la base de test à la base d'apprentissage, car peu d'exemples de celle-ci en contiennent également un.

La complexité du modèle est déterminée par une première étude visant à comparer les scores de leave-one-out virtuel des modèles de 3 à 5 neurones cachés (les réseaux de plus grande complexité comportent un trop grand nombre de paramètres ajustables par rapport au nombre d'exemples dans la base d'apprentissage). Les performances des meilleurs modèles obtenus sont données dans le Tableau 5 (coût d'apprentissage et score de leave-one-out virtuel, notés EQMA et E_p).

Modèle	**3N**	**4N**	**5N**
EQMA (kJ/mol)	0,58	0,44	**0,36**
E_p (kJ/mol)	0,86	0,78	**0,74**

Tableau 5 : Sélection du modèle pour l'apprentissage de l'énergie libre de solvatation

Le coût d'apprentissage (EQMA), qui mesure l'ajustement du modèle aux données, est optimum pour un modèle à 5 neurones cachés, de même que le score de leave-one-out virtuel. Ce modèle est donc retenu pour la prédiction sur la base de test. Les résultats obtenus sont reportés dans le Tableau 6 et sur la Figure 39.

Modèle	**GM**	**RNNs [83]**
EQMT (kJ/mol)	**0,44**	0,77
R^2	**0,998**	0,994

Tableau 6 : Prédiction de l'énergie libre de solvatation

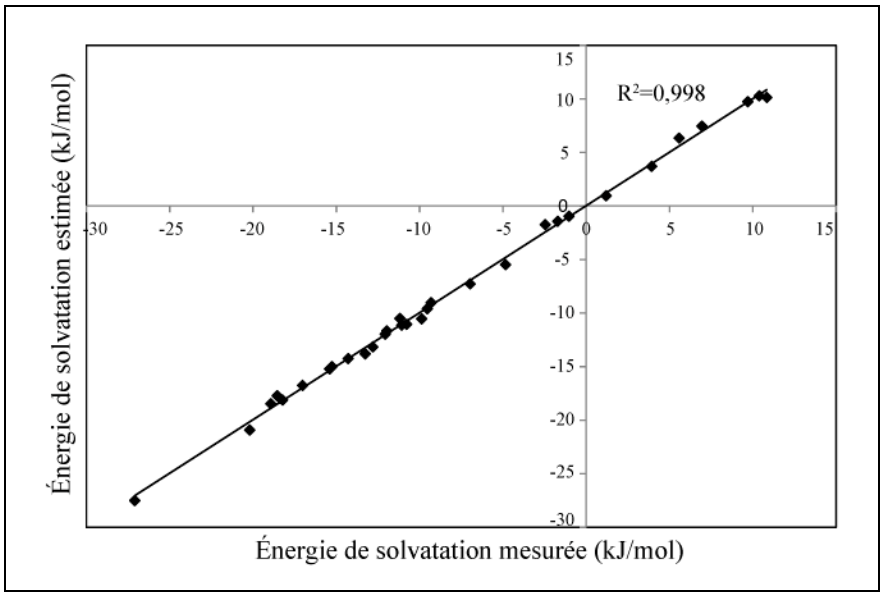


Figure 39: Prédiction de l'énergie libre de solvatation sur la base de test

Les performances des *graph machines* sont ici bien supérieures à celles des réseaux récursifs, et aucune erreur importante de prédiction n'est réalisée. L'énergie libre de solvatation de molécules peut donc être prédite très précisément à partir de leurs seules structures.

I.4 - Prédiction de propriétés sur une même base de molécules

Les méthodes traditionnelles de modélisation (contribution de groupe, descripteurs…) présentent l'inconvénient d'être spécifiques à une grandeur donnée. En effet, les groupes d'atomes importants, ainsi que les descripteurs pertinents, varient d'une propriété à une autre. Il est donc nécessaire de les sélectionner et de déterminer la contribution associée aux groupes, ou de calculer les descripteurs, à chaque nouvelle modélisation. Par opposition, les *graph machines* permettent davantage de souplesse, et ne sont pas spécifiques à une propriété. En effet, la conversion des structures en *graph machines* minimise la perte d'information, contrairement à la conversion des molécules en vecteurs de données, lors de laquelle seules les informations que l'on estime nécessaires à la modélisation d'une propriété donnée sont gardées.

La modélisation de propriétés de molécules a été largement étudiée, à l'aide de diverses méthodes, et des bases de données généralement riches et fiables sont disponibles pour nombre d'entre elles. L'application des *graph machines* à ces modélisations nous permet donc à la fois d'évaluer leurs performances et de les comparer à celles d'autres méthodes. De plus, le devenir d'un composé dans l'eau, l'air ou le sol, est fortement lié à ses propriétés physico-chimiques telles que la pression de vapeur saturante, la solubilité dans l'eau, le coefficient de partage eau-octanol… En particulier, l'assimilation d'une molécule par l'organisme est un critère important de sélection pour l'industrie. La prédiction de telles propriétés permettrait par conséquent d'accélérer le processus de recherche de nouveaux médicaments. Par ailleurs, l'évolution de composés potentiellement toxiques dans l'environnement dépend également de

ces propriétés, dont la connaissance est donc nécessaire lors du développement de produits chimiques ou de leur retraitement.

Nous avons étudié une base unique de molécules, dont nous avons modélisé plusieurs propriétés à l'aide des *graph machines*. Nous disposons ainsi d'une base de 417 molécules, issue de la base de données PhysProp [84], pour lesquelles les valeurs d'une ou plusieurs propriétés sont connues :

- le coefficient de partage eau-octanol (logP)
- la température d'ébullition à la pression de 760 mm Hg (notée T_{eb})
- la pression de vapeur saturante (notée P_{sat}, en Torr)
- la solubilité dans l'eau (log S, en log mol/L).

Ces molécules peuvent contenir divers groupes fonctionnels, faisant intervenir des atomes de carbone, oxygène et d'azote. La répartition de ces groupes est représentée sur la Figure 40.

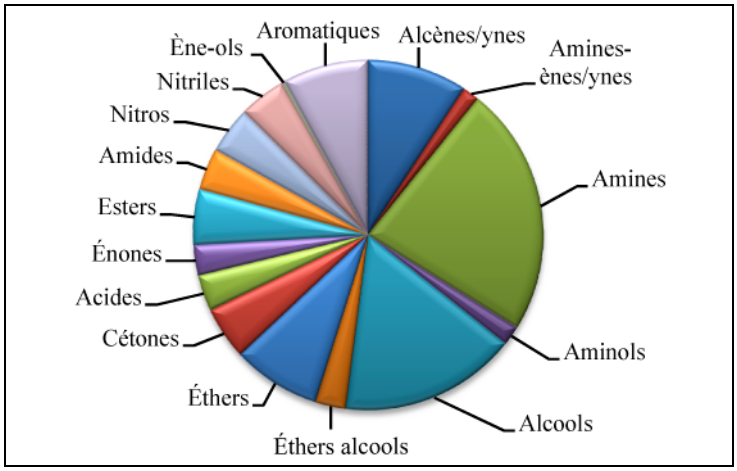


Figure 40 : Répartition en familles des molécules de la base

I.4.1 - Prédiction de la température d'ébullition

La température d'ébullition d'un composé dépend de sa masse molaire et des interactions intermoléculaires dans la phase liquide, par exemple des interactions dipolaires ou des liaisons hydrogène. Puisque ces caractéristiques sont liées à la structure de la molécule, les *graph machines* devraient être à même de modéliser cette propriété. La température d'ébullition a fait l'objet de nombreuses modélisations, souvent spécifiques à une famille particulière de molécules, par exemple aux hydrocarbures ou aux composés halogénés.

I.4.1.1 - Température d'ébullition de composés de la même famille

Nous avons d'abord construit différents modèles de la température d'ébullition, chacun d'eux étant associé à une famille donnée de molécules. Les performances de deux de ces modèles sont résumées dans le Tableau 7. Dans ces deux exemples, les bases de données étudiées sont identiques à celles des articles cités, qui présentent la modélisation de la

température d'ébullition par des réseaux de neurones récursifs (RNNs) [85] et par régression multilinéaire (MLR) [86].

Famille de molécules	**Alcanes**		**Halogénoalcanes**	
N_{app} / N_{test}	135 / 15		507 / 36	
Méthode	GM 4N	RNN [85]	GM 6N	MLR [86]
EQMA (K)	**1,0**	2,0	**3,5**	6,6
EQMT (K)	**1,5**	3,0	**3,7**	7,4

Tableau 7 : Résultats de modélisation de la température d'ébullition d'alcanes et d'halogénoalcanes

Les résultats de modélisation obtenus sur une base d'alcanes se révèlent très bons, et surpassent ceux de réseaux de neurones récursifs [85], à la fois en apprentissage et en test. Les performances sur la base d'halogénoalcanes sont également satisfaisantes, bien que les coûts soient plus élevés que sur la base d'alcanes. En effet, la précision sur les données expérimentales est moins bonne sur les halogénoalcanes que sur les alcanes, pour plusieurs raisons :

- Les valeurs de la température d'ébullition sont issues de la base Beilstein [87]. Or, plusieurs valeurs différentes sont parfois disponibles pour la même molécule. Les auteurs de l'article de référence en ont alors conservé la moyenne, lorsque la différence entre les mesures n'excédait pas 4 K.
- Les auteurs de l'article [86] n'excluent pas que certaines de ces valeurs soient en réalité estimées, et non pas mesurées.
- Les valeurs reportées dans la base sont parfois données sans indication de pression, et il possible que certaines d'entre elles ne soient pas mesurées à 760 mm Hg, pression de référence pour la température d'ébullition.

Les *graph machines* fournissent des résultats satisfaisants, puisque les erreurs en apprentissage et en test sont du même ordre que l'erreur sur les données mesurées. Ces résultats sont également meilleurs que ceux obtenus par régression multilinéaire, à partir de 6 descripteurs 1D et 2D sélectionnés parmi plus de 1000. Ce modèle de régression présente en particulier des lacunes pour les fluoroalcanes. Selon les auteurs, ces erreurs seraient dues à la présence de forts moments dipolaires entre les atomes de carbone et de fluor, qui créent des interactions inter- et intramoléculaires, et dont les descripteurs calculés ne tiennent pas compte. En revanche, ces difficultés de prédiction ne se présentent pas avec les *graph machines*.

La modélisation de la température d'ébullition est donc très satisfaisante lorsque les molécules de la base appartiennent toutes à la même famille. Ce phénomène s'explique bien si l'on suppose que la température d'ébullition dépend principalement de variables structurales au sein de ces familles de molécules : ces variables sont en effet toutes prises en compte par les *graph machines*.

I.4.1.2 - Base de composés divers

Nous avons alors modélisé cette propriété sur une base de 407 molécules diverses, issues de la base globale de 417 molécules. L'erreur expérimentale dépend des composés considérés, et sa borne supérieure a été estimée à environ 10 K [88]. Les performances des *graph machines* sur cette base sont comparées à celles de régression multilinéaire [89], dans le Tableau 8.

Méthode	**GM**	**Descripteurs - MLR [89]**
N_{app} / N_{test}	326 / 81	268 / 78
EQMA (K)	**9,41**	12,6
EQMT (K)	**10,8**	16,7

Tableau 8 : Comparaison des résultats de modélisation de la température d'ébullition

Les prédictions sur la base de test sont représentées sur la Figure 41. Elles sont satisfaisantes, car l'erreur quadratique moyenne est proche de l'erreur expérimentale. De plus, les erreurs observées sont inférieures à celles obtenues par régression, à partir de 129 descripteurs. Les erreurs sont cependant plus importantes pour certains types de composés, tels que les amines. On remarque également que la diversité des molécules provoque une augmentation des erreurs de prédiction.

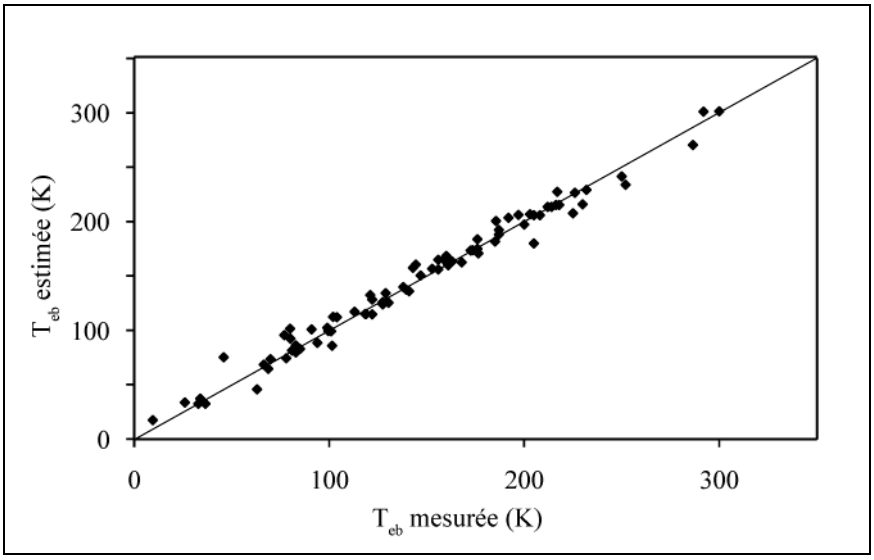


Figure 41 : Prédiction de la température d'ébullition de divers composés organiques (base de test)

Nous pouvons supposer que d'une famille de composés à une autre, la variation de la température d'ébullition dépend de variables qui ne sont pas prises en considération par le modèle, telles que l'existence de liaisons intermoléculaires. Ces variables n'interviendraient pas lors de la modélisation de la température d'ébullition de composés issus de la même famille, car elles seraient identiques pour toutes les molécules. Par exemple, dans un acide carboxylique, le groupe carbonyle C=O est polaire, le moment dipolaire étant orienté vers l'atome d'oxygène. Les molécules ont donc tendance à former des liaisons hydrogène, et à se

présenter sous la forme de dimères, ayant une température d'ébullition supérieure à celle attendue. Les amides forment également des liaisons hydrogène, mais elles sont plus fortes que celles des acides, ce qui conduit à une température d'ébullition plus élevée. Cette hypothèse permet d'expliquer l'augmentation des erreurs de prédiction, qui restent cependant plus faibles que celles obtenues avec le modèle multilinéaire.

I.4.2 - Modélisation du coefficient de partage eau-octanol (logP)

Nous avons ensuite appliqué les *graph machines* à la prédiction du logP sur la base complète des 417 molécules, et évalué leurs performances en comparaison avec des réseaux de neurones. Les entrées de ces réseaux sont des descripteurs topologiques sélectionnés par régression multilinéaire (MLR-NN) ou par analyse en composantes principales (PC-NN) [90].

Méthode	GM	MLR-NN [90]	PC-NN [90]
N_{app} / N_{test}	335 / 82	322 / 57	322 / 57
EQMA	**0,15**	0,35	0,22
EQMT	**0,19**	0,43	0,27

Tableau 9 : Comparaison des prédictions du logP par différentes méthodes

Les erreurs sur les bases d'apprentissage et de test sont données dans le Tableau 9, et les prédictions sur la base de test sont représentées sur la Figure 42.

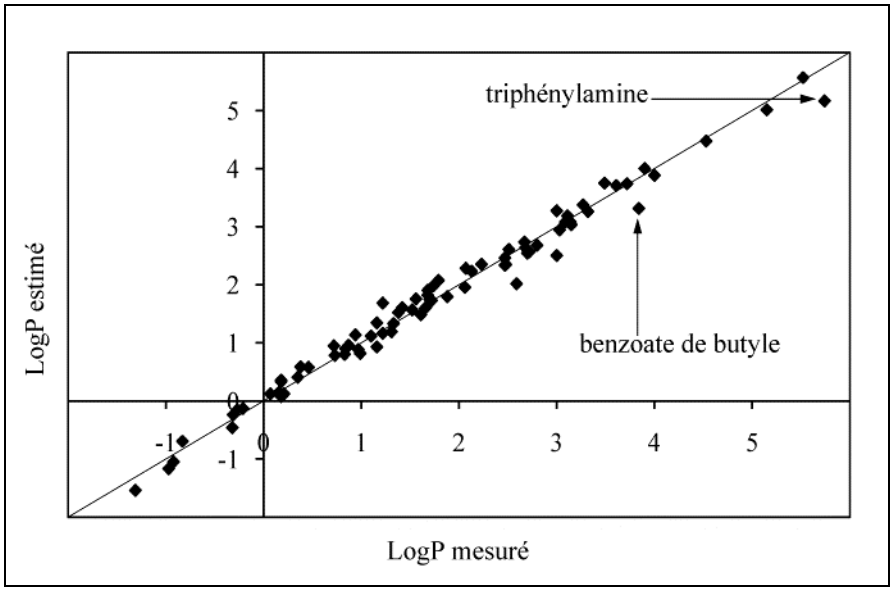


Figure 42 : Prédiction du coefficient de partage eau-octanol

L'erreur standard moyenne est à peu près égale à l'erreur expérimentale, on peut donc dire que le modèle établi est de bonne qualité et que ses prédictions sont fiables, dans son domaine d'application. Nous avons d'ailleurs constaté que les molécules dont le logP est le moins bien prédit sont les molécules les plus marginales par rapport à la base d'apprentissage. Ainsi la triphénylamine est la seule molécule comprenant trois cycles, et le benzoate de butyle est l'ester le plus lourd de la base.

I.4.3 - Prédiction de la solubilité aqueuse

La solubilité aqueuse d'une molécule est un paramètre important lors de la recherche d'un médicament, car cette solubilité définit la vitesse de dissolution du composé, ainsi que la quantité maximale pouvant être dissoute. Elle influe donc sur le devenir de la molécule dans le corps humain et sur son action pharmacologique. La solubilité aqueuse a déjà fait l'objet de modélisations, à l'aide de méthodes telles que la méthode de contribution de groupes, ou par mise en corrélation avec d'autres propriétés physico-chimiques de la molécule et/ou avec des descripteurs calculés. Nous avons comparé les résultats de modélisation obtenus par ces méthodes avec ceux donnés par les *graph machines*. Les valeurs de solubilités aqueuses sont disponibles pour 343 des molécules retenues. Ces valeurs sont comprises entre 35 mol/L et 10^{-6} mol/L, ce qui nous conduit à modéliser le logarithme de la solubilité (noté log S). Les résultats de modélisation figurent dans le Tableau 10, où ils sont comparés à ceux obtenus par régression multilinéaire (MLR) [91] et par des réseaux de neurones probabilistes (PNN) [92].

Méthode	GM	MLR [91]	PNN [92]
N_{app} / N_{test}	274 / 69	267 / 56	348 / 51
EQMA (log mol/L)	**0,20**	0,56	0,58
EQMT (log mol/L)	**0,29**	0,86	0,69

Tableau 10 : Résultats de prédiction de la solubilité aqueuse

Les prédictions sur la base de test, représentées sur la Figure 43, sont globalement très bonnes.

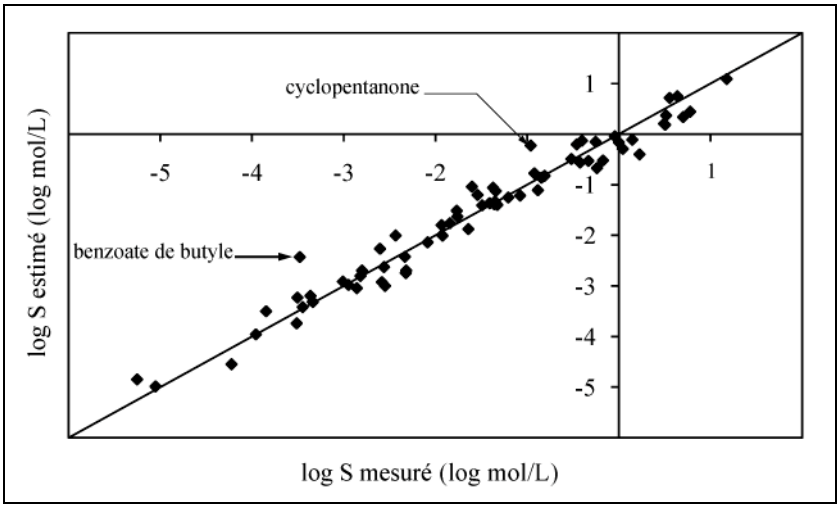


Figure 43 : Prédiction de la solubilité aqueuse (base de test)

Les erreurs les plus importantes sont réalisées pour la prédiction de la solubilité de la cyclopentanone et du benzoate de butyle. Après vérification, la valeur de solubilité fournie dans la base Physprop [84] pour la cyclopentanone est une valeur estimée, et ne peut donc pas être considérée comme une valeur de référence. On obtient une nouvelle fois une erreur assez importante pour le benzoate de butyle. Cette erreur peut être due à la présence dans la base d'apprentissage de molécules telles que le benzoate de méthyle et le benzoate d'éthyle : leurs

structures sont proches de celle du benzoate de butyle, mais leurs solubilités sont bien plus grandes. La solubilité du benzoate de butyle est par conséquent surestimée par le modèle. Cette erreur serait sans doute plus faible si la base contenait davantage d'esters.

I.4.4 - Modélisation de la pression de vapeur saturante

La pression de vapeur saturante (P_{sat}) d'un composé, qui mesure sa volatilité, est la pression partielle à laquelle sa phase gazeuse est en équilibre avec sa phase liquide ou solide. Cette pression varie en fonction de la température. Sa mesure peut être problématique, car les faibles valeurs de pression de vapeur saturante sont difficiles à évaluer, et les mesures expérimentales sont délicates pour des composés toxiques. Il est toutefois possible de calculer approximativement la pression de vapeur grâce à l'équation (57), obtenue à partir de la relation de Clapeyron :

$$\ln \frac{P_{sat}}{P_0} = \frac{M.L_V}{R}\left(\frac{1}{T_0} - \frac{1}{T}\right) \qquad (57)$$

où T_0 est la température d'ébullition à la pression P_0, M la masse molaire du composé, et L_V sa chaleur latente de vaporisation. Cette relation est cependant peu précise lorsque les hypothèses ne sont plus vérifiées, c'est-à-dire lorsque le gaz ne peut pas être considéré comme parfait et que l'enthalpie de vaporisation ne peut être considérée comme constante dans la plage de température considérée. Un modèle capable de prédire avec précision cette propriété serait une alternative à la mesure ou à une estimation trop approximative par l'équation (57).
Nous disposons des valeurs de pression de vapeur saturante pour 415 molécules, à partir desquelles nous avons formé de manière aléatoire une base d'apprentissage et une base de test de respectivement 332 et 83 exemples. Ces pressions de vapeur, exprimées en torrs, sont données pour une température de 25 °C. En raison de la grande gamme de valeurs mesurées ($2{,}7.10^{-5} \leq P_{sat} \leq 5{,}2.10^{4}$), nous avons modélisé le logarithme de la pression. L'erreur expérimentale est difficile à évaluer car les données proviennent de différentes sources [93], et les méthodes de mesure ne sont pas toutes les mêmes. Cette erreur a cependant été estimée à 0,32 (unité log) [94].
Le Tableau 11 présente une comparaison des résultats de prédiction des *graph machines* avec ceux d'autres méthodes : réseaux de neurones traditionnels (NN) et régressions multilinéaires (MLR), faisant appel à des descripteurs. Les bases étudiées ne sont pas identiques, mais elles sont composées de molécules similaires, ce qui rend possible la comparaison des résultats.

Méthode	**GM**	**NN [95]**	**MLR [94]**	**MLR [96]**
N_{app} / N_{test}	332 / 83	290 / 65	411	479
EQMA (log Torr)	**0,18**	0,19	0,331	0,534
EQMT (log Torr)	**0,28**	0,33	-	-

Tableau 11 : Comparaison des résultats de modélisation de la pression de vapeur saturante par diverses méthodes

Les *graph machines* aboutissent aux meilleurs résultats, à la fois en apprentissage et sur la base de test, et ce alors que les autres modèles nécessitent le calcul de nombreux descripteurs (200 pour la modélisation par des réseaux de neurones et jusqu'à 800 pour la régression multilinéaire).

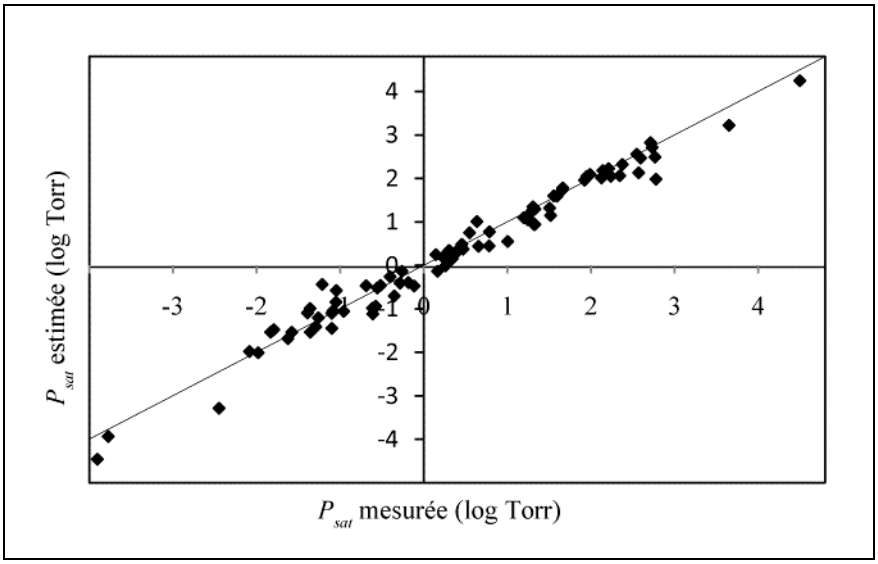


Figure 44 : Prédiction de la pression de vapeur saturante (base de test)

Les prédictions sur la base de test sont représentées sur la Figure 44. Les erreurs les plus importantes correspondent aux composés dont la pression de vapeur saturante est faible (P_{sat} < 1 Torr). Or, ces composés sont justement ceux pour lesquels l'imprécision de mesure est la plus grande. On peut donc estimer que les erreurs de prédiction sont de l'ordre de l'erreur expérimentale.

II - Prédiction d'activités moléculaires

II.1 - Toxicité de molécules diverses sur un être vivant, le *Pimephales promelas*

Le besoin de mesurer l'impact des polluants sur l'environnement est en constante augmentation. La mesure de cet impact nécessite de connaître non seulement la toxicité des produits chimiques rejetés, mais aussi celle des molécules issues de leur dégradation. La quantité de mesures à effectuer pour mesurer ces toxicités est donc importante, ce qui augmente les coûts et les délais de développement de nouveaux produits chimiques. Une alternative à la mesure systématique de la toxicité de tels composés sur des animaux est le recours à un modèle, pour prédire l'activité de molécules appartenant à une famille donnée.
Une base de données relative à la toxicité a été compilée par l'Agence des États-Unis pour la Protection de l'Environnement (EPA) [97]. Cette base, nommée ECOTOX, recense les toxicités connues de molécules diverses sur la vie aquatique, les animaux terrestres et les plantes. L'EPA a en particulier établi une base dans le but de développer un modèle capable

de prédire l'activité toxique de molécules ; elle est constituée des valeurs des toxicités de 617 composés organiques industriels sur un poisson d'eau douce, le *Pimephales promelas*, également appelé *fathead minnow* ou *tête-de-boule*.
Les valeurs de toxicité correspondent à la concentration du composé pour laquelle 50 % des animaux meurent en 96 heures ; cette toxicité est notée « 96h LC_{50} ». Les composés n'agissent pas tous selon le même mode d'action, et se différencient par les symptômes qu'ils provoquent. On distingue ainsi différentes classes, dont celle des narcotiques, qui sont les plus courants, les inhibiteurs respiratoires ou les inhibiteurs de l'acétylcholine. Deux méthodes de modélisation peuvent dès lors être mises en place. La plus directe consiste à établir un modèle global, valable pour toutes les classes de molécules. Il semble cependant plus approprié d'utiliser des modèles distincts pour modéliser des phénomènes ou des mécanismes différents. L'approche qui consiste à établir un modèle pour chaque mode d'action conduit ainsi à des résultats plus précis. Elle nécessite cependant deux étapes : dans un premier temps, chaque molécule est affectée à une classe donnée ; sa toxicité est ensuite prédite grâce au modèle développé pour cette classe. Nous avons ainsi étudié dans un premier temps la classification des molécules selon leur mode d'action, puis modélisé la toxicité des composés appartenant à la famille des narcotiques, qui sont les plus nombreux de la base.
Ces données ont fait l'objet de prédictions à l'aide d'autres méthodes, telles que la régression des moindres carrés partiels [98], le logiciel ECOSAR [99], une méthode de partition floue adaptative (AFP) [100], ou les réseaux de neurones probabilistes [101]. Les descripteurs le plus souvent retenus sont le coefficient de partage eau-octanol, le volume molaire et la polarisabilité.

II.1.1 - Classification selon le mode d'action

La base de données étudiée est composée de molécules dont les modes d'action sur le *Pimephales promelas* ont été déterminés par des experts, par observation des symptômes provoqués [102]. À chaque molécule est également associé un niveau de confiance, qui dépend du type de données disponibles pour déterminer le mode d'action. Les modes d'action attribués aux molécules peuvent donc être erronés.
Les molécules étudiées appartiennent aux classes suivantes :

- les narcotiques, qui perturbent le fonctionnement des membranes cellulaires ; la narcose est qualifiée de « toxicité de base » ;
- les découpleurs de la phosphorylation oxydative, qui empêchent le couplage entre la respiration et la synthèse d'ATP ;
- les inhibiteurs de l'acétylcholine estérase (AChE), enzyme qui régule la transmission de l'influx nerveux en assurant l'hydrolyse rapide de l'acétylcholine ;
- les inhibiteurs respiratoires ;
- les composés électrophiles, ou proélectrophiles ;
- certains composés agissent sur le système nerveux central (neurotoxiques).

Les narcotiques sont les plus nombreux de la base de données, et la toxicité des composés de cette classe est la plus couramment prédite. Nous avons donc choisi d'étudier la toxicité de ces molécules, en les séparant d'abord des composés agissant selon un autre mode d'action. Nous avons pour cela utilisé les *graph machines* en classification, c'est-à-dire en associant à chaque nœud racine un réseau de neurones dont le neurone de sortie est à fonction d'activation sigmoïde, comme décrit dans la section III.2 du chapitre 3.

II.1.1.1 - Apprentissage

La base d'apprentissage est composée de 470 molécules. Leur répartition est indiquée dans le Tableau 12.

<table>
<tr><th colspan="2">Mode d'action</th><th colspan="2">Nombre d'exemples</th></tr>
<tr><td colspan="2">Narcotiques</td><td colspan="2">254</td></tr>
<tr><td rowspan="5">Autres modes d'action</td><td>Découpleurs</td><td>18</td><td rowspan="5">216</td></tr>
<tr><td>Inhibiteurs de l'AchE</td><td>26</td></tr>
<tr><td>Inhibiteurs respiratoires</td><td>6</td></tr>
<tr><td>Réactifs électrophiles</td><td>154</td></tr>
<tr><td>Composés neurotoxiques</td><td>12</td></tr>
</table>

Tableau 12 : Répartition des molécules de la base d'apprentissage selon leur mode d'action

L'attribution des modes d'action des composés considérés comme narcotiques est globalement plus fiable que celle des autres composés (le niveau de confiance associé est meilleur).
Plusieurs apprentissages ont été réalisés, en utilisant comme fonctions de nœuds des réseaux de neurones de complexité croissante (de 2 à 5 neurones cachés). Le calcul des taux d'erreur en apprentissage et des scores de LOO virtuel a permis de sélectionner le meilleur modèle : il est obtenu avec des réseaux à 4 neurones cachés. Les résultats de ce modèle sont donnés sur la Figure 45.
Lorsque la complexité augmente, on observe une diminution du coût d'apprentissage : le modèle s'ajuste de mieux en mieux aux données d'apprentissage. Lorsque l'on passe de 4 à 5 neurones cachés, le taux d'erreur de classification des molécules narcotiques change peu, alors que celui des non-narcotiques continue à diminuer. Les données associées à ces molécules sont davantage bruitées que celles associées aux narcotiques, ce qui explique l'écart entre les taux d'erreur pour ces deux classes de molécules. En revanche, les taux d'erreurs obtenus pour ces deux classes avec un modèle à 5 neurones cachés sont comparables : il est donc possible que ce modèle soit surajusté au bruit.

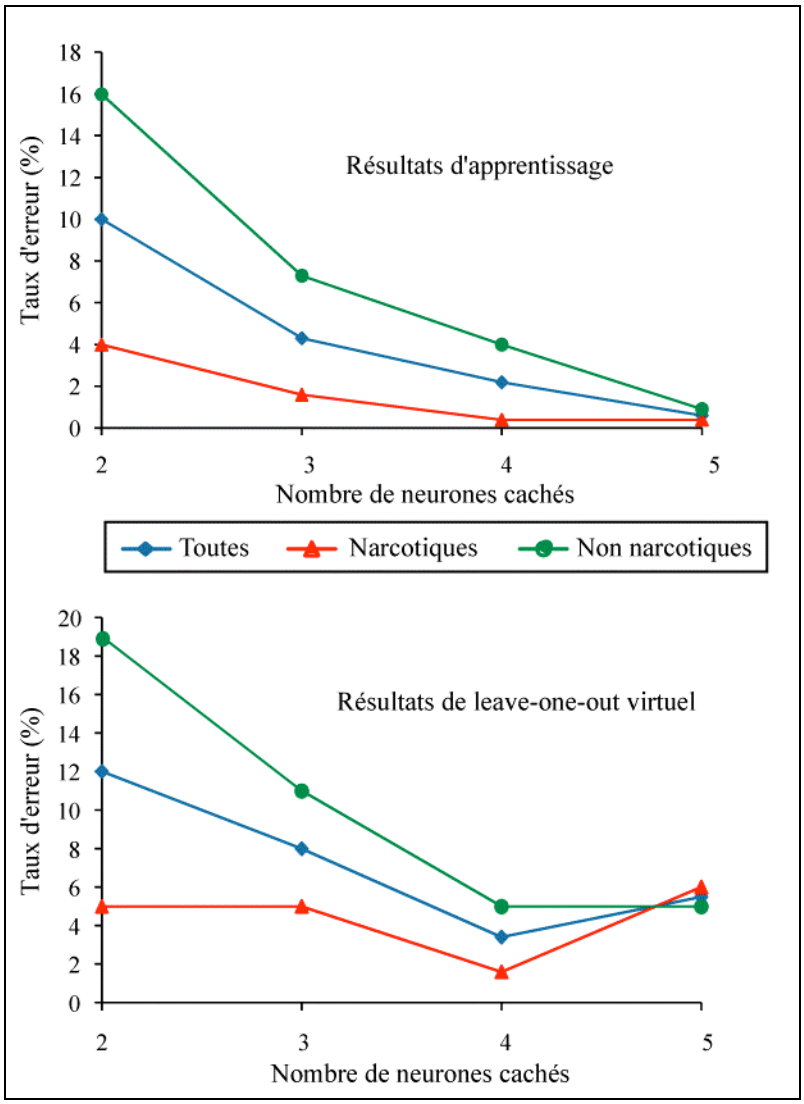


Figure 45 : Résultats d'apprentissage et de LOO virtuel pour la classification selon le mode d'action toxique

Cette hypothèse peut être vérifiée par l'analyse des résultats de leave-one-out virtuel : le taux d'erreur en classification diminue globalement lorsque la complexité du modèle augmente, jusqu'à 4 neurones cachés. En revanche, les performances en généralisation d'un modèle à 5 neurones cachés sont moins bonnes, ce qui traduit un phénomène de surapprentissage.
Nous avons donc sélectionné le modèle à 4 neurones cachés, qui présente à la fois de bonnes capacités d'apprentissage et de généralisation.

II.1.1.2 - Classification sur la base de test

Nous avons alors utilisé le modèle sélectionné pour classer 90 nouvelles molécules, constituant une base de test, et déterminer si elles appartiennent ou non à la classe des narcotiques. Le taux d'erreur s'élève à 10 %. Ces résultats sont donc comparables aux résultats d'apprentissage et de leave-one-out obtenus avec ce modèle. Les molécules mal classées sont 4 molécules neurotoxiques, et 5 non neurotoxiques. Nous avons cherché à expliquer ces erreurs, en comparant nos résultats avec ceux obtenus par une méthode fondée sur l'étude de

fragments moléculaires [102]. Le taux d'erreur de classification avec cette méthode s'élève à 14 % ; les *graph machines* présentent donc de meilleurs résultats. Nous avons également comparé les molécules pour lesquelles les prédictions étaient erronées.

- Parmi les 4 molécules narcotiques mal classées, 3 ont également été mal classées par le modèle à fragments. Le niveau de confiance associé à leur appartenance au groupe des narcotiques est « modéré » ou « bas ». Il est donc possible que ces molécules ne soient en réalité pas narcotiques.
- Une des molécules non narcotiques (le 2,2,2-trifluoroéthanol), classée à tort dans le groupe des narcotiques, est également mal classée par le modèle à fragments.
- La 2,3,5,6-tétrachloroaniline est un découpleur de la phosphorylation oxydative, mais est classée comme narcotique. Cette erreur peut être causée par la faible représentation des découpleurs dans la base d'apprentissage (moins de 4 %).
- Les 3 autres molécules mal classées sont toutes des composés électrophiles, avec un niveau de confiance « bas » ou « insuffisant » : les données associées à ces molécules sont donc peut-être erronées.

Les prédictions du modèle sur la base de test s'avèrent donc satisfaisantes, et la plupart des erreurs commises peuvent être expliquées.

II.1.2 - Modélisation de la toxicité des narcotiques

Nous avons ensuite mis en œuvre les *graph machines* pour la modélisation de la toxicité de composés de la famille des narcotiques. Le mécanisme d'action de ces molécules n'est pas encore connu de façon précise, mais les hypothèses les plus sûres suggèrent que ces molécules perturbent le fonctionnement des membranes lipidiques, ou agissent en se liant à des protéines lipidiques. La base étudiée est constituée de 311 molécules, de la famille des narcotiques, réparties en une base d'apprentissage et une base de test comprenant respectivement 208 et 103 composés. Les toxicités sont exprimées en log mmol/L. Cette base a également fait l'objet d'une modélisation par la méthode de régression des moindres carrés partiels (PLSR) [98], à partir d'un jeu de 867 descripteurs. Nous avons ainsi évalué les performances des *graph machines*, malgré l'absence de marge d'erreur sur les mesures expérimentales, par comparaison avec les performances de la méthode PLSR.

Les molécules font intervenir 8 types d'atomes (C, N, O, F, Cl, Br, P et S), mais seules deux molécules contiennent du phosphore ; elles sont donc retirées de la base. Le choix du modèle a été effectué à l'aide du calcul des coûts d'apprentissage et des scores de leave-one-out virtuel des modèles de complexités diverses (les fonctions de nœud envisagées sont des réseaux de neurones de 2 à 6 neurones cachés). Ces résultats figurent dans le Tableau 13.

Modèle	2N	3N	4N	5N	6N
EQMA (log mmol/L)	0,53	0,42	0,32	0,26	**0,20**
E_p (log mmol/L)	0,57	0,46	0,41	0,33	**0,27**

Tableau 13 : Sélection de modèle pour la prédiction de la toxicité de narcotiques

On observe une diminution du coût d'apprentissage lorsque le nombre de neurones cachés augmente. De même, le score de leave-one-out virtuel est optimum avec un modèle à 6 neurones cachés : ce modèle ne semble donc pas être surajusté aux données d'apprentissage, et semble avoir les meilleures capacités de généralisation. Nous n'avons cependant pas envisagé de modèles de complexité supérieure, pour lesquels le nombre de paramètres serait trop important par rapport au nombre d'exemples.
Les activités des 103 composés narcotiques de la base de test ont ensuite été prédites grâce au modèle sélectionné. Les résultats obtenus figurent dans le Tableau 14, où ils sont comparés à ceux obtenus par une régression des moindres carrés partiels, ainsi que sur la Figure 46.

Modèle	PLSR [98]	GM - 6N
EQMT (log mmol/L)	0,43	**0,39**
R^2	0,885	**0,900**

Tableau 14 : Résultats de prédiction de la toxicité de narcotiques sur le *Pimephales promelas* (base de test)

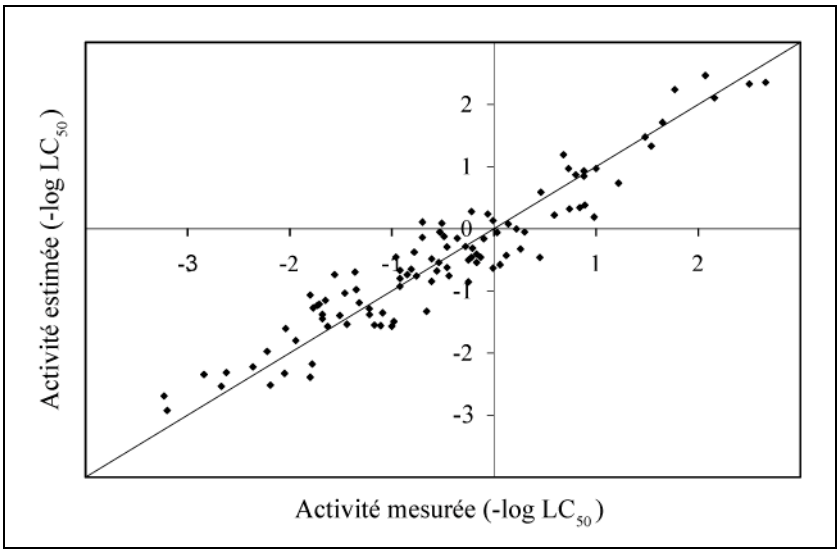


Figure 46 : Prédiction de la toxicité de narcotiques sur le *Pimephales promelas* (base de test)

Les prédictions obtenues à partir des réseaux à 6 neurones cachés sont donc comparables à ceux obtenus par régression PLS, mais sans calcul ni sélection de descripteur.
Nous avons également montré dans [52] (reproduit en annexe 2) la capacité des *graph machines* à modéliser la toxicité d'une famille de phénols sur un protozoaire cilié, le *Tetrahymena pyriformis*.

II.2 - Prédiction de l'activité agoniste de dérivés ecdystéroïdes

Les ecdystéroïdes sont des hormones responsables du déclenchement et de la régulation de la mue pour la plupart des insectes. En particulier, la 20-hydroxyecdysone, représentée en Figure 47, est la principale hormone de reproduction, de mue et de métamorphose des arthropodes. Elles sont également présentes dans les plantes (phytoecdystéroïdes), et sont de fait les stéroïdes les plus courants. Ces hormones des sont des bons points de départ pour l'élaboration de structures moléculaires qui constitueraient de bons insecticides, et ne présenteraient pas d'effets néfastes pour les plantes.

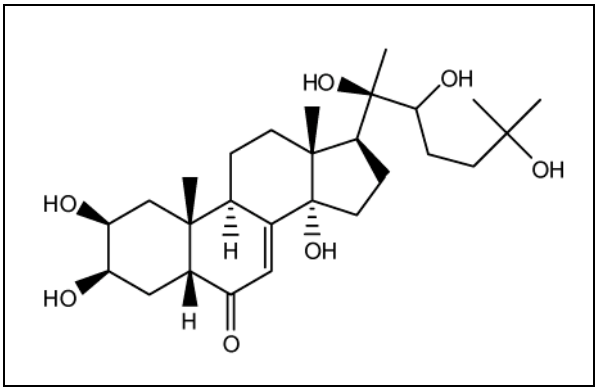


Figure 47 : 20-hydroxyecdysone

Les ecdystéroïdes agissent soit en se liant à leur récepteur (ECR), ce qui active la transcription de gènes, soit par la régulation de facteurs de transcription. Les mécanismes d'action moléculaires semblent impliquer trois sites d'interaction, mais ils ne sont pas connus précisément. Par exemple, certaines molécules, bien qu'actives, sont privées de l'un ou de l'autre de ces sites. La modélisation de l'activité d'ecdystéroïdes pourrait donc aider à déterminer quelles sont les relations entre la structure moléculaire et l'activité biologique de ces composés. De plus, le récepteur ECR, qui contrôle le développement des insectes, est une cible de choix pour de nouveaux insecticides sans effet sur les vertébrés ou les plantes, tels que les dibenzoylhydrazines. Les agonistes de la 20-hydroxyecdysone, dont nous cherchons à modéliser l'activité, seraient donc des candidats possibles, car ils seraient à même de bloquer le récepteur ECR, donc le développement des insectes.

II.2.1 - Constitution de la base de données

L'affinité des stéroïdes pour le récepteur ECR peut être mesurée sur des cellules sanguines de la drosophile (*Drosophila melanogaster* BII cell). L'absorbance à 450 nm de ces cellules est mesurée après incubation avec des solutions de stéroïdes de diverses concentrations. Cette absorbance est comparée à celle obtenue après incubation avec une solution de 20-hydroxyecdysone de concentration 10^{-7} mol/L, notée A_{max}. L'activité de chaque molécule est alors quantifiée par EC_{50} (half maximal effective concentration), qui est la concentration nécessaire pour obtenir une absorbance égale à la moitié de A_{max}. Ces valeurs d'activités sont répertoriées au sein d'une base issue du regroupement de différents travaux

[103]. Nous disposons ainsi d'une base de 88 ecdystéroïdes, répartis en une base d'apprentissage / validation et une base de test, composées respectivement de 71 et 17 composés. L'activité de ces ecdystéroïdes a déjà fait l'objet de modélisations, par des méthodes telles que l'analyse comparative des champs moléculaires (CoMFA, voir le paragraphe IV.2 du chapitre 1) [104] et l'analyse 4D-QSAR [105, 106]. Nous avons donc mesuré les performances des *graphs machines* par comparaison avec ces méthodes.

II.2.2 - Stéréoisomérie de configuration

Les ecdystéroïdes sont des composés très voisins, qui ne diffèrent que par quelques groupements, et parfois seulement par leur stéréoisomérie : ils ont alors la même formule semi-développée plane, mais des formules différentes dans l'espace à trois dimensions. Les ecdystéroïdes possèdent en particulier plusieurs atomes asymétriques, c'est-à-dire liés à 4 atomes ou groupes d'atomes différents, autour desquels ces groupes d'atomes peuvent être disposés de deux manières différentes (notées *R* et *S*). Les deux molécules qui possèdent un tel atome sont alors images l'une de l'autre dans un miroir.
La base étudiée comporte ainsi plusieurs stéréoisomères. Or, la configuration des atomes dans l'espace peut être déterminante lors de la liaison d'une molécule avec son récepteur puisque son interaction avec le site actif de ce récepteur ne peut se faire que si leurs dispositions géométriques s'accordent. Nous avons d'ailleurs réalisé des essais de modélisation sans tenir compte de l'isomérie : l'activité de deux composés isomères est alors estimée à la même valeur, moyenne de leurs deux activités. La modélisation de l'activité de ces ecdystéroïdes nécessite donc la prise en compte de l'isomérie de configuration. Nous avons mentionné dans le paragraphe I.2 du chapitre 3 qu'il est possible de caractériser la stéréoisomérie autour d'un atome par l'intermédiaire d'une étiquette. Nous avons ainsi attribué l'étiquette *R* ou *S* aux atomes asymétriques, et codé cette information en ajoutant deux entrées (*R* et *S*) à chaque réseau de neurones. L'entrée « *R* » (respectivement « *S* ») n'est reliée à la couche cachée que si l'atome correspondant est *R* (respectivement *S*).

II.2.3 - Modélisation de l'activité des ecdystéroïdes

Les fonctions de nœud des modèles envisagés sont des réseaux de neurones de 2 à 4 neurones cachés. Les exemples sont en nombre trop faible pour utiliser des réseaux à 5 neurones cachés, qui comporteraient 66 paramètres ajustables. Les coûts d'apprentissage ainsi que les scores de leave-one-out virtuel, donnés dans le Tableau 15, conduisent à choisir un modèle à 3 neurones cachés.

Modèle	2N	3N	4N
EQMA (-log mol/L)	0,72	0,48	0,20
E_p (-log mol/L)	0,82	**0,68**	0,79

Tableau 15 : Sélection de la complexité du modèle pour la modélisation de l'activité d'ecdystéroïdes

Les estimations de ce modèle pour les molécules de la base d'apprentissage sont reportées sur la Figure 48.

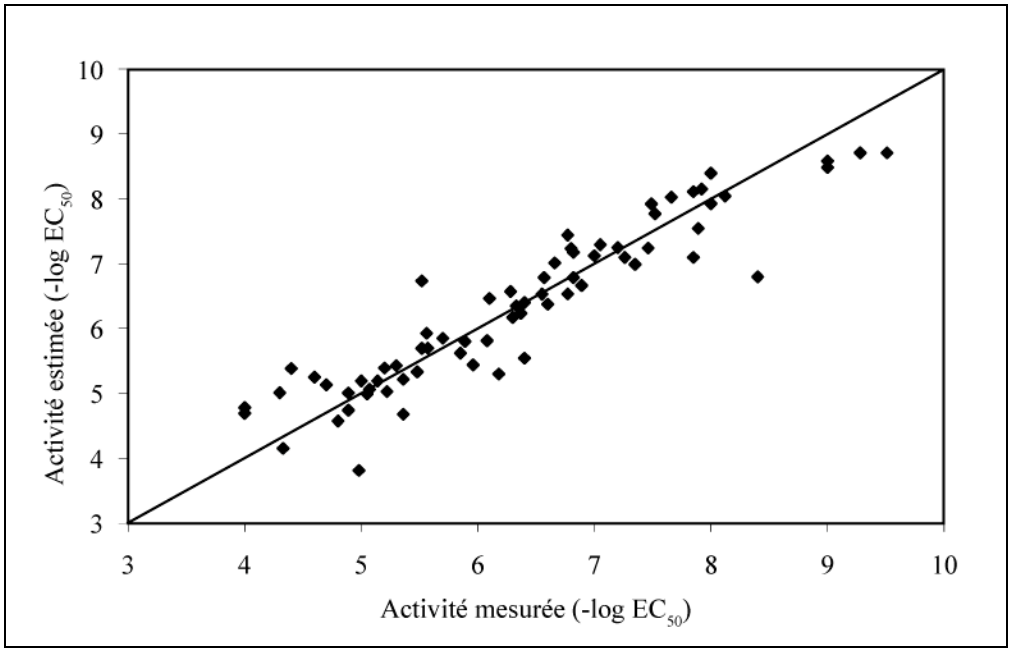


Figure 48 : Apprentissage de l'activité d'ecdystéroïdes

La base de test se compose de 17 exemples. L'erreur sur les prédictions s'élève à 0,55, ce qui constitue un résultat satisfaisant, du même ordre de grandeur que le coût d'apprentissage (voir Figure 49).

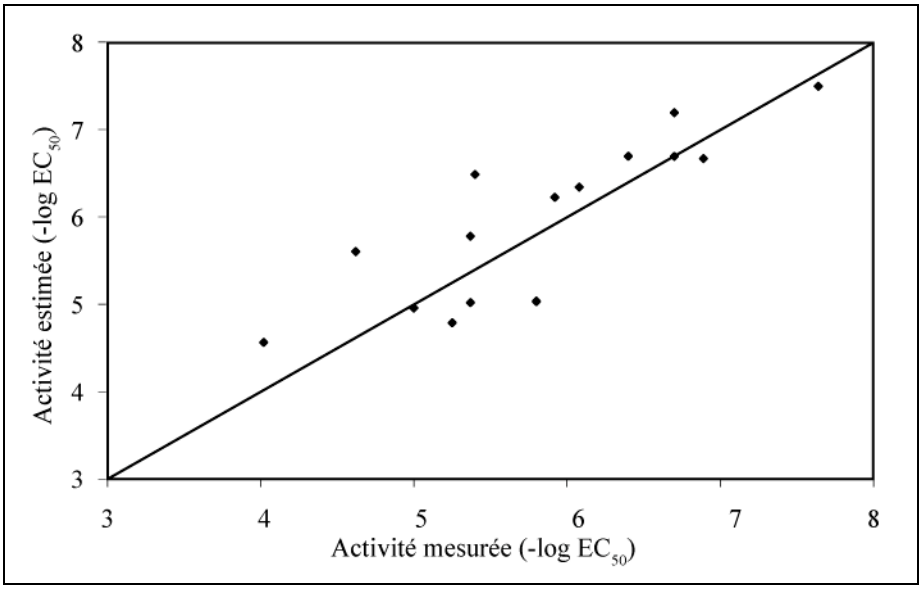


Figure 49 : Prédiction de l'activité d'ecdystéroïdes (base de test)

La comparaison des performances des *graph machines* avec celles des méthodes CoMFA et 4D-QSAR (Tableau 16), indique que les *graph machines*, bien qu'ayant de moins bons résultats d'apprentissage que la méthode CoMFA, présentent de meilleures capacités de généralisation. Le modèle obtenu avec des *graph machines* à 4 neurones cachés, bien que

fournissant de meilleures performances que le modèle CoMFA en apprentissage, a été rejeté : le score de leave-one-out virtuel indiquait en effet que ce modèle était surajusté aux données d'apprentissage. Il est d'ailleurs probable que le modèle CoMFA soit également surajusté.

Modèle	Graph machines - 3N	CoMFA [104]	4D-QSAR [105, 106]
EQMA (-log EC_{50})	**0,48**	0,37	-
EQMT (-log EC_{50})	**0,55**	1,73	1,23

Tableau 16 : Comparaison des performances de différents modèles pour la modélisation de l'activité d'ecdystéroïdes

Les précisions des mesures expérimentales ne sont pas fournies, et il n'est pas possible de comparer l'erreur du modèle au bruit des mesures. Les prédictions sont cependant de bonne qualité par rapport aux modèles déjà existants.

III - Classification

Dans la partie II.1.1 de ce chapitre, nous avons exposé comment la technique de classification peut être appliquée pour classer des molécules toxiques suivant leur mode d'action. Nous avons par ailleurs montré que les *graph machines* étaient capables de distinguer les molécules aromatiques des non-aromatiques [53]. Ce résultat met en évidence la capacité des *graph machines* à coder la structure des molécules, grâce aux étiquettes indiquant le degré de chaque nœud : l'information relative au caractère aromatique des molécules est conservée, malgré la suppression des liaisons doubles délocalisées et l'ouverture des cycles.

Il est également possible d'utiliser les *graph machines* comme un classifieur capable de déterminer si une molécule présente une activité donnée ou non. Nous avons ainsi construit un modèle permettant de classer des molécules selon leur caractère cancérigène pour des rats femelles. Les résultats obtenus sont meilleurs que ceux obtenus, sur la même base, par une méthode de *graph kernels* [107]. Ces résultats figurent dans l'article [52], reproduit dans l'annexe 2.

IV - Un exemple complexe : la prédiction de l'activité d'analogues de l'épothilone

Nous avons vu dans les paragraphes précédents que les *graph machines* peuvent être utilisées pour prédire des propriétés ou activités de molécules, à partir de bases d'exemples publiques. Nous abordons ici un problème caractérisé par le faible nombre de données disponibles, ce qui le rend particulièrement difficile : dans le cadre de notre collaboration avec le Laboratoire de Chimie Organique de l'ESPCI, nous avons contribué à la recherche de molécules anti-tumorales, analogues des épothilones (dont une des formes est représentée sur la Figure 51). Ces molécules agiraient selon le même mécanisme qu'un autre agent bien connu, le Taxol (paclitaxel), qui freine la prolifération des cellules cancéreuses en ciblant les microtubules cellulaires, nécessaires à leur mitose. Les épothilones seraient cependant plus actives, plus solubles (donc disponibles sous forme orale) et plus faciles à produire que le paclitaxel. De plus, certaines cellules ont une résistance intrinsèque ou acquise au paclitaxel, qu'elles n'auraient pas envers ces analogues.

L'activité de ces molécules peut être mesurée par leur toxicité envers différentes lignées de cellules cancéreuses. Elle est quantifiée par leur IC_{50} (half maximal inhibitory concentration), qui est la concentration nécessaire pour inhiber la prolifération cellulaire de 50 %. Ainsi, plus cette valeur est importante, moins la molécule est active. La base initiale est constituée de 81 molécules, pour lesquelles les valeurs d'activité pour les cellules cancéreuses 1A9, qui sont des lignées ovariennes humaines, sont connues. Les molécules sont composées d'atomes de carbone, azote, oxygène et soufre. Les composés halogénés, en nombre trop faible, sont exclus de la base, car ils risqueraient d'avoir une influence trop importante sur le modèle. La base d'apprentissage comporte donc finalement 63 exemples, dont l'activité est comprise entre 0,06 et 540 nmol/L [108]. La précision des valeurs expérimentales n'est pas très bonne, ce qui rend la modélisation difficile. L'activité des molécules n'a pas été mesurée sur une souche unique de cellules. Or, l'activité d'un composé donné varie avec la souche utilisée pour la mesure. Par exemple, nous disposons pour la molécule EpoB de deux mesures de l'activité : 0,13 nmol/L et 0,99 nmol/L. De plus, les marges d'erreurs expérimentales, dont nous disposons pour certaines molécules, sont parfois très importantes (incertitude de 15 à 110 %). Ces deux facteurs d'incertitude rendent donc la modélisation de l'activité des épothilones plus difficile, et sont des sources potentielles d'erreurs.

L'histogramme des valeurs de cette activité est représenté sur la Figure 50. Cet histogramme fait apparaître qu'un nombre important de molécules ont une valeur d'IC_{50} faible (plus de la moitié entre 0 et 10 nmol/L), tandis que les activités des autres molécules sont réparties sur un intervalle beaucoup plus important.

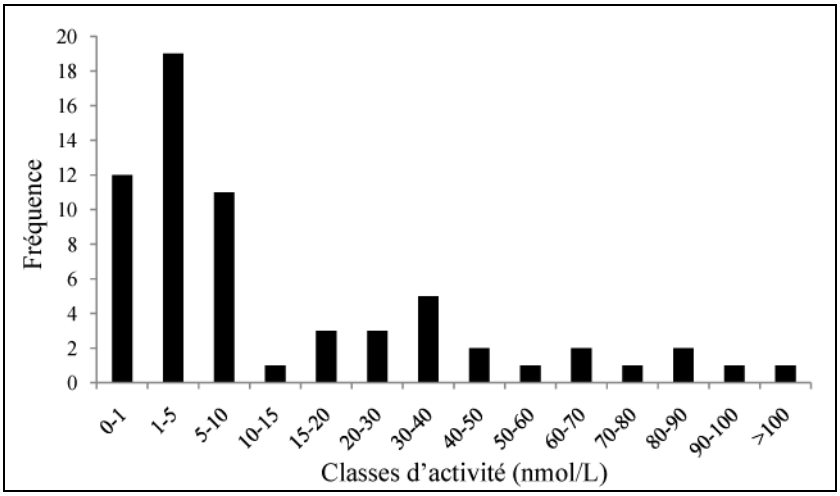


Figure 50 : Histogramme des activités de la base d'analogues de l'épothilone

Nous avons donc tenté deux approches pour la prédiction de cette activité. Nous avons d'abord modélisé le logarithme des activités : les valeurs à modéliser sont alors mieux réparties. Un modèle en deux étapes a également été envisagé, afin d'obtenir une modélisation plus précise des activités dans les zones de faible ou de forte activité. Nous avons ainsi construit un classifieur pour séparer les molécules de faible et forte IC_{50}, puis deux modèles adaptés à chacune des classes.

IV.1 - Modélisation directe de l'activité des 63 molécules

Nous avons tout d'abord modélisé le logarithme de l'activité IC_{50}, en sélectionnant le meilleur modèle à l'aide du coût d'apprentissage et du score de leave-one-out virtuel. Les fonctions de nœud envisagées sont des réseaux de neurones de 1 à 3 neurones cachés. Les performances des meilleurs modèles de chaque complexité sont données dans le Tableau 17.

Modèle	1N	2N	3N
EQMA (nmol/L)	15,3	5,5	2,2
E_p (nmol/L)	20	20,2	63

Tableau 17 : Sélection du modèle pour l'apprentissage de l'activité d'analogues de l'épothilone

Ces résultats sont peu satisfaisants, et l'on remarque que de fortes erreurs sont réalisées sur certains exemples de fort, mais également de faible IC_{50}. Le modèle manque donc de précision pour les molécules actives (de faible IC_{50}). La complexité du problème nécessiterait sans doute l'utilisation d'un modèle plus complexe. Nous disposons cependant d'un nombre d'exemples trop faible pour développer un tel modèle de manière simple. Nous avons cependant tenté de prédire les activités de 6 molécules constituant une base de test, avec un modèle à 2 neurones cachés. Les résultats sont récapitulés dans le Tableau 18.

Molécule (n°)	Activité estimée (nmol/L)	Activité mesurée (nmol/L)
19	6,4	0,1
41	18,0	0,05
58	1,4	0,1
63	2,1	8,6
69	9,7	181
71	23,1	0,5

Tableau 18 : Prédiction de l'activité d'analogues de l'épothilone - base de test

Comme les résultats d'apprentissage le laissaient prévoir, les capacités de prédiction de ce modèle sont nettement insuffisantes. Nous avons donc eu recours à un deuxième type de modèle pour améliorer ces performances.

IV.2 - Modélisation en deux étapes : classification puis régression

Une des difficultés de la modélisation provient de la dispersion des valeurs de l'activité. Un grand nombre d'entre elles se situent dans l'intervalle 0-10 nmol/L, les autres se répartissant sur une large plage de 10 à 540 nmol/L. Nous avons donc opté pour une modélisation en deux étapes. Les molécules sont dans un premier temps réparties en deux classes, les molécules actives ($IC_{50} \leq 10$ nmol/L) et les peu actives ($IC_{50} > 10$ nmol/L), et un modèle est élaboré pour chacune de ces classes. Lors de la prédiction, chaque nouvelle molécule est d'abord classée dans un des deux groupes, puis la prédiction est réalisée à l'aide du modèle correspondant.

IV.2.1 - Classification

La base de données est constituée, avec les critères précédemment précisés, de 42 molécules actives et 21 peu actives. Nous avons réalisé une première classification sur l'ensemble de la base, en utilisant comme fonctions de nœud des réseaux de neurones à 1, 2 ou 3 neurones cachés, avec une fonction de sortie sigmoïde.

Le classifieur à 1 neurone caché conduit à une seule erreur de classification (soit un taux d'erreur de 1,6 %). Il s'agit de la molécule n°21, qui est classée à tort comme active. Or, son activité est de 9 nmol/L (donc proche du seuil choisi pour séparer les deux classes de molécules), alors que sa structure est très proche de celle de la molécule EpoC, dont l'activité est de 32 nmol/L et qui est donc peu active. Les structures de ces molécules sont représentées sur la Figure 51. La ressemblance entre ces molécules peut expliquer l'erreur de classification commise pour la molécule n°21.

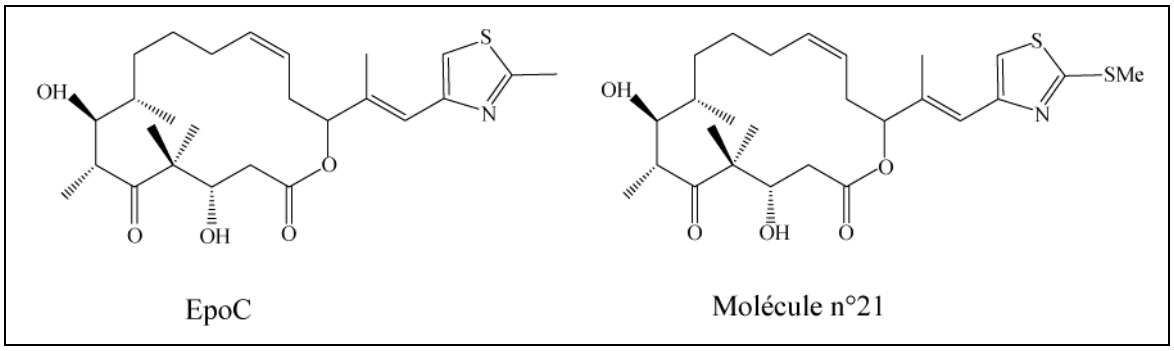


Figure 51 : Structures des molécules EpoC et n°21

Les classifieurs à 2 et 3 neurones cachés ne font par contre aucune erreur de classification en apprentissage.

Nous avons ensuite utilisé la méthode du leave-one-out afin de déterminer la complexité appropriée au problème. Les résultats obtenus avec les différents modèles, ainsi que les erreurs de classification, sont récapitulés dans le Tableau 19 (où FP désigne les faux positifs, c'est-à-dire les molécules inactives classées à tort comme actives, et FN les faux négatifs, c'est-à-dire les molécules actives classées à tort comme inactives).

Modèle	1N	2N	3N
Score de LOO (taux de classification correcte)	95,23 %	96,8 %	100 %
Erreurs en validation (n° des molécules)	FN : 25 FP : 2 -12	FN : 25 FP : 15	Aucune

Tableau 19 : Résultats de leave-one-out pour la classification des analogues de l'épothilone

Certaines molécules sont quasiment identiques à d'autres composés de la base d'apprentissage, alors que leurs activités diffèrent de façon significative, ce qui peut être à l'origine des erreurs de classification.

La molécule 25 est classée à tort dans la classe des molécules peu actives par deux des modèles. Or, son activité est égale à 10 nmol/L, donc égale au seuil. De plus, sa structure est proche de celle de nombreuses molécules inactives, ce qui peut expliquer l'erreur de classification réalisée.

Le modèle à trois neurones cachés réalise les meilleurs scores d'apprentissage et de leave-one-out : nous l'avons donc choisi pour la classification.

IV.2.2 - Modélisation pour les molécules actives

La modélisation de l'activité des molécules les plus actives est réalisée sur une base d'apprentissage de 42 exemples. Les activités sont à nouveau exprimées en nmol/L. Étant donné le faible nombre d'exemples, le risque de surajustement est grand. Nous avons calculé le score de leave-one-out pour des modèles dont la fonction de nœud est un réseau de neurones à 1, 2 ou 3 neurones cachés. Les résultats sont résumés dans le Tableau 20.

Modèle	1N	2N	3N
Score de LOO (nmol/L)	2,04	1,56	**0,83**
EQMA (nmol/L)	2,08	1,14	**0,51**

Tableau 20 : Sélection de modèle pour la prédiction de l'activité d'analogues de l'épothilone (molécules actives)

Un modèle à 3 neurones cachés paraît donc être le plus approprié pour la modélisation de l'activité des molécules actives. Un modèle à 4 neurones cachés pourrait être plus précis en apprentissage, mais le nombre d'exemples est insuffisant pour l'envisager : le nombre de paramètres serait quasiment égal au nombre d'exemples.

IV.2.3 - Modélisation pour les molécules non-actives

La même procédure ne peut pas être appliquée pour choisir le nombre de neurones cachés nécessaires à la modélisation de l'activité des 21 molécules les moins actives, car un réseau à 2 neurones cachés comporte déjà trop de paramètres (25, soit davantage que le nombre d'exemples). Il faudrait donc se limiter à des réseaux à 1 neurone caché, ce qui paraît insuffisant étant donnée la complexité de l'activité à modéliser. Cependant, cette modélisation n'est pas nécessaire car ces molécules sont non actives, et sont dès lors rejetées.

IV.2.4 - Classification et prédiction sur la base de test

Une première étape de classification a été réalisée sur les 6 molécules de la base de test, avec le modèle préalablement établi. Toutes les molécules actives ont effectivement été classées comme telles. Par contre, la molécule n°69, dont l'activité est égale à 181 nmol/L (cette molécule est donc non-active), est classée à tort comme active. Divers facteurs peuvent expliquer cette erreur : la précision de mesure sur cet exemple est très mauvaise, et la valeur de l'activité est donnée avec une marge d'erreur de 81,6 nmol/L. De plus, alors que de nombreux exemples de la base d'apprentissage ont des structures très proches, la molécule n°69 possède des particularités qui ne se retrouvent sur aucun autre exemple.
La prédiction des activités des molécules actives a ensuite été réalisée à partir du modèle à 3 neurones cachés obtenu après apprentissage sur les molécules actives. Ces prédictions sont présentées dans le Tableau 21, où elles sont comparées aux valeurs réelles ainsi qu'aux prédictions faites à partir du modèle basé sur l'ensemble des exemples (voir le paragraphe IV.1), sans classification préalable. Les marges d'erreurs sur les valeurs expérimentales sont données lorsqu'elles sont connues.

Molécule (n°)	Activité mesurée (nmol/L)	Activité estimée sans classification (nmol/L)	Activité estimée avec classification (nmol/L)
19	0,1	6,4	1,36
41	0,05	18,0	1,90
58	0,1	1,4	0,54
63	8,6	2,1	6,53
69	181 (±81,6)	9,7	1,97
71	0,5 (±0,3)	23,1	0,92

Tableau 21 : Comparaison des prédictions de l'activité d'analogues de l'épothilone sur la base de test

Le modèle construit en deux étapes – classification puis modélisation – fournit de meilleures prédictions qu'un modèle valable pour l'ensemble des exemples, comme le suggéraient les scores de leave-one-out obtenus par les deux approches.

Ces résultats pourraient être améliorés par la constitution d'une base plus homogène, et par l'ajout de nouveaux exemples dans la base d'apprentissage. Il serait alors possible, notamment lors de la classification, d'utiliser des modèles plus complexes, qui rendraient compte de manière plus précise du phénomène modélisé. Cependant, compte tenu des erreurs expérimentales, ces résultats se révèlent satisfaisants : les quatre molécules les plus actives (molécules n° 19, 41, 58 et 71) sont effectivement prédites comme les plus actives. De plus, l'erreur sur l'exemple n° 69 ne porte pas à conséquence, car le seul risque couru est la synthèse inutile de cette molécule. Le classement d'une molécule active comme non-active aurait eu davantage de conséquences, car seules les molécules actives sont retenues.

Cette collaboration devrait se poursuivre par la prédiction par les *graph machines* de l'activité de molécules proposées par les chimistes du Laboratoire de Chimie Organique, mais n'ayant pas encore été synthétisées. Ces prédictions permettraient d'éviter la synthèse de molécules inactives, de tester un nombre bien plus important de composés, et de déterminer quels types de groupements sont susceptibles d'augmenter ou de diminuer l'activité. Le problème qui reste ouvert est celui du choix des synthèses à effectuer pour enrichir la base de données de manière aussi économique que possible : c'est donc le problème de la planification expérimentale pour les *graph machines*.

CHAPITRE 5
Conclusions et perspectives

Dans cette thèse, nous avons présenté une nouvelle méthode de régression et de classification à partir de données structurées, que nous avons appelée *graph machines*, ainsi que les applications de cette méthode à la prédiction de propriétés et d'activités moléculaires.

Les techniques traditionnelles de modélisation établissent une relation entre la grandeur modélisée et un vecteur de variables qui la déterminent. Les principaux inconvénients de ces méthodes résident dans la difficulté du choix des variables pertinentes, et dans leur calcul ou leur mesure préalable. Nous avons développé une technique qui s'affranchit de ces problèmes lorsque les données sont structurées, car elle établit une relation directe entre la structure de ces données et la grandeur modélisée. L'apprentissage s'effectue donc non plus à partir de vecteurs de données, mais à partir de graphes. Notre méthode consiste à faire correspondre à chaque graphe de la base de données d'apprentissage une fonction de même structure mathématique que le graphe associé. Cette fonction est la combinaison de fonctions paramétrées identiques, obtenues en associant à chaque nœud du graphe une fonction paramétrée. La modélisation d'une propriété consiste ensuite à déterminer ces paramètres par apprentissage, à partir de la base d'apprentissage constituée de couples structures/sorties.

Les fonctions partagent toutes le même jeu de paramètres, ce qui permet d'utiliser les techniques traditionnelles d'apprentissage. Nous avons montré que les techniques habituelles de sélection de modèle peuvent également être appliquées, notamment le calcul des leviers des exemples, qui sont une mesure de l'influence de chacun d'eux sur le modèle. Ce calcul permet de détecter les catégories de molécules sous-représentées dans la base d'apprentissage, et éventuellement de modifier celles-ci pour améliorer le modèle. Les leviers permettent également d'évaluer les capacités de généralisation des modèles obtenus par apprentissage.

Ainsi, cette nouvelle approche présente l'avantage d'éviter le calcul des descripteurs moléculaires habituellement utilisés pour ce type de problème, puisqu'il suffit de connaître la structure en graphe de la molécule. De plus, contrairement aux méthodes traditionnelles de modélisation, les *graph machines* ne sont pas spécifiques à une propriété ou une activité donnée. Un aspect remarquable des *graph machines* est que la même machine peut permettre de prédire différentes propriétés, au prix seulement d'un apprentissage, et sans qu'il soit nécessaire de la reconstruire.

Nous avons appliqué les *graph machines* à la modélisation de propriétés et d'activités de molécules. Celles-ci peuvent en effet être représentées par des graphes à partir de la seule donnée de leur code SMILES. De très bons résultats ont été obtenus par l'utilisation de cette

technique sur un grand nombre de bases de données de propriétés ou d'activités. Nous avons ainsi modélisé avec succès de nombreuses propriétés physico-chimiques, telles que la température d'ébullition, le coefficient de partage eau-octanol ou la pression de vapeur saturante. Ces modélisations nous ont permis d'estimer les performances des graph machines, en les comparant à celles obtenues par les méthodes traditionnelles ; les propriétés étudiées présentent également un enjeu industriel : le devenir d'un composé dans l'eau, l'air ou le sol, ainsi que l'assimilation d'une molécule par l'organisme, sont fortement liés à ses propriétés physico-chimiques. La prédiction de ces propriétés permettrait par conséquent d'accélérer le processus de recherche de nouveaux médicaments.
Nous avons également testé l'efficacité des *graph machines* pour la prédiction d'activités très diverses, et d'enjeux de plus en plus importants. Nous avons ainsi modélisé l'activité toxique d'une famille de phénols et d'un ensemble plus varié de molécules sur deux familles de poissons. La mesure de l'impact des polluants sur l'environnement nécessite en effet de connaître non seulement la toxicité des produits chimiques rejetés, mais aussi celle des molécules issues de leur dégradation. La prédiction de la toxicité de molécules est alors une alternative à la synthèse et à la mesure systématique de la toxicité de tels composés sur des animaux. Les *graph machines* se sont également révélées performantes pour la prédiction de l'activité agoniste de dérivés ecdystéroïdes, hormones permettant l'étude des structures moléculaires qui constitueraient de bons insecticides, mais ne seraient pas néfastes aux plantes.

Perspectives

Les *graph machines* constituent une méthode de prédiction nouvelle, pour laquelle de nombreuses perspectives sont ouvertes.

Planification d'expériences

Puisque les outils classiques d'évaluation et de sélection de modèle peuvent être appliqués aux *graph machines*, nous envisageons de mettre en œuvre la planification d'expériences, qui met en jeu ces outils. Cette méthode a pour but d'optimiser le choix des expériences à sélectionner ou à réaliser, afin d'obtenir le plus d'information significative pour le modèle. Il serait alors possible de minimiser le nombre de mesures à effectuer pour compléter la base de données ; en QSAR et QSPR, le nombre de molécules à synthétiser, puis de mesures à effectuer, pourrait être réduit.

Nous avons vu que les leviers mesurent l'influence des exemples sur le modèle, et qu'un levier élevé pour une famille de molécules peut traduire un manque d'informations sur cette famille. Les leviers permettent ainsi de déterminer quels types d'exemples doivent être ajoutés à la base d'apprentissage. D'un point de vue plus global, des intervalles de confiance

(proportionnels à $\sqrt{h_{ii}}$) élevés dans une zone de l'espace des graphes peuvent traduire un manque d'informations dans cette zone. Il pourra donc être utile d'y ajouter de nouveaux exemples.

Utilisation de descripteurs

Les applications des *graph machines* à la prédiction de propriétés et d'activités de molécules que nous avons présentées se sont toutes révélées satisfaisantes, et de meilleure qualité que les modélisations obtenues par d'autres méthodes. Il semble donc que la structure des molécules soit suffisante et contienne assez d'informations pour prédire ces grandeurs. Cependant, la modélisation de propriétés ou d'activités à partir de la seule structure moléculaire demande qu'aucun critère, autre que structural, n'ait d'influence significative sur la grandeur modélisée. Or, certaines propriétés peuvent faire intervenir d'autres facteurs que la structure des molécules. La modélisation d'une grandeur qui serait fonction à la fois de la structure de la molécule et d'une ou plusieurs variables peut alors être réalisée à l'aide d'étiquettes associées au nœud racine, comme nous l'avons décrit dans le paragraphe I.2 du chapitre 3.

Lorsque la structure des molécules ne suffit pas à déterminer une propriété, il est en effet possible d'enrichir les *graph machines* par l'ajout de descripteurs, afin de fournir les informations pertinentes manquantes. Cette approche devrait par exemple permettre de prédire la température de fusion de composés organiques, qui dépend à la fois de la structure d'une molécule seule, des interactions intermoléculaires et de l'arrangement spatial des molécules dans l'état solide. La modélisation de cette propriété conduit d'ailleurs à de mauvais résultats quelle que soit la méthode adoptée, et l'erreur standard est généralement de l'ordre de 40 °C.

Considérons d'autre part la modélisation d'une propriété dont la valeur dépend de données exogènes, telles que les conditions de mesure (température, pression…). Dans la plupart des cas, ces données sont identiques sur tous les exemples de la base d'apprentissage. Ainsi, les conditions expérimentales des mesures des propriétés et des activités que nous avons modélisées étaient toutes fixées : les valeurs des températures d'ébullition ont été mesurées et prédites à la pression de 760 Torr, la toxicité des molécules sur le *Pimephales promelas* ont été évaluées après une durée d'exposition fixe de 96 heures…

Il est cependant possible qu'un ou plusieurs facteurs exogènes ne soient pas constants d'un exemple à un autre de la base d'apprentissage. Cela peut être le cas lorsque la propriété étudiée est difficile à mesurer dans les conditions fixes, et n'est connue que pour des conditions différentes pour certains composés. Il peut également arriver que l'on souhaite prédire une propriété non pas à une température ou à une pression donnée, mais sur une gamme de températures ou de pressions. Si les données sont en nombre suffisamment important, il peut même être possible de modéliser une courbe, par exemple la pression de vapeur saturante en fonction de la température. Ce type de modélisation peut être nécessaire lors de la recherche d'un composé ayant une activité précise, quand les conditions (de

température, de pression…) ne sont pas fixées par avance mais peuvent varier dans un intervalle donné. Cette application peut se révéler particulièrement intéressante dans l'industrie, lorsque les conditions expérimentales peuvent être choisies, contrairement aux conditions fixées (température, acidité…) du corps humain.

Mise en œuvre de méthodes de rééchantillonnage et de méthodes d'ensemble

Dans le présent travail, nous avons mis en œuvre la validation croisée et le leave-one-out pour l'estimation de l'erreur de généralisation. D'autres méthodes de rééchantillonnage peuvent être mises en œuvre, telles que le bootstrap.
D'autre part, les méthodes d'ensemble telles que le boosting ou le bagging peuvent être avantageusement utilisées pour améliorer les performances de classification ou de régression. De telles méthodes peuvent également s'appliquer aux *graph machines*.

De nombreux développements peuvent donc être envisagés pour cette nouvelle approche de l'apprentissage de données structurées, tant du point de vue méthodologique que du point de vue de la diversité des applications potentielles.

BIBLIOGRAPHIE

1. Crum-Brown, A., et Frazer, T. On the connection between chemical constitution and physiological action. *Transactions of the Royal Society of Edinburgh* 1868-69, **25**, p. 151-203.
2. Hansch, C., Leo, A., et Hoekmann, D. Exploring QSAR : hydrophobic, electronic and steric constants. Washington, DC : American Chemical Society, 1995.
3. Wiener, H. Structural determination of paraffin boiling points. *Journal of Chemical Information and Computer Sciences*, 1947, **69**, p. 17-20.
4. Randić, M. On characterization of molecular branching. *Journal of the American Chemical Society*, 1975, **97**, p. 6609-6614.
5. Kier, L.B., et Hall, L.H. Molecular connectivity in chemistry and drug research. New-York : Academic Press, 1976.
6. Balaban, A.T. Highly discriminating distance-based topological index. *Chemical Physics Letters*, 1982, **89**, p. 399-404.
7. Heritage, T.W., *et al.* EVA : A novel theoretical descriptor for QSAR studies. *Perspectives in Drug Discovery and Design*, 1998, **9-11** (0), p. 381-398.
8. Schuur, J.H., Selzer, P., et Gasteiger, J. The coding of the three-dimensional structure of molecules by molecular transforms and its application to structure-spectra correlations and studies of biological activity. *Journal of Chemical Information and Computer Sciences*, 1996, **36** (2), p. 334-344.
9. Jolliffe, I.T. Principal Component Analysis. New-York, NY : Springer, 2ème édition, 2002.
10. Martens, H., et Næs, T. Multivariate calibration. Chichester : Wiley, 1989.
11. Wold, H. Estimation of principal components and related models by iterative least squares, in *Multivariate Analysis*, Krishnaiaah, P.R., Editor. 1966, New York : Academic Press. p. 391-420.
12. Höskuldson, A. PLS regression methods. *Journal of Chemometrics*, 1988, **2**, p. 211-228.
13. Stoppiglia, H., *et al.* Ranking a random feature for variable and feature selection. *Journal of Machine Learning Research*, 2003, **3**, p. 1399-1414.
14. Stoppiglia, H. Méthodes statistiques de sélection de modèles neuronaux ; applications financières et bancaires [thèse en ligne]. Paris : Université Pierre et Marie Curie, 1997. Disponible sur : http://www.neurones.espci.fr.
15. Dreyfus, G., *et al.* Réseaux de neurones, méthodologie et applications. Paris : Eyrolles, 2ème édition, 2004.
16. McCulloch, W.S., et Pitts, W. A logical calculus of ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 1943, **5**, p. 115-133.
17. Geman, S., Bienenstock, E., et Doursat, R. Neural networks and the bias/variance dilemma. *Neural Computation*, 1992, **4** (1), p. 1-58.
18. Bengio, Y., et Grandvalet, Y. No unbiased estimator of the variance of K-fold cross-validation. *Journal of Machine Learning Research*, 2003, **5**, p. 1089-1105.
19. Vapnik, V.N. The nature of statistical learning theory. Springer ed, 1995.

20. Monari, G. Sélection de modèles non linéaires par leave-one-out : étude théorique et application des réseaux de neurones au procédé de soudage par points [thèse en ligne]. Paris : Université Pierre et Marie Curie (Paris 6), 1999. Disponible sur : http://www.neurones.espci.fr/Theses_PS/MONARI_G/THESE.pdf.
21. Monari, G., et Dreyfus, G. Local overfitting control via leverages. *Neural Computation*, 2002, **14**, p. 1481-1506.
22. Vapnik, V.N. Estimation of dependences based on empirical data. Nauka ed. Moscow, 1979, traduction anglaise : Springer Verlag, B., 1982.
23. Vapnik, V., et Chervonenkis, A. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 1971, **16** (2), p. 264-280.
24. Boser, B.E., Guyon, I.M., et Vapnik, V.N. A training algorithm for optimal margin classifiers. *5th Annual ACM Workshop on COLT*, 1992, Pittsburgh, PA : ACM Press, p. 144-152.
25. Kuhn, H.W., et Tucker, A.W. Nonlinear programming. *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, 1951, Berkeley : University of California Press, p. 481-492.
26. Chapelle, O.*, et al.* Choosing multiple parameters for support vector machines. *Machine Learning*, 2002, **46** (1-3), p. 131-159.
27. Burbidge, R.*, et al.* Drug design by machine learning : support vector machines for pharmaceutical data analysis. *Computational Chemistry*, 2001, **26** (1), p. 5-14.
28. Micheli, A., Portera, F., et Sperduti, A. QSAR/QSPR studies by kernel machines, recursive neural networks and their integration. *14th Italian Workshop on Neural Nets, WIRN*, 2003, Vietri sul Mare, Italy : Springer, p. 308-315.
29. Byvatov, E.*, et al.* Comparison of support vector machine and artificial neural network systems for drug/nondrug classification. *Journal of Chemical Information and Computer Sciences*, 2003, **43** (6), p. 1882-1889.
30. Muller, K.R.*, et al.* Classifying 'drug-likeness' with kernel-based learning methods. *Journal of Chemical Information and Modeling*, 2005, **45** (2), p. 249-253.
31. Yao, X.J.*, et al.* Comparative study of QSAR/QSPR correlations using support vector machines, radial basis function neural networks, and multiple linear regression. *Journal of Chemical Information and Computer Sciences*, 2004, **44** (4), p. 1257-1266.
32. Lind, P., et Maltseva, T. Support vector machines for the estimation of aqueous solubility. *Journal of Chemical Information and Computer Sciences*, 2003, **43** (6), p. 1855-1859.
33. Gärtner, T. A survey of kernels for structured data. *ACM SIGKDD Exploration Newsletter*, 2003, **5** (1), p. 49-58.
34. Mahé, P.*, et al.* Graph kernels for molecular structure-activity relationship analysis with support vector machines. *Journal of Chemical Information and Modeling*, 2005, **45** (4), p. 939-951.
35. Kashima, H., Tsuda, K., et Inokuchi, A. Marginalized kernels between labeled graphs. *Twentieth International Conference on Machine Learning*, 2003 AAAI Press, p. 321-328.
36. Jaakkola, T., Diekhans, M., et Haussler, D. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 2000, **7** (1-2), p. 95-114.
37. Leslie, C., Eskin, E., et Noble, W.W.S. The spectrum kernel : a string kernel for SVM protein classification. *Proceedings of the Pacific Symposium on Biocomputing*, 2002, p. 564-575.

38. Ding, C., et Dubchak, I. Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, 2001, **17**, p. 349-358.
39. Vert, J.-P., Saigo, H., et Akutsu, T. Local alignment kernels for biological sequences, in *Kernel Methods in Computational Biology*, Schölkopf, B., Tsuda, K., et Vert, J., Editors. 2004 MIT Press. p. 131-154.
40. Haussler, D. Convolution kernels on discrete structures. Rapport technique, UCSC-CRL-99-10, 1999, UC Santa Cruz.
41. Tsuda, K., Kin, T., et Asai, K. Marginalized kernels for biological sequences. *Bioinformatics*, 2002, **18**, p. 268-275.
42. Mahé, P.*, et al.* Extensions of marginalized graph kernels, in *Twenty-first International Conference on Machine Learning*. 2004, Banff, Alberta, Canada : ACM Press.
43. Goulon, A., Duprat, A., et Dreyfus, G. From Hopfield nets to recursive networks to graph machines : numerical machine learning for structured data. *Theoretical Computer Science*, 2005, **344** (2-3), p. 298-334.
44. Leo, A.*, et al.* Calculation of hydrophobic constant (logP) from π and f constants. *Journal of Medicinal Chemistry*, 1975, **18**, p. 865.
45. Klopman, G.*, et al.* Computer automated logP calculations based on an extended group approach. *Journal of Chemical Information and Computer Sciences*, 1994, **34** (4), p. 752-781.
46. Jalowka, J.W., et Daubert, T.E. Group contribution method to predict critical temperature and pressure of hydrocarbons. *Industrial and Engineering Chemistry Process Design and Development*, 1986, **25** (1), p. 139-142.
47. Daubert, T.E., et Bartakovits, R. Prediction of critical temperature and pressure of organic compounds by group contribution. *Industrial & Engineering Chemistry Research*, 1989, **28** (5), p. 638-641.
48. Cramer, R.D., Patterson, D.E., et Bunce, J.D. Comparative Molecular Field Analysis (CoMFA). 1. Effect of shape on binding of steroids to carrier proteins. *Journal of the American Chemical Society*, 1988, **110** (18), p. 5959.
49. Sadowski, J., et Gasteiger, J. From atoms and bonds to three-dimensional atomic coordinates : automatic model builders. *Chemical Reviews*, 1993, **93**, p. 2567-2581.
50. Kohonen, T. Self-organization and associative memory. *Springer Series in Information Sciences*. Vol. 8. Berlin : Springer Verlag, 1984.
51. Frasconi, P., Gori, M., et Sperduti, A. A general framework for adaptive processing of data structures. *IEEE Transactions on Neural Neworks*, 1998, **9** (5), p. 768-786.
52. Goulon, A.*, et al.* Predicting activities without computing descriptors : graph machines for QSAR. *SAR and QSAR in Environmental Research*, 2007, **18** (1 & 2), p. 141-153.
53. Goulon, A., Duprat, A., et Dreyfus, G. Graph Machines and their applications to computer-aided drug design : a new approach to learning from structured data, in *Lecture Notes in Computer Science*, Springer, Editor. 2006, Berlin / Heidelberg. p. 1-19 (Invited paper).
54. Berge, C. Graphes. Gauthier-Villars ed. Paris : Bordas, 3ème édition, 1983.
55. Diligenti, M.*, et al.* Adaptive graphical pattern recognition for the classification of company logos. *Pattern Recognition*, 2001, **34** (10), p. 2049-2061.
56. Gori, M., Maggini, M., et Sarti, L. A Recursive Neural Network model for processing directed acyclic graphs with labeled edges. *Proceedings of the International Joint Conference on Neural Networks*, 2003, p. 1351-1355.
57. Joachims, T. Text categorization with Support Vector Machines : learning with many relevant features. *ECML-98, 10th European Conference on Machine Learning*, 1998 Springer Verlag, Heidelberg, DE.

58. Collins, M., et Duffy, N. New ranking algorithms for parsing and tagging : kernels over discrete structures, and the voted perceptron. *40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002, Philadelphia, p. 263-270.
59. Menchetti, S.*, et al.* Wide coverage natural language processing using kernel methods and neural networks for structured data. *Pattern Recognition Letters*, 2005, **26** (12), p. 1896-1906.
60. Denoyer, L., et Gallinari, P. Bayesian network model for semi-structured document classification. 2004, **40** (5), p. 807-827.
61. Piwowarski, B., et Gallinari, P. A Machine Learning Model for Information Retrieval with Structured Documents, in *Machine Learning and Data Mining in Pattern Recognition*, Springer, Editor. 2003, Berlin / Heidelberg. p. 425-438.
62. Pollack, J. Recursive distributed representations. *Artificial Intelligence*, 1990, **46**, p. 77-106.
63. Sperduti, A. Labeling RAAM. *Connection Science*, 1994, **6** (4), p. 429-459.
64. Goller, C., et Küchler, A. Learning task-dependent distributed structure representations by backpropagation through structure. *IEEE International Conference on Neural Networks*, 1996, p. 347-352.
65. Hammer, B. On the approximation capability of recurrent neural networks. *Neurocomputing*, 2000, **31** (1-4), p. 107-123.
66. Hammer, B. Recurrent networks for structured data - A unifying approach and its properties. *Cognitive Systems Research*, 2002, **3** (2), p. 145-165.
67. Hammer, B. Learning with recurrent neural networks, in *Lecture Notes in Control and Information Sciences*. 2000, New York : Springer-Verlag.
68. Jochum, C., et Gasteiger, J. Canonical numbering and constitutional symmetry. *Journal of Chemical Information and Computer Sciences*, 1977, **17** (2), p. 113-117.
69. Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 1988, **28** (1), p. 31-36.
70. Weininger, D., Weininger, A., et Weininger, J.L. SMILES. 2. Algorithm for generation of unique SMILES notation. *Journal of Chemical Information and Computer Sciences*, 1989, **29** (2), p. 97-101.
71. Cronin, M.T.D.*, et al.* Comparative assessment of methods to develop QSARs for the prediction of the toxicity of phenols to Tetrahymena pyriformis. *Chemosphere*, 2002, **49**, p. 1201-1221.
72. NeuroOne Suite, 7.0.1.1, Netral, www.netral.com, 2007.
73. ACD/LogP, v. 10, Advanced Chemistry Development, Inc., Toronto, ON, Canada, www.acdlabs.com, 2007.
74. Meylan, W.M., et Howard, P.H. Atom/fragment contribution method for estimating octanol-water partition coefficients. *Journal of Pharmaceutical Sciences*, 1995, **84**, p. 83-92.
75. SRC KOWWIN software, SRC-LOGKOW Version 1.66, Syracuse Research Corporation, Syracuse, USA, http://www.syrres.com/eSc/kowwin.htm.
76. Hansch, C., et Leo, A.J. Substituent constants for correlation analysis in chemistry and biology. New-York, NY : Wiley, 1979.
77. Gombar, V.K. Reliable assessment of logP of compounds of pharmaceutical relevance. *SAR and QSAR in Environmental Research*, 1999, **10**, p. 371-380.
78. Devillers, J.*, et al.* Simulating lipophilicity of organic molecules with a back-propagation neural network. *Journal of Pharmaceutical Sciences*, 1998, **87** (9), p. 1086-1090.

79. Breindl, A., Beck, B., et Clark, T. Prediction of the n-octanol/water partition coefficient, logP, using a combination of semiempirical MO-calculations and a neural network. *Journal of Molecular Modeling*, 1997, **3**, p. 142-155.
80. Duprat, A.F., T. Huynh, T., et Dreyfus, G. Towards a principled methodology for neural network design and performance evaluation in QSAR ; Application to the prediction of LogP. *Journal of Chemical Information and Computer Sciences*, 1998, **38**, p. 586-594.
81. Bodor, N., Gabanyi, Z., et Wong, C.-K. A new method for the estimation of partition coefficient. *Journal of the American Chemical Society*, 1989, **111**, p. 3783-3786.
82. Cabani, S., *et al.* Group contributions to the thermodynamic properties of non-ionic organic solutes in dilute aqueous solution. *Journal of Solution Chemistry*, 1981, **10** (8), p. 563-595.
83. Bernazzani, L., *et al.* Predicting physical-chemical properties of compounds from molecular structures by recursive neural networks. *Journal of Chemical Information and Modeling*, 2006, **46** (5), p. 2030-2042.
84. Physical Properties Database [en ligne]. Syracuse Research Corporation (SRC). Disponible sur : http://esc.syrres.com/interkow/physdemo.htm.
85. Micheli, A., *et al.* A novel approach to QSPR/QSAR based on neural networks for structures, in *Soft Computing Approaches in Chemistry*, Cartwright, H. et Sztandera, L.M., Editors. 2003, Heidelberg : Springer-Verlag. p. 265-296.
86. Rücker, C., Meringer, M., et Kerber, A. QSPR using MOLGEN-QSPR : the example of haloalkane boiling points. *Journal of Chemical Information and Computer Sciences* 2004, **44**, p. 2070-2076.
87. Beilstein database : Beilstein Informationsysteme [en ligne]. GmbH, Frankfurt, Germany. Disponible sur : http://www.beilstein.com.
88. Wessel, M.D., et Jurs, P.C. Prediction of normal boiling points for a diverse set of industrially important organic compounds from molecular structure. *Journal of Chemical Information and Computer Sciences*, 1995, **35** (5), p. 841-850.
89. Stanton, D.T. Development of a quantitative structure-property relationship model for estimating normal boiling points of small multifunctional organic molecules. *Journal of Chemical Information and Computer Sciences*, 2000, **40** (1), p. 81-90.
90. Shamsipur, M., *et al.* Highly correlating distance-connectivity based topological indices. 3 : PCR and PC-ANN based prediction of the octanol-water partition coefficient of diverse organic molecules. *Internet Electronic Journal of Molecular Design*, 2005, **4** (12), p. 882-910.
91. Chen, X., *et al.* Prediction of aqueous solubility of organic compounds using a quantitative structure-property relationship. *Journal of Pharmaceutical Sciences*, 2002, **91** (8), p. 1838-1852.
92. Mosier, P.D., et Jurs, P.C. QSAR/QSPR studies using probabilistic neural networks and generalized regression neural networks. *Journal of Chemical Information and Computer Sciences*, 2002, **42** (6), p. 1460-1470.
93. Mackay, D., Shiu, W.Y., et Ma, K.C. Illustrated handbook of physical-chemical properties and environmental fate for organic chemicals. Vol. 1-4. Boca Raton, FL : Lewis Publishers, 1992.
94. Katritzky, A.R., *et al.* QSPR studies on vapor pressure, aqueous solubility, and the prediction of water-air partition coefficients. *Journal of Chemical Information and Computer Sciences*, 1998, **38**, p. 720-725.
95. McClelland, H.E., et Jurs, P.C. Quantitative structure-property relationships for the prediction of vapor pressures of organic compounds from molecular structures. *Journal of Chemical Information and Computer Sciences*, 2000, **40**, p. 967-975.

96. Liang, C.K., et Gallagher, D.A. QSPR prediction of vapor pressure from solely theoretically-derived descriptors. *Journal of Chemical Information and Computer Sciences*, 1998, **38**, p. 321-324.
97. ECOTOX User Guide : ECOTOXicology Database System Version 4.0 [en ligne]. U.S. Environmental Protection Agency (EPA), 2007. Disponible sur : http://cfpub.epa.gov/ecotox/.
98. Öberg, T. A QSAR for baseline toxicity : validation, domain of application, and prediction. *Chemical Research in Toxicology*, 2004, **17** (12), p. 1630-1637.
99. Clements, R.G., *et al.* Estimating toxicity of industrial chemicals to aquatic organisms using structure activity relationships. 1988, Office of Pollution Preventions and Toxics, US Environmental Protection Agency, Washington, DC.
100. Pintore, M., *et al.* Predicting toxicity against the fathead minnow by adaptative fuzzy partition. *QSAR & Combinatorial Science*, 2003, **22**, p. 210-219.
101. Kaiser, K.L.E., et Niculescu, S.P. Using probabilistic neural networks to model the toxicity of chemicals to the fathead minnow (Pimephales promelas) : a study based on 865 compounds. *Chemosphere*, 1999, **38** (14), p. 3231-3245.
102. Russom, C.L., *et al.* Predicting modes of toxic action from chemical structure : acute toxicity in the fathead minnow (Pimephales promelas). *Environmental Toxicology and Chemistry*, 1997, **16** (5), p. 948-967.
103. The Ecdysone Handbook 3rd edition [en ligne]. Disponible sur : http://ecdybase.org.
104. Dinan, L., Hormann, R.E., et Fujimoto, T. An extensive ecdysteroid CoMFA. *Journal of Computer-Aided Molecular Design*, 1999, **13**, p. 185-207.
105. Ravi, M., *et al.* 4D-QSAR analysis of a set of ecdysteroids and a comparison to CoMFA modeling. *Journal of Chemical Information and Computer Sciences*, 2001, **41** (6), p. 1587-1604.
106. Hormann, R.E., Dinan, L., et Whiting, P. Superimposition evaluation of ecdysteroid agonist chemotypes through multidimensional QSAR. *Journal of Computer-Aided Molecular Design*, 2003, **17**, p. 135-153.
107. Swamidass, S.J., *et al.* Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity. *Bioinformatics*, 2005, **21** (Supplement 1), p. i359-i368.
108. Nicolaou, K.C., *et al.* Molecular design and chemical synthesis of a highly potent epothilone. *ChemMedChem*, 2006, **1** (1), p. 41-44.

ANNEXES

Annexe 1 : Numérotation canonique des atomes d'une molécule

La numérotation canonique des atomes est effectuée par un algorithme inspiré de [68]. Il permet de choisir le nœud racine du graphe associé à une molécule : c'est celui qui porte le numéro 1. Cet algorithme se déroule de la façon suivante :

1. Les atomes d'hydrogène sont supprimés et la molécule est représentée par un graphe étiqueté. Soit *n* le nombre de nœuds de ce graphe.
2. Les atomes monovalents (ne pouvant former qu'une liaison simple), tels que les atomes d'halogène, sont temporairement écartés ; ils sont numérotés après les autres atomes.
3. Pour chaque atome non-monovalent *i*, l'indice suivant est calculé :

$$C_i = \max_{j=1\ldots n} L_{i,j} + val_i - d_i$$

où $L_{i,j}$ représente la distance du nœud *i* au nœud j, val_i la valence du nœud *i* et d_i son degré. Notons que bien que l'indice ne soit pas calculé pour les atomes monovalents, ces atomes sont pris en compte pour le calcul de $\max_{j=1\ldots n} L_{i,j}$ et de d_i.

Un exemple de calcul d'indices est ainsi donné sur la Figure 52, qui représente la molécule de 3-méthylbut-2-én-1-ol : les indices sont indiqués près des atomes (les lettres a-f identifient les atomes).

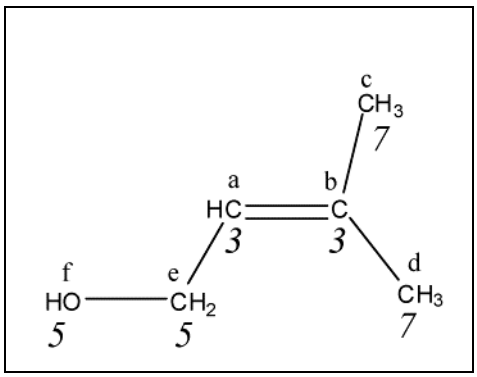


Figure 52 : Attribution d'indices aux atomes d'une molécule

4. Les atomes de même indice sont regroupés dans des classes d'équivalence, numérotées par ordre croissant des indices. Un ordre partiel est donc déjà établi.

Ainsi, pour la molécule de 3-méthylbut-2-én-1-ol, les classes d'équivalence sont les suivantes :

- classe 1 : atomes a et b ;
- classe 2 : atomes e et f ;
- classe 3 : atomes c et d.

5. Les atomes de chaque classe sont alors ordonnés séparément, en commençant par la classe 1. Soit N^i le numéro attribué au nœud associé à l'atome *i*. Les critères successivement pris en compte sont les suivants :

   A. Le numéro atomique Z :

   $$Z^i > Z^j \Rightarrow N^i < N^j$$

   B. Le nombre d'électrons libres n_{e^-} :

   $$n_{e^-}^i > n_{e^-}^j \Rightarrow N^i < N^j$$

   Si ces critères ne départagent pas deux atomes, on compare alors les atomes en position α de ces deux atomes (i.e. qui leur sont directement liés), ainsi que les liaisons avec ces atomes.

   C. Le nombre d'atomes en alpha n_α :

   $$n_\alpha^i > n_\alpha^j \Rightarrow N^i < N^j$$

   D. Le numéro atomique de ces atomes :

   $$\max_{k=1\ldots n_\alpha^i}(Z^{i,k}) > \max_{k=1\ldots n_\alpha^j}(Z^{j,k}) \Rightarrow N^i < N^j$$

   où $Z^{i,k}$ désigne le numéro atomique du *k*-ième atome en α de l'atome *i*.

   E. Le nombre d'électrons libres des atomes en alpha :

   $$\max_{k=1\ldots n_\alpha^i}\left(n_{e^-}^{i,k}\right) > \max_{k=1\ldots n_\alpha^j}\left(n_{e^-}^{j,k}\right) \Rightarrow N^i < N^j$$

   F. L'ordre des liaisons avec les atomes en position α. Soit $o(i,k)$ l'ordre de la liaison entre les atomes *i* et *k* (ordre 1 pour une liaison simple, 2 pour une liaison double…).

   $$\max_{k=1\ldots n_\alpha^i}\left(o(i,k)\right) > \max_{k=1\ldots n_\alpha^j}\left(o(j,k)\right) \Rightarrow N^i < N^j$$

   Si ces deux maxima sont égaux, on compare le numéro atomique des atomes formant ces liaisons.

   Puis, si les atomes *i* et *j* ne sont toujours pas départagés, on compare les atomes en positions β (c'est-à-dire séparés de l'atome étudié par deux liaisons), avec les critères C à F.

6. Enfin, les atomes monovalents, provisoirement écartés, sont numérotés à la suite des autres atomes, en suivant les règles 3 à 6.

Reprenons l'exemple de la Figure 52, et numérotons les atomes de cette molécule.

Atomes de la classe 1 : le critère C, appliqué aux atomes en α, départage les atomes a et b. En effet $n_\alpha(\mathrm{b}) = 3$ et $n_\alpha(\mathrm{a}) = 2$.

Atomes de la classe 2 : le critère A suffit à départager les atomes e et f, car $Z(\mathrm{O}) = 8$ tandis que $Z(\mathrm{C}) = 6$.

Atomes de la classe 3 : aucun des critères ne permet de les départager, les numéros sont attribués au hasard.

On obtient donc la numérotation suivante :

$N(\mathrm{b}) = 1$; $N(\mathrm{a}) = 2$; $N(\mathrm{f}) = 3$; $N(\mathrm{e}) = 4$; $N(\mathrm{c}) = 5$; $N(\mathrm{d}) = 6$.

Le nœud correspondant à l'atome b est donc choisi comme nœud racine de l'arbre associé à cette molécule.

Annexe 2 : Reproduction des publications

1. Goulon, A., Duprat, A., et Dreyfus, G. From Hopfield nets to recursive networks to graph machines: numerical machine learning for structured data. *Theoretical Computer Science*, 2005, **344** (2-3), p. 298-334.
2. Goulon, A., Duprat, A., et Dreyfus, G. Learning numbers from graphs. *International Symposium on Applied Stochastic Models and Data Analysis (ASMDA)*, 2005, Brest (France).
3. Goulon, A., Duprat, A., et Dreyfus, G. Graph Machines and their applications to computer-aided drug design: a new approach to learning from structured data, in *Lecture Notes in Computer Science*, Springer, Editor. 2006, Berlin / Heidelberg. p. 1-19 (Invited paper).
4. Goulon, A., Picot, T., Duprat, A. et Dreyfus, G. Predicting activities without computing descriptors: graph machines for QSAR. *SAR and QSAR in Environmental Research*, 2007, **18** (1 & 2), p. 141-153.