Statistique en grande dimension

Projet 2014–2015

Aurélien Canet et Ulysse Herbach

26 février 2015

Article de référence :

[Mahé et al., 2005] *Graph Kernels for Molecular Structure-Activity Relationship Analysis with Support Vector Machines*, J. Chem. Inf. Model. 45, 939–951.

1 Introduction

Les processus d'élaboration de nouveaux médicaments ou de nouveaux composés chimiques sont en général extrêmement coûteux en temps et en argent. Dans ce contexte, il est très intéressant de pouvoir prédire *l'activité* d'une molécule, c'est-à-dire une certaine propriété chimique ou biologique d'intérêt (capacité de se lier à certains sites, toxicité, mutagénicité...), sans être contraint de réaliser toute une série de tests.

Pour répondre à ce problème, de nombreuses méthodes de *machine learning* ont été développées pour détecter puis utiliser d'éventuels liens entre l'activité des molécules et leur structure chimique : on parle d'analyse SAR (*Structure-Activity Relationship*).

Outre le choix d'une méthode statistique adaptée, la question de la représentation des composés chimiques est essentielle. Il est possible de décrire les molécules à travers certaines de leurs propriétés physico-chimiques (caractère aromatique, électronégativité...), mais le choix de ces descripteurs est difficile. On peut aussi s'intéresser plus directement à la structure spatiale des molécules : c'est l'approche utilisée dans l'article.

Plus précisément, il s'agit de prédire le caractère mutagène ou non d'un ensemble de molécules, à partir de leur représentation en 2D par des graphes labellisés. Une fois ce choix fait, l'idée générale est d'obtenir une seconde représentation, vectorielle cette fois, afin de pouvoir utiliser avec bonheur tout l'éventail statistique moderne. Là encore, ce changement de représentation n'est pas simple et il existe plusieurs manières d'aborder le problème.

L'article étudié se concentre sur une méthode de classification utilisée dans de nombreux domaines, la *Support Vector Machine* (SVM). Cette méthode présente le grand avantage de ne dépendre des données que via leurs produits scalaires : on va voir que ceci permet de se passer d'une description vectorielle explicite des molécules, en faisant apparaître un *noyau défini positif* qui mesure directement les similarités entre les graphes.

Le noyau utilisé est celui introduit par [Kashima et al., 2003], basé sur des marches aléatoires le long des graphes. L'article propose deux améliorations de ce noyau. La première utilise la notion de *Morgan index* afin de mieux représenter les spécificités des molécules sur les graphes, la seconde tente de prévenir les trajectoires "titubantes" (i.e. qui bouclent sur elles-mêmes), peu représentatives de la structure réelle des molécules.

2 *Support Vector Machine* et lien avec les noyaux

2.1 Le principe de SVM

Dans cette partie, nous présentons d'abord brièvement le principe de la méthode SVM pour la classification supervisée binaire. Étant donné $(\mathcal{H}, \langle\cdot,\cdot\rangle)$ un espace de Hilbert, on considère un *ensemble d'entraînement* fini :

$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

où les x_j sont des éléments de $\mathcal{H}$ et les y_j des variables binaires, par exemple $y_j \in \{-1, 1\}$. Dans notre cas, 1 et -1 correspondent respectivement à *mutagène* et *non mutagène*. On souhaite alors "apprendre" une fonction capable de séparer les vecteurs associés à 1 de ceux associés à -1.

Si l'on suppose les données linéairement séparables, i.e. s'il est possible de trouver un hyperplan (affine) de $\mathcal{H}$ d'équation $\langle w, x\rangle + b = 0$ qui sépare $\mathcal{D}$ en deux selon la valeur de y_j, alors pour tout nouveau vecteur x_{new}, on prédit la valeur de y_{new} par la formule :

$$\hat{y}_{new} = sign(\langle w, x_{new}\rangle + b). \tag{1}$$

L'algorithme SVM consiste dans ce cas à trouver, parmi l'ensemble (infini) des hyperplans qui séparent $\mathcal{D}$, celui possédant la plus grande *marge*, notée γ et définie comme la plus petite distance des x_j à cet hyperplan. Intuitivement, il s'agit de l'hyperplan qui sépare le mieux les deux classes en vue de futures prédictions (cf. figure 1).

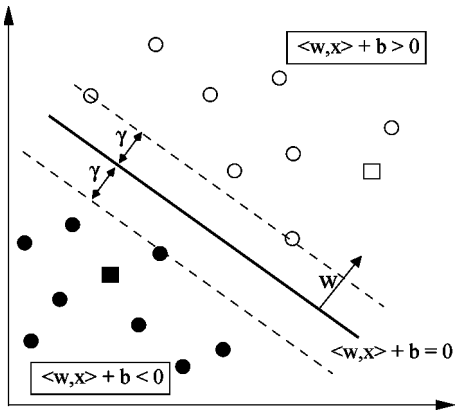

FIGURE 1 – Idée générale de la SVM (image tirée de [Mahé et al., 2005])

Le calcul de la marge conduit alors à $\gamma = \frac{2}{\|w\|}$ où $\|\cdot\|$ est la norme associée à $\langle\cdot,\cdot\rangle$, et on peut formuler SVM comme un problème d'optimisation sous contrainte :

$$\min_{w,b}\left(\frac{\|w\|^2}{2}\right) \quad \text{s.c.} \quad \forall i \in \{1,\ldots,n\},\ \ y_i(\langle w, x_i\rangle + b) - 1 \geqslant 0.$$

Dans le cas où les données de l'ensemble d'entraînement ne sont pas linéairement séparables, cette technique conserve tout son intérêt si l'on est prêt à accepter un biais. On introduit donc des variables de relaxation $\xi_i \in \mathbb{R}$ pour $i \in \{1,\ldots,n\}$, et le problème d'optimisation peut être formulé de la manière suivante :

$$\min_{w,b,\xi}\left(\frac{\|w\|^2}{2} + C\sum_{i=1}^{n}\xi_i\right) \quad \text{s.c.} \quad \forall i \in \{1,\ldots,n\},\ y_i(\langle w, x_i\rangle + b) \geqslant 1 - \xi_i \ \text{ et } \ \xi_i \geqslant 0, \tag{2}$$

où $C > 0$ est lié au compromis acceptable : un C petit signifie un grand compromis et $C = +\infty$ signifie qu'on n'accepte aucune erreur sur l'ensemble d'entraînement. Pour le résoudre, on est amené à résoudre le problème dual :

$$\max_{\alpha \in \mathbb{R}^n} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \right) \quad \text{s.c.} \quad \begin{cases} \forall i \in \{1, \dots, n\}, \ \ 0 \leqslant \alpha_i \leqslant C \\ \text{et} \ \ \sum_{i=1}^{n} \alpha_i y_i = 0. \end{cases} \tag{3}$$

De plus, lorsque la valeur optimale α^* est atteinte, elle vérifie $w = \sum_{i=1}^{n} \alpha_i^* y_i x_i$ et la prédiction se fait alors selon la valeur de la *fonction de décision* f définie par :

$$f(x) = sign \left(\sum_{i=1}^{n} \alpha_i^* \langle x, x_i \rangle + b^* \right)$$

où b^* s'obtient à partir de α^* et des produits scalaires $\langle x_i, x_j \rangle$. En pratique, il s'agit d'un problème classique d'optimisation convexe non lisse pour lequel on dispose d'un algorithme de résolution dont SVM est un cas particulier. En outre, de nombreux *packages* l'ont déjà implémenté, comme `kernlab` pour R que nous utiliserons.

2.2 Le *kernel trick*

A priori, nous ne sommes pas dans le cadre d'application de SVM puisque nos molécules appartiennent à l'ensemble des graphes labellisés, qui n'est pas un espace vectoriel. De manière plus générale, si les x_i appartiennent à un ensemble $\mathcal{X}$ quelconque, et si $\Phi : \mathcal{X} \to \mathcal{H}$ est une application qui fait le lien entre $\mathcal{X}$ et la représentation vectorielle $\mathcal{H}$, on souhaite plutôt effectuer la SVM sur les $\Phi(x_i)$.

Bien qu'il soit possible de prendre $\mathcal{H} = \mathbb{R}^d$, ce qui revient à considérer un certain nombre de propriétés des molécules comme dans l'introduction, on souhaite être bien plus précis et traduire de manière la plus exhaustive possible les graphes en vecteurs : on voit tout de suite que seul un $\mathcal{H}$ de dimension infinie pourra convenir, et on pressent que les éventuels calculs explicites sur cet espace vont être difficiles voire impossibles à implémenter.

Cependant, on remarque deux faits importants sur la méthode SVM :

- d'une part, comme les y_i valent -1 ou 1, la formulation (3) montre que le calcul de α^* (et a fortiori celui de f) ne nécessite que la connaissance des produits scalaires $\langle x_i, x_j \rangle$, qui correspondent aux distances entre les points de l'ensemble d'entraînement ;
- d'autre part, étant donnée une nouvelle molécule x_{new}, la prédiction de sa classe y_{new} ne nécessite que les produits scalaires $\langle x_{new}, x_i \rangle$.

C'est grâce à ce constat qu'on va pouvoir effectuer le *kernel trick*, c'est-à-dire s'affranchir des calculs explicites sur $\mathcal{H}$ en remplaçant les produits scalaires par des noyaux. Tout d'abord, il convient de préciser la définition de noyau qu'on utilisera dans toute la suite.

Définition. Une application $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ est appelée *noyau défini positif* sur $\mathcal{X}$ si :
(1) $\forall (x, y) \in \mathcal{X}^2$, $k(x, y) = k(y, x)$, i.e. k est *symétrique* ;
(2) $\forall n \in \mathbb{N}$, $\forall (x_1, \dots, x_n) \in \mathcal{X}^n$, $\forall (a_1, \dots, a_n) \in \mathbb{R}^n$, $\sum_{i,j=1}^{n} a_i a_j k(x_i, x_j) \geqslant 0$.

Remarque. Cette définition ne correspond pas exactement à celle utilisée dans le cadre des applications bilinéaires symétriques (on parlerait plutôt de noyau *semi-défini positif*), mais c'est de loin la plus répandue dans la littérature sur le sujet [Cuturi, 2010].

On pose alors pour tout $(x, x') \in \mathcal{X}^2$, $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$. Alors k est bien un noyau défini positif par définition du produit scalaire, et le problème dual (3) s'écrit :

$$\max_{\alpha \in \mathbb{R}^n} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \right) \quad \text{s.c.} \quad \begin{cases} \forall i \in \{1, \dots, n\}, \ \ 0 \leqslant \alpha_i \leqslant C \\ \text{et} \ \ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases} \tag{4}$$

et la fonction de décision devient

$$f(x) = sign \left(\sum_{i=1}^{n} \alpha_i^* k(x, x_i) + b^* \right).$$

Remarque. En fait, un résultat classique établit la réciproque, au sens où n'importe quel noyau défini positif sur $\mathcal{X}$ peut être représenté via une certaine fonction Φ et un produit scalaire sur un certain espace vectoriel $\mathcal{H}$. Un noyau entre deux objets correspond donc toujours à une mesure de colinéarité entre ces objets vus dans une certaine représentation vectorielle : on peut ainsi le voir comme une certaine mesure de similarité.

3 Détermination des noyaux

Dans cette section, on détaille la manière de générer des graphes qui représentent les molécules de manière pertinente, et surtout le noyau de deux graphes pour mesurer effectivement leurs similarités. Un graphe est noté $G = (V, E)$ où V est l'ensemble des sommets et E l'ensemble des arrêtes, et ses labels sont représentés par une fonction $l : V \cup E \to A$, qui à un sommet ou une arrête x associe un label $l(x)$ pris dans un alphabet A qui représente tous les atomes et toutes les liaisons possibles des molécules. Par exemple :

$$A = \{-, =, \equiv, C, O, H, \ \dots \}$$

Ainsi, V correspond à l'ensemble des atomes de la molécule et E à l'ensemble des liaisons entre ces atomes. Notons qu'ici on considère des graphes orientés.

D'autre part, on note $V^* = \cup_{n=1}^{+\infty} V^n$ l'ensemble des séquences finies de sommets. Une *trajectoire* est alors un élément $h = v_1 \dots v_n \in V^*$ qui vérifie $(v_i, v_{i+1}) \in E$ pour tout $i \in \{1, \dots, n-1\}$. On note $H(G) \subset V^*$ l'ensemble des trajectoires du graphe G. On étend alors l à $H(G)$ en posant $l(h) = (l(v_1), l(v_1, v_2), l(v_2), \dots, l(v_{n-1}, v_n), l(v_n)) \in A^{2n-1}$.

Enfin, on note $\mathcal{S}(A) \subset A^*$ l'ensemble des *fragments linéaires* de molécules, c'est-à-dire les séquences finies d'atomes/liaisons qui existent bien chimiquement sur des molécules. Cet ensemble infini est complexe à décrire en pratique (et on va justement s'en passer), mais par exemple il est clair que $H - C = O \in \mathcal{S}(A)$ tandis que $H - -CC = O \notin \mathcal{S}(A)$.

Remarque. D'un point de vue mathématique, il paraît quand même utile de préciser la chose suivante : l'ensemble $\mathcal{S}(A)$ est dénombrable puisque

$$\mathcal{S}(A) \subset A^* = \bigcup_{n=1}^{+\infty} A^n,$$

qui est dénombrable en tant qu'union dénombrable d'ensembles finis. On va voir juste après que notre espace vectoriel $\mathcal{H}$ sera défini de telle sorte que $\mathcal{H} \subset \mathbb{R}^{\mathcal{S}(A)} \simeq \mathbb{R}^{\mathbb{N}}$. Plus précisément, on pourra même prendre :

$$\mathcal{H} = \left\{ (u_n) \in \mathbb{R}^{\mathbb{N}} \mid \sum u_n^2 < +\infty \right\},$$

qui une fois muni de la norme $\| \cdot \|_2$ est un espace de Hilbert séparable. C'est bien pratique (voire indispensable) si l'on envisage sérieusement de prouver ce qui suit.

3.1 Noyaux marginalisés

Comme on l'a dit plus haut, l'approche retenue est celle de [Kashima et al., 2003], qui considère les trajectoires comme issues de marches aléatoires sur les graphes. Plus précisément, pour deux graphes G_1 et G_2, le noyau que l'on va utiliser s'écrit

$$K(G_1, G_2) = \sum_{(h_1,h_2)\in V_1^* \times V_2^*} p_{G_1}(h_1) p_{G_2}(h_2) \delta(l(h_1), l(h_2)) \tag{5}$$

où $\delta(l_1, l_2)$ est la fonction de Dirac sur $A \times A$ qui vaut 1 si $l_1 = l_2$ et 0 sinon, et p_G est la loi sur V^* d'une "marche aléatoire stoppée", c'est-à-dire :

$$p_G(v_1 \dots v_n) = p_s(v_1) \prod_{i=2}^{n} p_t(v_i \mid v_{i-1}) \tag{6}$$

avec p_s et p_t définies par

$$p_s(v) = p_0(v) p_q \quad \text{et} \quad p_t(u \mid v) = (1 - p_q) p_a(u|v)$$

où p_0 est la loi d'entrée de la marche, p_a est la probabilité de transition d'une chaîne de Markov classique, et $0 < p_q < 1$ est la probabilité d'arrêt de la trajectoire. Finalement, en posant, pour tout $s \in \mathcal{S}(A)$,

$$\phi_s(G) = \sum_{h\in H(G)} p_G(h) \delta(l(h), s)$$

On obtient l'écriture simplifiée du noyau :

$$K(G_1, G_2) = \sum_{s\in\mathcal{S}(A)} \phi_s(G_1) \phi_s(G_2). \tag{7}$$

On vérifie aisément que ceci correspond bien à définir $\mathcal{H} = \mathbb{R}^{\mathbb{N}}$ comme dans la remarque précédente et à considérer le produit scalaire usuel $\langle u, v\rangle = \sum u_n v_n$. L'avantage est que toutes les coordonnées correspondant à une séquence d'atomes/liaisons absente de la molécule sont automatiquement nulles.

L'avantage de cette approche probabiliste est d'offrir une certaine flexibilité dans la comparaison des fragments entre deux molécules : on espère ainsi détecter une grande gamme de similarités entre molécules, même les plus difficiles à repérer.

3.2 Calcul pratique des noyaux

Bien entendu, l'écriture simplifiée (7) du noyau ne permet pas un calcul effectif sans connaître explicitement $\mathcal{S}(A)$. C'est là que le *kernel trick* prend toute son importance : on va utiliser une astuce pour ne calculer que des puissances de matrices.

On introduit d'abord le *graphe produit* de deux graphes $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$ comme le graphe $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ dont les sommets $\mathcal{V} \subset V_1 \times V_2$ sont les paires de sommets de même label, avec des arrêtes reliant ces sommets si et seulement si les labels des arrêtes associées dans les deux graphes sont aussi les mêmes (on affecte alors ce label commun à l'arrête du graphe produit). On définit ensuite l'application $\pi : H(\mathcal{G}) \to \mathbb{R}$ par

$$\pi((u_1, v_1)(u_2, v_2) \cdots (u_n, v_n)) = \pi_s(u_1, v_1) \prod_{i=2}^{n} \pi_t((u_i, v_i) \mid (u_{i-1}, v_{i-1})),$$

où $\pi_s(u_1, u_2) = p_s^{(1)}(u_1) p_s^{(2)}(u_2)$ et $\pi_t((v_1, v_2) \mid (u_1, u_2)) = p_t^{(1)}(v_1 \mid u_1) p_t^{(2)}(v_2 \mid u_2)$.

Par construction, on sait qu'il y a bijection entre $H(\mathcal{G})$ et $H(G_1) \cap H(G_2)$ et que le noyau K s'écrit

$$K(G_1, G_2) = \sum_{h \in \mathcal{G}} \pi(h).$$

Tout se passe alors comme si on considérait des marches aléatoires sur le graphe produit, pour la matrice de transition $\Pi_t = (\pi_t(v|u))_{(u,v)\in\mathcal{V}^2}$ qui est tout à fait calculable explicitement. On obtient finalement :

$$K(G_1, G_2) = \sum_{n=1}^{+\infty} \sum_{\substack{h \in H(\mathcal{G}) \\ |h|=n}} \pi(h) = \pi_s^\top (I - \Pi_t)^{-1} \mathbf{1} \tag{8}$$

où $|h|$ est la longueur de la trajectoire h et $\mathbf{1}$ est le vecteur $(1, \ldots, 1)^\top \in \mathbb{R}^{|\mathcal{V}|}$.

On s'est donc ramené à une simple inversion de matrice. Comme elle est typiquement mal conditionnée (ce serait trop simple...), on l'estime de manière classique par

$$(I - \Pi_t)^{-1} \approx \sum_{n=1}^{N} \Pi_t^n,$$

où N est choisi de manière à vérifier un critère de convergence de la série (fixé à 10^{-3} dans la suite). On dispose à présent d'un moyen de calculer efficacement tous les noyaux $K(x, x')$ pour tout $x, x' \in \mathcal{X}$.

4 Une amélioration : le *Morgan index*

Le calcul effectif du noyau fait apparaître un problème important en pratique : lorsque beaucoup de sommets ont les mêmes labels, le graphe produit peut être très grand, ce qui induit un temps de calcul prohibitif si les molécules sont de grande taille. De plus, il semble un peu trop naïf de ne considérer que les labels des trajectoires communes, sans tenir compte des atomes voisins "aperçus" le long de ces trajectoires.

Un indice introduit par [Morgan, 1965] permet d'apporter une réponse à ces deux problèmes en rajoutant aux labels du graphe une information chimiquement pertinente relative au voisinage de chaque sommet. Pour $i \in \mathbb{N}^*$ fixé, le calcul du *Morgan index* d'ordre i d'un sommet donné consiste à compter le nombre de chemins de longueur i issus de ce sommet, pondérés par les *Morgan Index* d'ordre $i-1$ des sommets visités. Plus précisément, on suit la procédure itérative suivante.

- Pour $i = 0$, on initialise le *Morgan index* à 1 sur chaque sommet, i.e. $M_0(j) = 1, \forall j$.
- Pour $i \geqslant 0$, on calcule $M_{i+1} = AM_i$, où A est la matrice de voisinage définie par

$$\begin{cases} A(i,k) = 0 \text{ pour tout } k\,; \\ A(j,k) = 1 \text{ si } i \neq j \text{ et les sommets } j \text{ et } k \text{ sont liés par une arête}\,; \\ A(j,k) = 0 \text{ si } i \neq j \text{ et les sommets } j \text{ et } k \text{ ne sont pas liés par une arête.} \end{cases}$$

La figure 2 montre un exemple plus parlant. L'utilisation du *Morgan index* présente finalement un double intérêt :

— elle affine la représentation des molécules, ce qui rend les similarités détectées entre les molécules moins nombreuses mais plus pertinentes chimiquement ;

— en diminuant le nombre de similarités, on réduit la dimension du graphe produit, ce qui entraîne une réduction significative du temps de calcul du noyau.

Itérer le *Morgan index* permet donc de représenter plus fidèlement les propriétés des molécules et de gagner en temps de calcul. Cependant, si on effectue un nombre trop important d'itérations, les graphes labellisés deviennent trop spécifiques et il devient alors très difficile de détecter des similarités. On voit ainsi apparaître la nécessité d'un compromis dans le choix du nombre d'itérations.

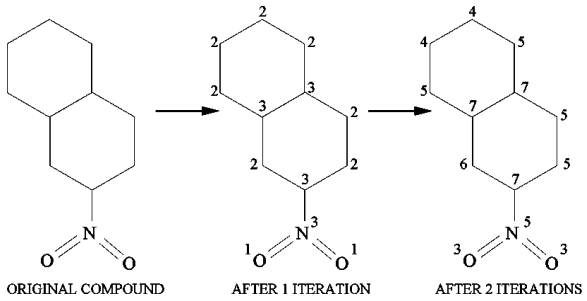

FIGURE 2 – Illustration sur un exemple du procédé d'itération du *Morgan index* pour calculer M_1 et M_2 (image tirée de [Mahé et al., 2005]).

5 Résultats numériques

5.1 Données

Le jeu de données à notre disposition provient de [Helma et al., 2004]. Il comporte 684 molécules, dont 341 sont mutagènes et 343 sont non-mutagènes. Il est donc bien équilibré, ce qui est un facteur important si l'on souhaite apprendre une fonction sur cet ensemble de données. De plus, comme ce jeu de données comprend des composants chimiques très variés, on peut parler de données "hétérogènes" (*non-congeneric*).

Pour chaque molécule, nous disposons également d'un ensemble de 17 propriétés physico-chimiques la caractérisant : masse, électronégativité, etc.

5.2 Méthodologie

Ne disposant pas d'un échantillon supplémentaire pour la validation, nous sommes contraints de tester la méthode en utilisant un procédé de validation croisée, qui permet artificiellement d'apprendre et de valider sur le même jeu de données.

Nous décidons de suivre la méthode de l'article en effectuant une 10-*fold cross validation* : nous découpons aléatoirement nos données en 10 échantillons de taille identique (sauf un qui en compte 4 supplémentaires), et successivement nous allons valider, sur l'un des 10 échantillons, la fonction apprise sur les 9 autres.

5.2.1 Principe général

La première étape consiste à calculer le noyau pour chaque itération du *Morgan index* et pour chaque valeur de la probabilité d'arrêt p_q, sous la forme d'une matrice de Gram. Pour cela nous utilisons le logiciel ChemCpp développé en C++ par les auteurs de l'article.

La deuxième étape consiste à effectuer la SVM à partir de la matrice de gram fournie. Pour cela nous utilisons le package `kernlab` de R.

5.2.2 Critère de performance

Il est nécessaire choisir un critère pour mesurer la qualité de la classification : nous prenons comme critère la valeur de l'*Area Under ROC Curve* (AUC), l'aire sous la courbe ROC issue de la validation croisée. Cette courbe représente le taux de vrais positifs en fonction du taux de faux positifs : un AUC proche de 1 indique que la fonction apprise par SVM est bonne en prédiction et se rapproche d'un classificateur parfait. En revanche, si l'AUC est proche de 0.5, la prédiction n'est pas meilleure qu'une classification par "pile ou face" équilibrée, donc la fonction apprise ne répond pas au problème.

5.2.3 Choix du coefficient C

Reste une dernière question en suspens, celle de la valeur de C utilisée dans la SVM. On pourrait certes la fixer au préalable, mais il n'y aurait alors aucune raison que cette valeur soit celle qui rende la méthode la plus performante. Pour automatiser le choix de C, on utilise une nouvelle fois la validation croisée.

- On définit une grille de valeurs de C (que l'on peut raffiner si l'on observe que toutes les valeurs sont du même ordre de grandeur). Nous avons finalement choisi la grille restreinte $[0.5, 1, 2, 4, 6]$, pour garder des temps de calculs compatibles avec le délai imparti pour ce projet. De plus, il se trouve que le temps de calcul de SVM augmente avec C. Prendre $C > 6$ rendait les calculs extrêmement longs, ce qui nous a empêchés de respecter une vraie échelle logarithmique en base 2.
- Sur chacun des 10 ensembles d'entraînement correspondants à la première validation croisée, nous effectuons une deuxième validation croisée, en séparant aléatoirement l'ensemble en 2 sous ensembles. Et ce pour chaque valeur de C de la grille. Ce choix de 2 folds a été fait par souci d'économie de temps de calcul, les valeurs obtenues étant assez similaires à celles obtenues en divisant en 10 folds.
- Sur chacun des 10 ensembles d'entrainement, nous choisissons la valeur de C qui maximise le critère AUC. Les valeurs de C ainsi obtenues sur chaque ensemble d'entrainement sont finalement celles utilisées pour la prédiction sur les 10 ensembles de validation.

5.2.4 Comparaison avec un noyau naïf

Après avoir obtenu nos résultats par la méthode utilisant les graphes, nous la comparons à une méthode SVM utilisant cette fois les caractéristiques physico-chimiques comme prédicteurs. Les objets manipulés étant des vecteurs de $\mathbb{R}^{17}$, on peut choisir le produit scalaire euclidien comme noyau et calculer la matrice de Gram normalisée associée. On applique ensuite la méthode SVM classique.

5.3 Résultats et Analyse

Nous utilisons la méthode précédente pour 5 valeurs du nombre MI d'itérations du *Morgan index* : 0,1,2,3, et 10 itérations. Pour chacune de ces valeurs, on calcule l'AUC (et d'autres critères classiques) pour les probabilités d'arrêt $p_q \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$. Les résultats sont répertoriés dans le tableau 1 (à comparer avec le tableau 6 de l'article).

On constate premièrement que ces résultats correspondent plutôt bien aux résultats de l'article. Ce n'est certes pas décisif en soi, mais cela confirme la validité de notre implémentation.

Deuxième constat formulable, les résultats obtenus sont encourageants. On observe en effet des valeurs pour l'*accuracy* supérieures à 75% dans les meilleurs cas (76.17% au mieux)

MI	p_q	Accuracy	Sensitivity	Specificity	AUC
0	0.1	74.71	67.74	81.63	81.51
	0.2	74.12	66.28	81.92	79.22
	0.3	72.08	62.76	81.34	78.42
	0.4	72.08	62.76	81.34	77.58
	0.5	70.61	61.58	79.59	75.72
1	0.1	74.85	72.14	77.55	76.25
	0.2	72.37	75.66	69.10	76.82
	0.3	76.17	74.19	78.13	82.62
	0.4	72.37	74.19	70.55	74.90
	0.5	69.88	70.67	69.10	72.34
2	0.1	74.42	76.25	72.59	76.74
	0.2	73.39	68.33	78.43	76.76
	0.3	75.58	74.19	76.97	78.55
	0.4	74.85	78.30	71.43	81.48
	0.5	75.15	73.31	76.97	81.68
3	0.1	73.39	71.26	75.51	80.43
	0.2	73.39	70.09	76.68	79.33
	0.3	73.39	72.73	74.05	77.39
	0.4	71.49	67.16	75.80	73.80
	0.5	66.67	65.10	68.22	71.84
10	0.1	58.04	55.43	60.64	61.81
	0.2	57.75	55.43	60.06	62.48
	0.3	59.36	56.60	62.10	63.35
	0.4	55.56	52.20	58.89	59.45
	0.5	57.75	61.58	53.94	59.32

TABLEAU 1 – Résultats obtenus par SVM à partir des *graph kernels*.

et des valeurs d'AUC supérieures à 80% (82.62% au mieux). Le classificateur exhibé est donc porteur d'intérêt pour prédire la mutagénicité. Ces valeurs sont globalement inférieures à celles de l'article, mais ceci est probablement du, au moins en partie, au fait nous n'avons pu considérer qu'une plage de C très restreinte.

5.3.1 Effets du *Morgan index* et de p_q

Si l'on s'intéresse à l'effet du nombre d'itérations du *Morgan index* sur la performance de la méthode (en termes d'AUC), on remarque qu'avec 1 ou 2 itérations les résultats sont légèrement meilleurs pour la meilleure probabilité p_q associée. À partir de 3 itérations, les performances de la méthode commencent à décliner, pour s'effondrer totalement pour 10 itérations. Ceci était prévisible : les sommets deviennent alors trop spécifiques au graphe. Effectuer 1 ou 2 itérations du *Morgan index* semble donc intéressant.

Cependant, l'évolution la plus spectaculaire ne concerne pas les résultats mais le temps de calcul. Il est d'ailleurs regrettable de ne pas avoir eu le temps d'étudier cette évolution. On imaginait déjà une diminution du temps de calcul avec le nombre d'itérations : en itérant, on diminue les similarités, donc la dimension du graphe produit. Or les résultats obtenus exhibent une autre conséquence intéressante. Si l'on regarde les cas de 0,1 et 2 itérations (les cas les plus intéressants en AUC), on remarque que la probabilité d'arrêt p_q donnant les meilleurs résultats augmente avec le nombre d'itérations. Comme une augmentation de p_q revient à considérer des chemins plus courts, on a une réduction de la dimension des vecteurs représentant les chemins obtenus, donc un temps de calcul inférieur pour les noyaux. Cela peut s'avérer intéressant pour de grandes bases de données.

5.3.2 Comparaison avec l'approche naïve

Enfin, si l'on compare l'utilisation des *graph kernels* à l'approche plus naïve utilisant les caractéristiques physico-chimiques des molécules comme prédicteurs pour la *SVM*, le résultat est sans appel. La SVM utilisant les prédicteurs physico-chimiques se comporte en effet comme un classificateur aléatoire. Cette différence est bien visibles sur les courbes ROC de la figure 3.

Il convient tout de même de nuancer cette affirmation car il est possible qu'un sous modèle puisse donner de meilleures performances, ou que les informations sur les molécules en notre possession ne soient pas les plus pertinentes. Nous n'avons malheureusement pas eu le temps d'analyser cela.

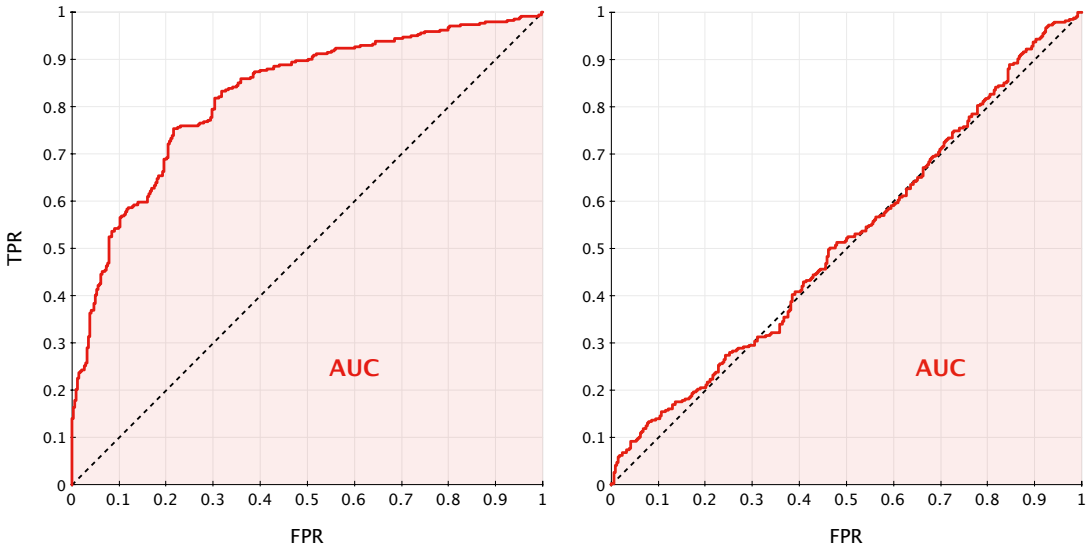

FIGURE 3 – Courbes ROC obtenues avec le *graph kernel* pour MI $= 2$ et $p_q = 0.3$ (à gauche) et avec le noyau linéaire associé aux prédicteurs physico-chimiques (à droite). Dans les deux cas les noyaux sont normalisés. La droite noire en pointillés est la courbe ROC d'un classificateur complètement aléatoire (pile ou face non biaisé).

6 Conclusions

Au cours de ce projet, nous avons étudié une méthode d'apprentissage pour la classification des molécules, basée sur les techniques SVM pour les *noyaux définis positifs*. Nous avons appliqué cette méthode pour prédire la mutagénicité des molécules, et avons constaté la pertinence de celle-ci dans ce cadre.

Gardons tout de même à l'esprit que le jeu de données était idéal (équilibré et hétérogène). Il convient également de ne pas extrapoler le résultat : la méthode est intéressante pour prédire la mutagénicité mais pourrait échouer pour prédire, par exemple, la capacité d'une molécule à attaquer un site particulier. Il est probable que des informations géométriques soient nécessaires dans ce cas. Par ailleurs, nous avons constaté l'inefficacité de la méthode utilisant les prédicteurs physico-chimiques dans ce même cadre.

L'article présentait également une autre piste d'amélioration en proposant de filtrer les *trajectoires titubantes*, c'est à dire les chemins bouclant sur eux-mêmes. Ils ne sont en effet

pas pertinents pour décrire la molécule. On n'a alors plus une chaine de Markov classique mais une chaîne d'ordre 2. Il est toutefois possible de revenir à une chaîne de Markov d'ordre 1 en construisant un nouveau graphe. Au vu du gain relativement faible et par manque de temps, nous avons décidé de ne pas traiter cette partie.

En revanche, nous avons nettement constaté l'amélioration apportée par l'utilisation du *Morgan index*. Si celle-ci n'est pas criante en termes de prédiction (elle est légère), elle l'est en revanche en termes de diminution du temps de calcul. Cela constitue une réelle avancée si l'on souhaite appliquer la méthode à une grande base de données, ou si les molécules sont de grande taille.

Références

[Cuturi, 2010] Cuturi, M. (2010). Positive Definite Kernels in Machine Learning. Technical report, Princeton University.

[Helma et al., 2004] Helma, C., Cramer, T., Kramer, S., and DeRaedt, L. (2004). Data mining and machine learning techniques for the identification of mutagenicity inducing substructures and structure activity relationships of noncongeneric compounds. *J. Chem. Inf. Comput. Sci., submitted*.

[Kashima et al., 2003] Kashima, H., Tsuda, K., and Inokuchi, A. (2003). Marginalized Graph Kernels between Labeled Graphs. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 321–328. Fawcett T., Mishra, N., Eds. ; AAAI Press.

[Mahé et al., 2005] Mahé, P., Ueda, N., Akutsu, T., Perret, J.-L., and Vert, J.-P. (2005). Graph Kernels for Molecular Structure-Activity Relationship Analysis with Support Vector Machines. *J. Chem. Inf. Model.*, (45) :939–951.

[Morgan, 1965] Morgan, H. L. (1965). The Generation of Unique Machine Description for Chemical Structures – A Technique Developped at Chemical Abstracts Service. *J. Chem. Doc.*, (5) :107–113.