

Apprentissage en bioinformatique et noyaux pour structures

Laurent Jacob

`laurent.jacob@ensmp.fr`

Center for Computational Biology
École des Mines de Paris, ParisTech

Apprentissage artificiel, 2007.

- Connaître les applications de l'apprentissage en bioinformatique.
- Comprendre les principes des méthodes à noyaux, y compris pour données structurées (applications en bioinformatique et ailleurs).

1 Apprentissage en bioinformatique

- Contexte
- Applications

2 Méthodes à noyaux

- Motivation
- Notion de noyau
- La SVM

3 Noyaux pour structures

- Motivation
- Noyaux sur marches
- Autres

1 Apprentissage en bioinformatique

- Contexte
- Applications

2 Méthodes à noyaux

- Motivation
- Notion de noyau
- La SVM

3 Noyaux pour structures

- Motivation
- Noyaux sur marches
- Autres

1 Apprentissage en bioinformatique

- Contexte
- Applications

2 Méthodes à noyaux

- Motivation
- Notion de noyau
- La SVM

3 Noyaux pour structures

- Motivation
- Noyaux sur marches
- Autres

Apprentissage en bioinformatique

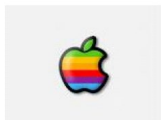
- 1 Apprentissage en bioinformatique
 - Contexte
 - Applications
- 2 Méthodes à noyaux
- 3 Noyaux pour structures
- 4 Conclusion

- Grandes quantités de données (technologies, littérature).
- Données souvent hétérogènes et/ou très bruitées.
- Analyser ces données pour
 - comprendre un système,
 - construire des outils de diagnostic,
 - concevoir de nouvelles thérapies.

Rappel : apprentissage supervisé (classification)



APPLE



APPLE



PEAR



PEAR



???



???



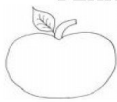
APPLE



PEAR



APPLE



APPLE



???

Formalisation

- Variables d'entrée $\mathbf{x} \in \mathcal{X}$
- Sorties $y \in \{-1, 1\}$.

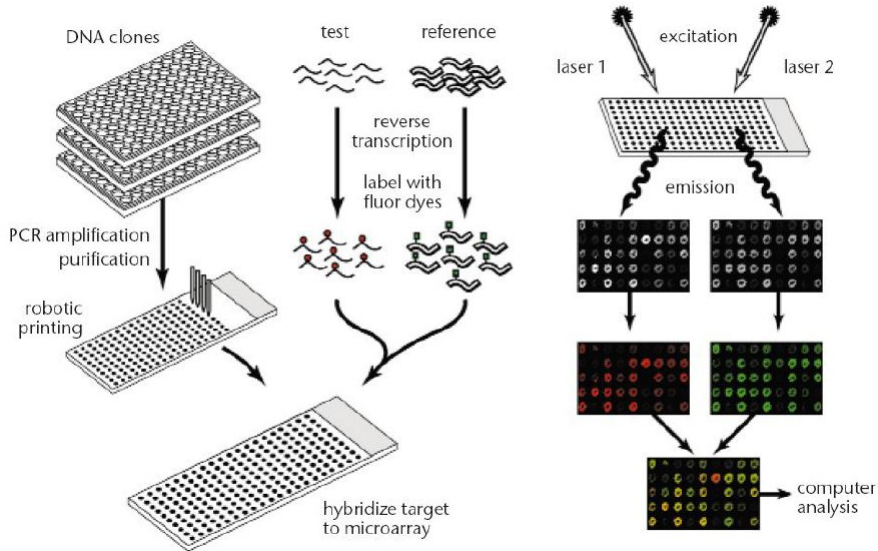
Rappel : apprentissage supervisé (classification)

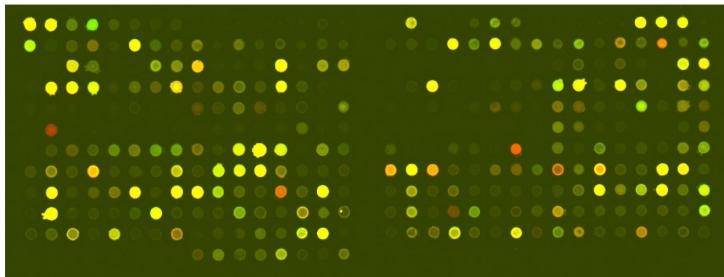
Problème

- **Ensemble d'entraînement** $\mathcal{S} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$.
- Construire un **classifieur** f à partir de \mathcal{S} , tel que $\hat{y} = f(x)$ prédise la classe d'un x inconnu le plus précisément possible.

- 1 Apprentissage en bioinformatique
 - Contexte
 - Applications
- 2 Méthodes à noyaux
- 3 Noyaux pour structures
- 4 Conclusion

Puces à ADN : principe (cas des puces cDNA)

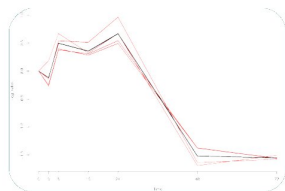
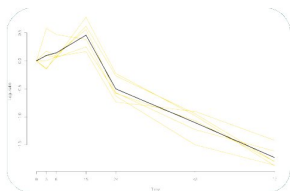
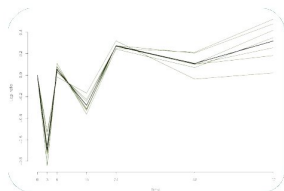




- Sortie (dans l'idéal...) : niveau d'expression de chaque gène.
- Le **génom**e d'une cellule ne change pas avec son type ou son état, le **transcriptome** si.
- Observation faite à différents instants, sous différentes conditions...

⇒ **Tendances ? Prédictions ?**

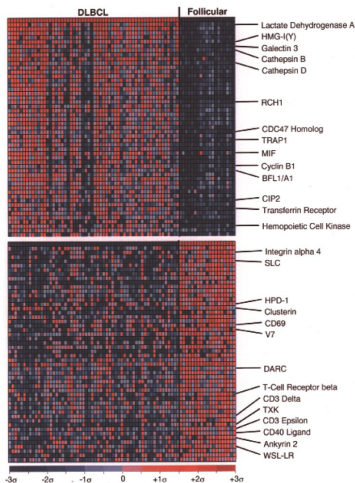
Puces à ADN : Clustering de profils d'expression de gènes



- Idée : comportement similaire dans une situation spécifique \Rightarrow gènes liés fonctionnellement ?
- Base pour établir des réseaux de régulation.

\Rightarrow Comprendre des systèmes...

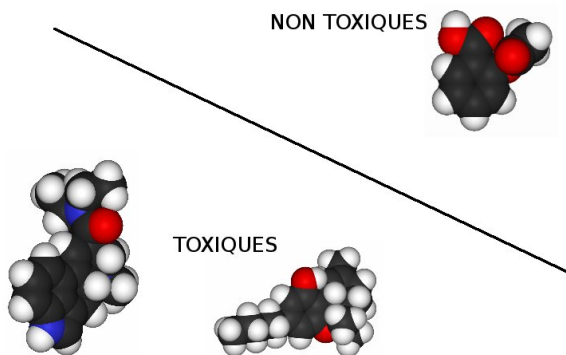
Puces à ADN : Classification de tumeur



- Idée : signature de la tumeur dans le comportement (l'expression) des gènes ?
- Savoir prédire le type de tumeur pour un nouveau patient en mesurant le transcriptome des cellules tumorales.
- Prédicteur interprétable biologiquement.

⇒ Établir des diagnostics

Toxicité de molécules



- **Données** : molécules toxiques ou non, description de ces molécules.
- Étant donnée une nouvelle molécule, est-elle toxique ?

⇒ Développement de thérapies, de bioproduits...

- Prédire si une molécule se lie à (\Rightarrow inhibe, active...) une **cible thérapeutique**.
- Prédire quelles parties d'un pathogène peuvent déclencher une **réaction immunitaire**.
- Détecter un site d'intérêt sur un chromosome, une protéine.
- Établir/enrichir des réseaux métaboliques (enzymes), de régulation de gènes, d'interaction de protéines...
- ...

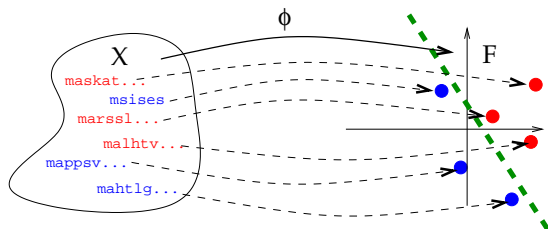
- Domaine riche en données et problèmes se prêtant à des approches *machine learning*.
- Beaucoup de problèmes non résolus avec des applications directes très importantes.

Méthodes à noyaux

- 1 Apprentissage en bioinformatique
- 2 Méthodes à noyaux
 - Motivation
 - Notion de noyau
 - La SVM
- 3 Noyaux pour structures
- 4 Conclusion

Motivation

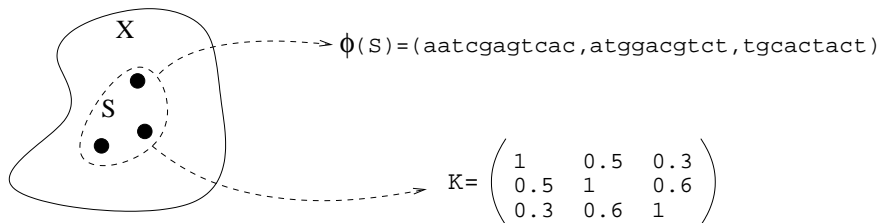
- Les algorithmes d'apprentissage (régression, KNN, réseaux de neurones, SVM...) utilisent une **représentation des données**.
- Il n'est pas forcément évident de **décrire** efficacement des données :
 - Quels descripteurs ?
 - Si on a trop de descripteurs, les calculs deviennent difficiles.
 - Un choix de descripteurs ne donne pas forcément une séparation linéaire des données.



- Les **noyaux définis positifs** proposent un cadre pour représenter des objets quelconques par comparaisons.

- 1 Apprentissage en bioinformatique
- 2 Méthodes à noyaux
 - Motivation
 - Notion de noyau
 - La SVM
- 3 Noyaux pour structures
- 4 Conclusion

Représentation par comparaisons



Idée

- Définir une “fonction de comparaison” : $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$.
- Représenter un ensemble de n données $\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ par la matrice $n \times n$:

$$[K]_{ij} := K(\mathbf{x}_i, \mathbf{x}_j)$$

Remarques

- Quelle que soit la nature des données (vecteurs, chaînes, graphes...), on a toujours une matrice $n \times n$.
- **Modularité** totale entre le **choix de K** et le **choix de l'algorithme**.
- Mauvais comportement quand on a beaucoup de données (n^2)
- On se restreint à une classe de fonctions K particulière : les noyaux définis positifs.

Noyaux définis positifs (d.p)

Definition

Un noyau **défini positif (d.p)** sur l'ensemble \mathcal{X} est une fonction $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ **symétrique** :

$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2, \quad K(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}', \mathbf{x}),$$

satisfaisant, pour tout $N \in \mathbb{N}$, $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) \in \mathcal{X}^N$ et $(a_1, a_2, \dots, a_N) \in \mathbb{R}^N$:

$$\sum_{i=1}^N \sum_{j=1}^N a_i a_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0.$$

Le noyau d.p le plus simple

Lemme

Soit $\mathcal{X} = \mathbb{R}^d$. La fonction $K : \mathcal{X}^2 \mapsto \mathbb{R}$ définie par :

$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2, \quad K(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle_{\mathbb{R}^d}$$

est d.p (c'est le **noyau linéaire**).

Preuve

- $\langle \mathbf{x}, \mathbf{x}' \rangle_{\mathbb{R}^d} = \langle \mathbf{x}', \mathbf{x} \rangle_{\mathbb{R}^d}$,
- $\sum_{i=1}^N \sum_{j=1}^N a_i a_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle_{\mathbb{R}^d} = \left\| \sum_{i=1}^N a_i \mathbf{x}_i \right\|_{\mathbb{R}^d}^2 \geq 0$

Le noyau d.p le plus simple

Lemme

Soit $\mathcal{X} = \mathbb{R}^d$. La fonction $K : \mathcal{X}^2 \mapsto \mathbb{R}$ définie par :

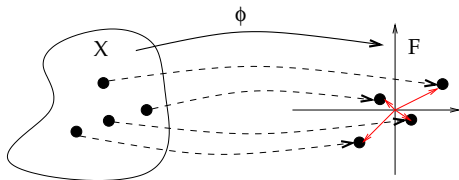
$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2, \quad K(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle_{\mathbb{R}^d}$$

est d.p (c'est le **noyau linéaire**).

Preuve

- $\langle \mathbf{x}, \mathbf{x}' \rangle_{\mathbb{R}^d} = \langle \mathbf{x}', \mathbf{x} \rangle_{\mathbb{R}^d}$,
- $\sum_{i=1}^N \sum_{j=1}^N a_i a_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle_{\mathbb{R}^d} = \left\| \sum_{i=1}^N a_i \mathbf{x}_i \right\|_{\mathbb{R}^d}^2 \geq 0$

Un noyau d.p. plus ambitieux



Lemme

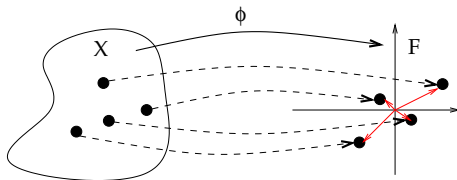
Soit \mathcal{X} un ensemble quelconque, et $\Phi : \mathcal{X} \mapsto \mathbb{R}^d$. La fonction $K : \mathcal{X}^2 \mapsto \mathbb{R}$ définie comme suit est d.p :

$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2, \quad K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathbb{R}^d} .$$

Preuve

- $\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathbb{R}^d} = \langle \Phi(\mathbf{x}'), \Phi(\mathbf{x}) \rangle_{\mathbb{R}^d}$,
- $\sum_{i=1}^N \sum_{j=1}^N a_i a_j \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle_{\mathbb{R}^d} = \left\| \sum_{i=1}^N a_i \Phi(\mathbf{x}_i) \right\|_{\mathbb{R}^d}^2 \geq 0$.

Un noyau d.p plus ambitieux



Lemme

Soit \mathcal{X} un ensemble quelconque, et $\Phi : \mathcal{X} \mapsto \mathbb{R}^d$. La fonction $K : \mathcal{X}^2 \mapsto \mathbb{R}$ définie comme suit est d.p :

$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2, \quad K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathbb{R}^d} .$$

Preuve

- $\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathbb{R}^d} = \langle \Phi(\mathbf{x}'), \Phi(\mathbf{x}) \rangle_{\mathbb{R}^d}$,
- $\sum_{i=1}^N \sum_{j=1}^N a_i a_j \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle_{\mathbb{R}^d} = \left\| \sum_{i=1}^N a_i \Phi(\mathbf{x}_i) \right\|_{\mathbb{R}^d}^2 \geq 0$.

Inversement : les noyaux comme produits scalaires

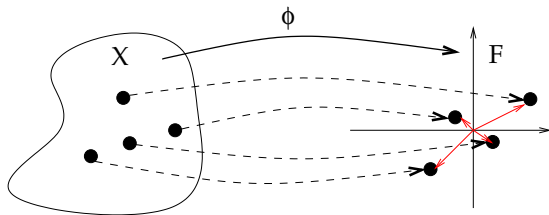
Théorème (Aronszajn, 1950)

K est un noyau d.p sur l'ensemble \mathcal{X} si et seulement si il existe un espace de Hilbert \mathcal{H} et un mapping

$$\Phi : \mathcal{X} \mapsto \mathcal{H},$$

tels que, pour tous $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$:

$$K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathcal{H}}.$$



Exemples de définitions indirectes

- Définir une description (*features*) des x et trouver une façon efficace de calculer le produit scalaire.
- Utiliser une mesure de similarité existante et vérifier qu'elle est d.p (ou la rendre d.p).

Exemples de définitions directes

- Polynomial : $K(x, x') = (xx' + 1)^d$
- Gaussien RBF : $K(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$
- Φ associés ?

Idée

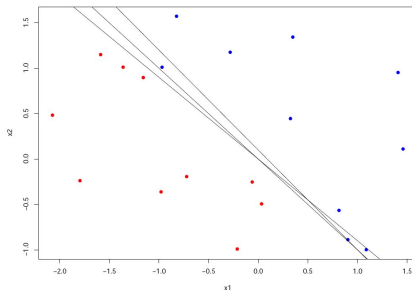
Soit un **algorithme** manipulant des vecteurs de dimension finie et pouvant s'exprimer uniquement en termes de **produits scalaires**. Remplacer les produits scalaires par un noyau K dans l'algorithme revient **implicitement** à appliquer l'algorithme aux vecteurs **décrits dans l'espace \mathcal{H}_K** .

Exemple (historique) d'algorithme "kernelisable" : la SVM.

Interêts

- Pas besoin de manipuler les $\Phi(x)$ (grande dimension, voire dimension infinie), ni même d'explicitier Φ .
- Il est parfois plus facile de définir une comparaison qu'une représentation.
- Permet d'introduire de la non linéarité à peu de frais dans un algorithme.

- 1 Apprentissage en bioinformatique
- 2 Méthodes à noyaux
 - Motivation
 - Notion de noyau
 - La SVM
- 3 Noyaux pour structures
- 4 Conclusion



Idée

- Soient des données x_i de classes $y_i \in \{-1, 1\}$ dans un certain espace.
- Comment trouver un hyperplan séparant avec le moins d'erreur possible des nouveaux points distribués de la même manière ?
- Proposition : choisir l'hyperplan de **marge maximale** sur les données d'entraînement.

Formalisation

- Si l'équation de l'hyperplan est $wx + b = 0$, alors la (demie) marge est $1/\|w\|$.
- Problème d'optimisation : minimiser $\|w\|^2$ sous les contraintes :

$$\forall i, y_i(w^T x_i + b) \geq 1$$

Cas non séparable

- Données pas toujours séparables (ou séparation parfaite n'est pas toujours souhaitable) \Rightarrow introduire des variables **ressort**.
- Nouveau problème :

$$\left\{ \begin{array}{l} \min_{w,b,\xi} \|w\|^2 + \frac{C}{n} \sum_{i=1}^n \xi_i \\ \forall i, y_i(w^T x_i + b) \geq 1 - \xi_i, \xi_i \geq 0 \end{array} \right.$$

Résolution

- Problème quadratique, contraintes linéaires convexes.
- On résout généralement le problème dual. Pour n points $x_i \in \mathbb{R}^d$, donne un problème dans \mathbb{R}^n au lieu de \mathbb{R}^d .
- C joue sur la régularité de la solution. Son choix est important et non trivial.
- La solution est de la forme $w = \sum_{i=1}^n \alpha_i x_i$, donc $\hat{y} = w \cdot x + b = \sum_{i=1}^n \alpha_i \langle x_i, x \rangle + b$.
- Si on remplace le produit scalaire par un noyau K , la solution est de la forme $f = \sum_{i=1}^n \alpha_i \Phi(x_i)$, donc $\hat{y} = f(x) = \sum_{i=1}^n \alpha_i K(x_i, x) + b$.

Analyse

- Le problème s'écrit sous la forme :

$$\min_f C \sum_{i=1}^n L(y_i, f(x_i)) + \|f\|^2.$$

- Le premier terme favorise l'ajustement de la solution au problème, le second favorise sa régularité. C gère un **compromis** entre ces deux objectifs.
- Une interprétation : si la norme de f est petite, la marge de la séparation **dans \mathcal{H}** sera grande (éventuellement au détriment de l'ajustement) \Rightarrow même comportement sur de nouvelles données.
- Une autre interprétation : en limitant la norme de f , on limite la **complexité** de la fonction recherchée ce qui garantit une meilleure généralisation en limitant le **surajustement**.

Noyaux pour structures

- 1 Apprentissage en bioinformatique
- 2 Méthodes à noyaux
- 3 Noyaux pour structures
 - Motivation
 - Noyaux sur marches
 - Autres
- 4 Conclusion

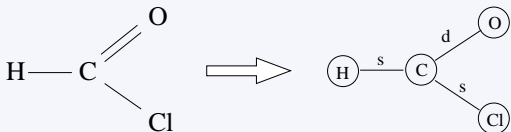
- Beaucoup de problèmes en bioinformatique impliquent des données **structurées**, non vectorielles : molécules, protéines...
- Exemple de problème où la description des objets n'est pas triviale.
- Plusieurs noyaux intégrant l'information de structure ont été proposés.
- Remarque : pas propre à la bioinformatique (images, sorties structurées...).

- But : comparer les molécules de façon pertinente au sens du problème biologique.
- On suppose donc qu'il existe un lien entre la structure de l'objet et ses propriétés : toxicité, capacité de liaison...
- Un exemple n'utilisant que la structure 2D : le noyau pour marches.

- 1 Apprentissage en bioinformatique
- 2 Méthodes à noyaux
- 3 Noyaux pour structures
 - Motivation
 - Noyaux sur marches
 - Autres
- 4 Conclusion

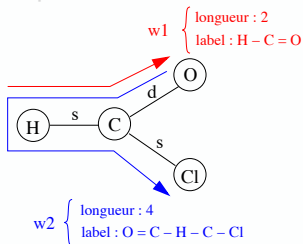
Définitions préliminaires

Grphe étiqueté : $G = (\mathcal{V}_G, \mathcal{E}_G)$, avec $l : \mathcal{V}_G \cup \mathcal{E}_G \rightarrow \mathcal{A}$



Marche sur un graphe

Séquence de **noeuds connectés** : (v_0, \dots, v_n) t.q. $(v_i, v_{i+1}) \in \mathcal{E}_G$

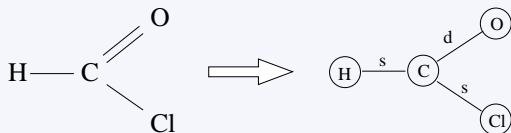


- **longueur** : nombre d'arêtes
- **label** : $l(v_0) + l(v_0, v_1) + \dots + l(v_n)$

$$\Rightarrow \mathcal{W}(G) = \{ \text{marches de } G \}$$

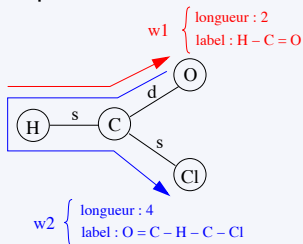
Définitions préliminaires

Grphe étiqueté : $G = (\mathcal{V}_G, \mathcal{E}_G)$, avec $l : \mathcal{V}_G \cup \mathcal{E}_G \rightarrow \mathcal{A}$



Marche sur un graphe

Séquence de **noeuds connectés** : (v_0, \dots, v_n) t.q. $(v_i, v_{i+1}) \in \mathcal{E}_G$



- **longueur** : nombre d'arêtes
- **label** : $l(v_0) + l(v_0, v_1) + \dots + l(v_n)$

$$\Rightarrow \mathcal{W}(G) = \{ \text{marches de } G \}$$

Définition générale

$$K(G_1, G_2) = \sum_{\substack{(w_1, w_2) \in \\ \mathcal{W}(G_1) \times \mathcal{W}(G_2)}} K_{\mathcal{W}}(w_1, w_2)$$

Noyau marginalisé (Kashima et al., 2003)

$$K(G_1, G_2) = \sum_{\substack{(w_1, w_2) \in \\ \mathcal{W}(G_1) \times \mathcal{W}(G_2)}} p_{G_1}(w_1) p_{G_2}(w_2) K_{\mathcal{L}}(l(w_1), l(w_2))$$

- $K_{\mathcal{L}}$ = noyau entre labels de marches
 - i.e., entre séquences de types d'atomes et de liaisons
- p_G = distribution de probabilité sur $\mathcal{W}(G)$
 - i.e., $\sum_{w \in \mathcal{W}(G)} p_G(w) = 1$

Définition générale

$$K(G_1, G_2) = \sum_{\substack{(w_1, w_2) \in \\ \mathcal{W}(G_1) \times \mathcal{W}(G_2)}} K_{\mathcal{W}}(w_1, w_2)$$

Noyau marginalisé (Kashima et al., 2003)

$$K(G_1, G_2) = \sum_{\substack{(w_1, w_2) \in \\ \mathcal{W}(G_1) \times \mathcal{W}(G_2)}} p_{G_1}(w_1) p_{G_2}(w_2) K_{\mathcal{L}}(l(w_1), l(w_2))$$

- $K_{\mathcal{L}}$ = noyau entre labels de marches
 - i.e., entre séquences de **types** d'atomes et de liaisons
- p_G = distribution de probabilité sur $\mathcal{W}(G)$
 - i.e., $\sum_{w \in \mathcal{W}(G)} p_G(w) = 1$

Noyau marginalisé : paramétrisation

Noyau entre labels de marches : $K_{\mathcal{L}}$

$$K_{\mathcal{L}}(l_1, l_2) = \begin{cases} 1 & \text{si } l_1 = l_2, \\ 0 & \text{sinon.} \end{cases}$$

⇒ noyau basé sur les **marches communes**

Distribution de probabilité sur les marches : p_G

Modèle de marche aléatoire du 1er ordre :

$$p_G(v_0, \dots, v_n) = p_s^{(G)}(v_0) \prod_{i=0}^{n-1} p_t^{(G)}(v_{i+1} | v_i)$$

En pratique = modèle "uniforme" **tué à chaque étape avec une probabilité p_q** :

$$p_G(v_0, \dots, v_n) = \frac{1}{|\mathcal{V}_G|} \prod_{i=0}^{n-1} \frac{1 - p_q}{d^+(v_i)} p_q$$

Noyau marginalisé : paramétrisation

Noyau entre labels de marches : $K_{\mathcal{L}}$

$$K_{\mathcal{L}}(l_1, l_2) = \begin{cases} 1 & \text{si } l_1 = l_2, \\ 0 & \text{sinon.} \end{cases}$$

⇒ noyau basé sur les **marches communes**

Distribution de probabilité sur les marches : p_G

Modèle de marche aléatoire du 1er ordre :

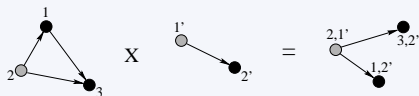
$$p_G(v_0, \dots, v_n) = p_s^{(G)}(v_0) \prod_{i=0}^{n-1} p_t^{(G)}(v_{i+1} | v_i)$$

En pratique = modèle "uniforme" **tué à chaque étape avec une probabilité p_q** :

$$p_G(v_0, \dots, v_n) = \frac{1}{|\mathcal{V}_G|} \prod_{i=0}^{n-1} \frac{1 - p_q}{d^+(v_i)} p_q$$

Implémentation : introduction

Produit de graphe : $G = G_1 \times G_2$



\Rightarrow bijection $\mathcal{W}(G)$ / marches communes à G_1 et G_2

Puissances de matrice d'adjacence

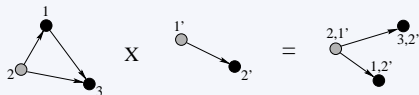
Matrice d'adjacence : $A_G[i,j] = \begin{cases} 1 & \text{si } (v_i, v_j) \in \mathcal{E}_G, \\ 0 & \text{sinon.} \end{cases}$

$\Rightarrow A_G^k[i,j] = \#$ marches $(v_i \rightarrow v_j)$ de longueur k

Π_t : même idée en "enrichissant" avec les probabilités de transition définies sur le graphe produit.

Implémentation : introduction

Produit de graphe : $G = G_1 \times G_2$



\Rightarrow bijection $\mathcal{W}(G)$ / marches communes à G_1 et G_2

Puissances de matrice d'adjacence

Matrice d'adjacence : $A_G[i,j] = \begin{cases} 1 & \text{si } (v_i, v_j) \in \mathcal{E}_G, \\ 0 & \text{sinon.} \end{cases}$

$\Rightarrow A_G^k[i,j] = \#$ marches $(v_i \rightarrow v_j)$ de longueur k

Π_t : même idée en "enrichissant" avec les probabilités de transition définies sur le graphe produit.

Calcul du noyau

- Avec la paramétrisation précédente, on a $K(G_1, G_2) = \sum_{w \in \mathcal{W}(\mathcal{G})} \pi(w)$, où π est la distribution induite sur le graphe produit.
- Par ailleurs, on montre que $\sum_{w \in \mathcal{W}(\mathcal{G}), |w|=n} \pi(w) = \pi_s^T \Pi_t^n \mathbf{1}$.
- On peut donc calculer :

$$K(G_1, G_2) = \sum_{n=0}^{\infty} \left(\sum_{h \in \mathcal{W}(\mathcal{G}), |h|=n} \pi(h) \right) = \pi_s^T (I - \Pi_t^n)^{-1} \mathbf{1}$$

⇒ **Complexité** : cubique en $|\mathcal{V}_G| = \#$ paires d'atomes identiques. Peut être accéléré dans la pratique en utilisant la structure sparse de Π_t ou en ne prenant en compte que les premiers termes de la somme.

Noyau marginalisé : remarques

Noyau marginalisé

- Feature space de dimension **infinie**
- Expression **analytique** et complexité **polynomiale**
 - **cas général** : cubique en fonction du produit de la taille des graphes
 - **avec marches communes** : cubique en fonction du nombre de paires d'atomes identiques

Limitations en pratique

- Calcul coûteux
- Structures de marches : peu expressives + bruitées

⇒ **Extensions** pour pallier à ces limitations

Noyau marginalisé : remarques

Noyau marginalisé

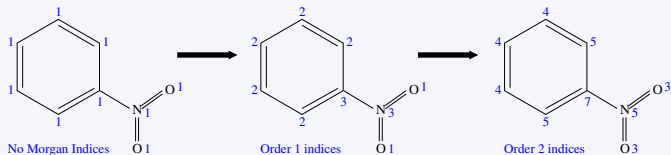
- Feature space de dimension **infinie**
- Expression **analytique** et complexité **polynomiale**
 - **cas général** : cubique en fonction du produit de la taille des graphes
 - **avec marches communes** : cubique en fonction du nombre de paires d'atomes identiques

Limitations en pratique

- Calcul coûteux
- Structures de marches : peu expressives + bruitées

⇒ **Extensions** pour pallier à ces limitations

Indices de Morgan

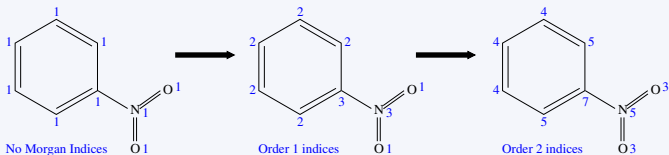


Extension = enrichir les labels

type d'atome = (type d'atome) + (indice de Morgan)

- introduction automatique d'information topologique
- avec $K_{\mathcal{L}}$ = noyau de Dirac : réduit le coût du noyau
- plusieurs niveaux de résolution

Indices de Morgan



Extension = enrichir les labels

type d'atome = (type d'atome) + (indice de Morgan)

- introduction automatique d'information topologique
- avec $K_{\mathcal{L}}$ = noyau de Dirac : réduit le coût du noyau
- plusieurs niveaux de résolution

- 1 Apprentissage en bioinformatique
- 2 Méthodes à noyaux
- 3 Noyaux pour structures
 - Motivation
 - Noyaux sur marches
 - Autres
- 4 Conclusion

Molécules

- Basés sur des sous-graphes plus complexes que les marches (par exemple des arbres).
- Basés sur la structure 3D.

Protéines

- Basés sur des scores d'alignement existant.
- Basés sur les structures secondaires.
- Basés sur les angles entre les atomes...

Conclusion

Conclusion (1/2)

Apprentissage en bioinformatique

Beaucoup de problèmes importants nécessitent de l'apprentissage :

- **Compréhension** des systèmes.
- Construction d'outils de **diagnostic**.
- *Design* de nouvelles **thérapies**.

Noyaux

- Définir un noyau d.p sur des objets est équivalent à utiliser un produit scalaire sur ces objets décrits dans un certain espace de Hilbert.
- “Astuce noyau” : remplacer les produits scalaires par un noyau d.p. revient à travailler implicitement dans l'espace des descripteurs correspondant.
- Permet d'introduire facilement des non-linéarités, de travailler dans des espaces de dimension grande voire infinie...

Noyaux pour structures

- Il est délicat de manipuler des données structurées, non vectorielles (molécules, protéines...).
- Définir un noyau comparant ces objets est une manière pratique d'utiliser la structure.
- Le noyau (la façon dont les structures sont comparées) peut être adapté en fonction du problème.