République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Ghardaïa

Faculté des Sciences et Technologie

Département des Mathématiques et Informatique

Projet de fin d'étude présenté en vue de l'obtention du diplôme de

LICENCE

Domaine : Mathématiques et Informatique

Spécialité : Informatique

THEME :

Les noyaux sur les mots

Développement d'une boite à outils.

PAR :

BEN ADBERRAHMANE Habiba

DJEKAOUA Khadra

Jury :

M. Slimane BELLAOUAR	Maitre Assistant A Univ. Ghardaïa	**Encadreur**
M. Abdelkader OULEDMAHREZ	Maitre Assistant B Univ. Ghardaïa	**Examinateur**

ANNEE UNIVERSITAIRE : 2013/2014

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Ghardaïa

Faculté des Sciences et Technologie

Département des Mathématiques et Informatique

Projet de fin d'étude présenté en vue de l'obtention du diplôme de

LICENCE

Domaine : Mathématiques et Informatique

Spécialité : Informatique

THEME :

Les noyaux sur les mots

Développement d'une boite à outils.

PAR :

BEN ADBERRAHMANE Habiba

DJEKAOUA Khadra

Jury :

M. Slimane BELLAOUAR	Maitre Assistant A Univ. Ghardaïa	**Encadreur**
M. Abdelkader OULEDMAHREZ	Maitre Assistant B Univ. Ghardaïa	**Examinateur**

ANNEE UNIVERSITAIRE : 2013/2014

Dédicaces

A ma chère mère qui a perdu sa jeunesse pour nous, et mon père,

A ma seule sœur, grâce à elle après ALLAH ce travail est terminé.

A mes chères frères,

A toute ma famille, ma binôme, et mes amis,

A tous les enseignants qui m'ont suivi durant ma formation,
Je dédie ce travail.

Habiba.

Je dédie ce modeste travail à mes parents qu'ils sont ma tuteur,

Et mes chères frères spécialement SALLAH,

Mes chères sœurs et amis,

A ma binôme et sa famille,

Et toute ma famille,
A tous ceux et celles qui m'aiment.

Khadra.

Remerciement

Nous tenons, dans un premier temps, à remercier ***ALLAH*** qui nous avoir guidé vers ce chemin, et nous aidées beaucoup tout au long de nos années d'étude et pour finaliser ce mémoire.

Nous adressons notre profond remerciement à M. **Slimane BELLAOUAR**, notre encadreur, s'est toujours montré à l'écoute et très disponible tout au long de la réalisation de ce mémoire, ainsi pour l'inspiration, l'aide et le temps qu'il a bien voulu nous consacrer et sans qui ce mémoire n'aurait jamais vu le jour.

Nous remercions également Pr **Djelloul ZIADI**, Ms. **Slimane OULAD NAOUI**, **Abdelkeder OULD MAHRAZ**, **Toufik GHARIB**, **Youcef MAHDJOUB** et Mme **Nacira BRAHIM**, et tous les enseignants de département de Mathématiques et Informatique de notre université.

Ainsi, nous adressons nos remerciements les plus chaleureux à toutes les personnes qui nous ont apporté leur aide de prés ou de loin par le fruit de leurs connaissances pendant toute la durée de notre parcours éducatif.

Enfin, nous adressons nos plus sincères remerciements à tous nos proches et amis, qui nous ont toujours soutenue et encouragée au cours de la réalisation de ce mémoire.

Résumé

Les méthodes classique de l'apprentissage automatique ne peuvent pas manipuler des données non linéairement séparables. Les méthodes à noyaux ont été proposées pour outrepasser cette limitation. Dans ce mémoire, nous présentons un survol sur l'apprentissage automatique et les méthodes à noyaux, dont le but est d'implémenter une boite à outils qui permet de calculer la fonction noyau entre les entrées (données brutes), de différentes façons et pour plusieurs besoins.

Nous décrivons, ensuite, les algorithmes de quelques méthodes à noyau sur les mots ainsi que leur analyse de complexité. Ces algorithmes sont présentés dans le livre **Kernel methods for pattern analysis** des auteurs : *John SHAWE-TAYLOR* et *Nello CRISTIANINI*.

En utilisant Octave qui est l'équivalent libre de MATLAB, nous commençons le développement d'une boite à outils simple et minimale des méthodes à noyaux sur les mots dans une perspective de l'enrichir dans le futur afin de participer à l'accélération des recherches dans le domaine de l'apprentissage automatique.

Mots clés

Noyau, mots, noyau sur les mots, apprentissage automatique.

Table des matières

Liste de Figures

Liste des Algorithmes

Liste de notations

$\mathcal{A}$	Algorithme d'apprentissage.
$\mathcal{D}$	Ensemble de données appelé également ensemble d'entraînement.
$\mathcal{H}$	Espace de Hilbert.
Φ	La fonction de description.
Σ	Alphabet
DP	Matrice intermédiaire calculé par les méthodes de programmation dynamique
f	modèle entraîné sur l'ensemble $\mathcal{D}$.
$G_{i,j}$	La matrice de Gram.
k	La fonction noyau qui calcul le produit scalaire
assqk	Noyau de toutes les sous-séquences non-contiguës
pflssqk	Noyau de toutes les sous-séquences de taille fixe p

Introduction

Les noyaux sur les mots font partie de l'apprentissage automatique qui s'intéresse à faire que les machines apprennent toutes seule (c.à.d., automatiquement) à travers des expériences.

Les méthodes de résolution des problèmes de non-linéarité sont très difficiles et coûteuses, sachant que la majorité des problèmes d'apprentissage et surtout de classification sont de cette catégorie. Par contre, les problèmes qui sont linéairement séparables sont très faciles à résoudre et moins coûteux.

Les méthodes à noyaux basées sur le principe de **transformer les problèmes non-linéairement séparables en problèmes linéairement séparables** afin d'appliquer les méthodes de résolution des problèmes linéaire. Cette transformation faite par le calcul de produit scalaire des entrées, en trouvant un nouveau espace appelé espace de description.

Le mémoire est organisé comme suit. Le premier chapitre présente le domaine d'étude, l'apprentissage automatique et sa relation avec les méthodes à noyaux.

Le chapitre 2 est consacré aux méthodes à noyaux sur les mots du point de vue algorithmique et complexité.

Le chapitre 3 est une ébauche à la construction d'une boite à outils pour les méthodes à noyaux sur les mots développé en utilisant Octave.

Chapitre 1

Généralités sur l'apprentissage automatique

Dans ce chapitre nous présentons le passage de l'apprentissage automatique classique aux méthodes à noyaux, ainsi que l'introduction des notions fondamentales utilisées par la suite.

1.1 Apprentissage automatique

L'apprentissage automatique ou artificiel (**Machine Learning** en anglais) est une sous discipline de l'*intelligence artificiel*, qui répond à la question : "***Comment peut-on programmer des systèmes qui apprennent automatiquement ?***".

définition 1 (Apprentissage automatique) *cette notion s'intéresse a l'écriture de programmes d'ordinateur capables de s'améliorer automatiquement au fil du temps, soit sur la base de leur propre expérience, soit à partir de données antérieures fournies par d'autres programmes. Donc elle englobe toute méthode permettant de construire un modèle de la réalité à partir de données, soit en améliorant un modèle partiel ou moins général, soit en créant complètement le modèle[6].*

Formellement, " Un programme d'ordinateur est capable d'apprendre à partir d'une expérience E et par rapport à un ensemble T de tâches et selon une mesure de performance P, si sa performance à effectuer une tâche de T, mesurée par P, s'améliore avec l'expérience E ". Donc, l'apprentissage automatique pour la machine est qu'avec l'ensemble de tâches T que la machine doit réaliser, elle utilise l'ensemble d'expériences E telle que sa performance sur T est améliorée[11].
Avant d'entamer les types d'apprentissage nous présentons la définition d'un algorithme d'apprentissage[9].

définition 2 (Algorithme d'apprentissage) *Un algorithme d'apprentissage $\mathcal{A}$ est un algorithme prenant en entrée un ensemble de données $\mathcal{D}$ et retournant une fonction f.*

Où $\mathcal{D}$ désigné comme **ensemble d'entraînement** ou ensemble d'apprentissage, et f comme **modèle**, sachant que les éléments de $\mathcal{D}$ sont appelés **exemples d'apprentissage**.
Autrement dit, on conventionne que *le modèle f a été entraîné sur l'ensemble $\mathcal{D}$.*
Le schéma suivant (1.1) résume cette définition :

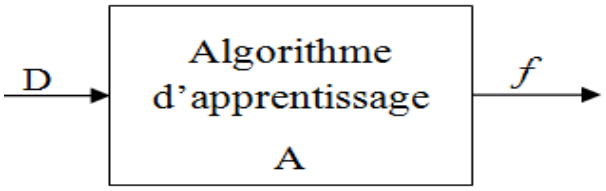

FIGURE 1.1 – L'algorithme d'apprentissage

1.1.1 Apprentissage automatique : types et domaines d'application

Il existe plusieurs classifications de type d'apprentissage automatique selon quelques critères (points de vues), une classification fait la distinction entre les tendances : celle issue de l'*intelligence artificielle* et qualifie de **symbolique**, et celle issue des *statistiques* et qualifiée de **numérique**[6]. La classification la plus répondue considère qu'il existe plusieurs types se distinguent essentiellement par leur objectif, c.à.d., la nature de ce qui doit être appris[11]. (Remarquez que dans un système on peut combiner différents types d'apprentissage).
Dans ce qui suit, nous investiguons la classification qui considère ces types : l'apprentissage supervisé, non-supervisé et semi-supervisé.

L'apprentissage supervisé

Ce type d'apprentissage est utilisé dans le cas ou l'objectif est déterminé explicitement [9]. Un expert doit étiqueter[11] (c.à.d, associer une cible) correctement à une entrée. Ceci peut être modélisé comme suit[9] :

$$\mathcal{D} = \{(x_t, y_t) | x_t \in X, y_t \in Y\}_{t=1}^{n} \tag{1.1}$$

où X est l'ensemble d'**entrées** et Y est l'ensemble de **cibles**, sachant que la nature de Y liée aux type de problème à résoudre.
Autrement dit, l'objectif de l'apprentissage supervisé est d'extraire une relation reliant y à x représentée par une fonction qui à son tour est extraite par des principes d'induction (parmi ces principes d'induction : **minimisation du risque théorique**, **minimisation du risque empirique**, **minimisation du risque structurel**, ainsi les approches **discriminante** et **générative**)[4].

L'apprentissage supervisé est appliqué principalement à des problèmes de **classification** et **régression**[9] :

– **Classification** : intuitivement, "une règle de classification est un acte cognitif ou une procédure permettant d'affecter à un objet la famille à laquelle il appartient, autrement dit de le reconnaître"[6]. Dans ce cas, Y correspond à un ensemble fini de classes. Si $x \in X$ est une séquence, Y peut aussi être un ensemble de séquences de classes. Parmi les problèmes de classification on trouve **la reconnaissance de visage** où l'entrée x est une image bitmap d'une personne fournie par une caméra, et la sortie y indiquerait de quelle personne il s'agit parmi l'ensemble de personnes que l'on souhaite voir le système reconnaître[10] ; **Classification syntaxique** où l'entrée x correspond à une phrase dans une langue donnée et y est la séquence des classes syntaxiques de chacun des mots de x[9]. La figure suivante illustre des problèmes de classification linéaire et non-linéaire :

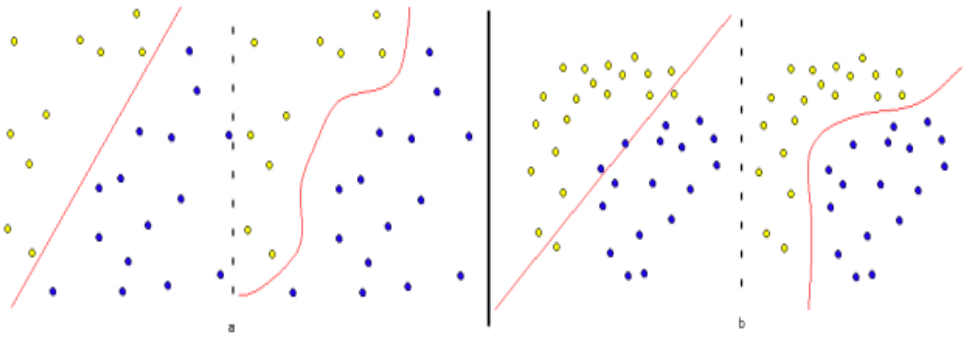

FIGURE 1.2 – Problèmes de classification : linéairement séparable (à gauche), non linéairement séparable (à droite)

– **Régression** : son principe et qu'à partir d'une distribution d'observations, rechercher la relation qui correspond le mieux entre la/les valeur(s) explicative(s) et la valeur expliquée[8]. Formellement elle définie une méthode d'estimation qui permet d'interpoler, voire d'extrapoler ou de prédire,la relation entre deux ou plusieurs variables c.à.d., la valeur des données en recherchant une relation entre celles-ci[5]. En cas de régression, Y correspond à un ensemble de valeurs continues ou de vecteurs de valeurs continues. On utilise cette méthode aux problèmes de **prédiction** comme **prédiction de poids** l'entrée x correspond à des caractéristiques physiques d'une personne (par exemple son age, son sexe, ...) et la sortie y correspond à son poids. La figure suivante (1.3) présente des exemples de problèmes de régression, ou les cibles prennent des valeurs réelles[9].

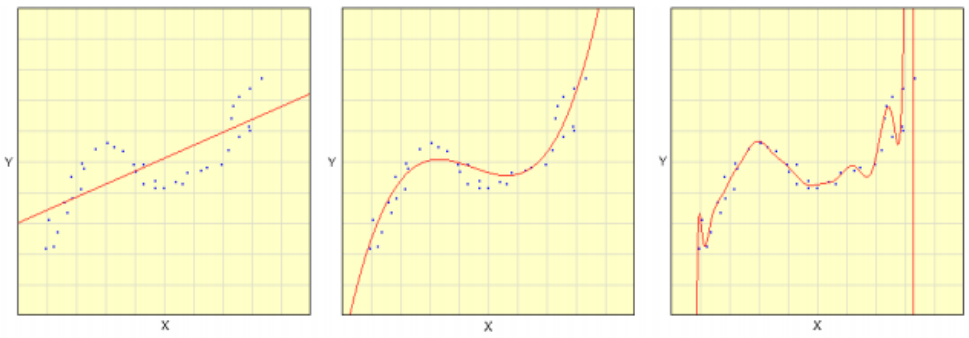

FIGURE 1.3 – problèmes de régression : de gauche à droite, un modèle linéaire, quadratique et polynomial de degré 20

L'apprentissage non-supervisé

Quand aucun expert n'est disponible ni requis[11] et aucune cible (étiquette) n'est prédéterminé et la nature de la fonction f (le modèle) retourné par l'algorithme d'apprentissage n'est pas défini explicitement, on parle de l'apprentissage non supervisé, il peut être modéliser par l'équation[9] :

$$\mathcal{D} = \{x_i | x_i \in X\}_{i=1}^{n} \tag{1.2}$$

L'utilisateur doit spécifier le problème à résoudre parmi :

* **Estimation de densité** ;
* **Extraction de caractéristiques** ;
* **Réduction de dimensionnalité** ;
* **Groupage (*clustering* en anglais)**

L'apprentissage semi-supervisé

L'apprentissage semi-supervisé consiste à utiliser un ensemble d'exemples non-étiquetés, en plus d'un ensemble des exemples étiquetés, afin d'influencer la procédure d'un algorithme d'apprentissage et d'améliorer sa performance[9].

Donc, c'est une combinaison entre l'apprentissage supervisé et non-supervisé.

Ce type est souvent appliqué en **régularisation** (c'est une approche générale fréquemment utilisée pour exercer un contrôle sur la capacité d'un modèle[9]).

1.1.2 modèle paramétrique ou non-paramétrique

On ne peut distinguer un algorithme d'apprentissage automatique uniquement par le type d'apprentissage qu'il implémente, mais aussi par le modèle donné par cet algorithme [9] :

Les modèles paramétriques ont une forme précise qui ne change pas en fonction de la quantité de données. Ces modèles sont basés sur des paramètres, Par exemple, un modèle linéaire entraîné par la minimisation de risque empirique. Au contraire, **les modèles non-paramétriques** deviennent de plus en plus complexes au fur et à mesure que le nombre d'exemples d'apprentissage augmente. La capacité de ces modèles à représenter des phénomènes complexes progresse avec le nombre d'exemples. Un exemple de modèles non-paramétriques, les machines à noyau entraîné par la minimisation de risque empirique.

1.2 L'astuce Noyau

Le modèle linéaire est un parmi les modèles les plus simple. Si les deux régions de l'espace d'entrée associées à n'importe quelle paire de classe d'un modèle sont **séparables par un hyperplan**, on dit que ce modèle est **linéaire**.

Les modèles linéaires sont couramment utilisés particulièrement sur les problèmes où l'entrée est de très haute dimension.

Il existe plusieurs algorithmes d'apprentissage pour les modèles linéaires qui se distinguent principalement par le coût utilisé pour l'entraînement, parmi lesquels citons : **perceptron**, **le classificateur à moindre carrés** et **la machine à vecteurs de support linéaire**.

Les modèles linéaires ont un inconvénient très important : la faible capacité. Ils sont incapables de résoudre suffisamment un problème de classification dont lequel les données sont linéairement inséparables, c.à.d., si la frontière entre les régions de l'espace d'entrée associées aux différentes classes est non-linéaire, sachant qu'en pratique il est rare qu'un problème de classification soit linéaire[9].

Pour résoudre les problèmes de non linéarité et couvrir la limitations d'autres méthodes (comme la méthode de **réseaux de neurones artificiel**) l'astuce **Noyau** apparaît.

Cette astuce consiste à partir de données non-linéairement séparable de trouver via une fonction

de passage Φ, un espace dans lequel ces données pourront être linéairement séparable (en appliquant les algorithmes linéaire). Cet nouveau espace est appelé **espace de description**, ou **d'attributs**, ou encore **de caractéristiques**[4].
La figure suivante (1.4) explique ce principe :

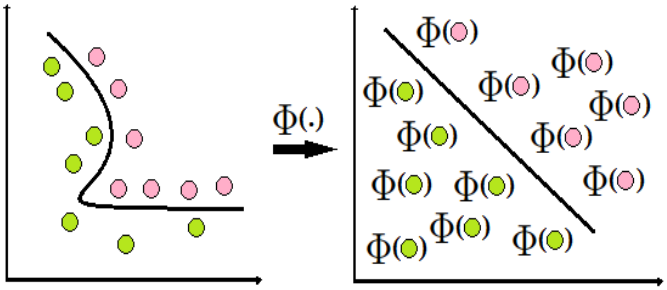

FIGURE 1.4 – Le principe de l'utilisation de noyau

1.3 Modularité de méthodes à noyau

La modularité des méthodes à noyau[13] se manifeste dans la réutilisation de l'algorithme d'apprentissage. Le même algorithme peut fonctionner avec n'importe quel noyau et donc pour n'importe quel domaine de données. Le composant du noyau est un ensemble de données spécifiques, mais peut être combiné avec différents algorithmes pour résoudre l'ensemble des tâches que nous allons considérer. Tout cela conduit à une approche très naturelle et élégante à l'apprentissage de conception de systèmes , où les modules sont combinés pour obtenir des systèmes complexes d'apprentissage.

La figure 1.5 présente les étapes de la mise en œuvre de l'analyse de modèle à noyau.

Les données sont traitées en utilisant un noyau pour créer une matrice de noyau, qui est à son tour traitée par un algorithme d'analyse de motif pour produire une fonction de configuration. Cette fonction est utilisée pour traiter des exemples invisibles.

1.4 Noyaux sur les mots

Dans cette section nous présentons la définition d'un noyau, sa caractérisation, ainsi quelques opérations sur les noyaux.

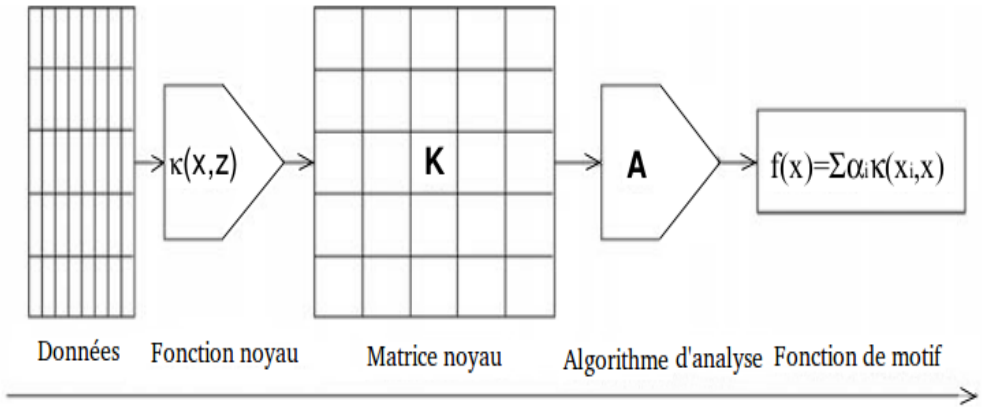

FIGURE 1.5 – La modularité des méthodes à noyau

1.4.1 définitions

définition 3 (Noyau, Kernel en anglais) *Un noyau c'est une fonction k pour tous $x, y \in E$ (E un espace vectoriel) satisfaisant[13] :*

$$k(x, y) = \langle \Phi(x), \Phi(y) \rangle . \tag{1.3}$$

Où Φ est une application linéaire de E dans F :

$$\begin{array}{rccc} \Phi : & E & \rightarrow & F \\ & x & \mapsto & \Phi(x) \end{array} \tag{1.4}$$

Rappels

Rappelons des notions importantes :

définition 4 (Application linéaire) *f est une application linéaire de E dans F si :*

$$\forall x, y \in E : f(x + y) = f(x) + f(y) \tag{1.5}$$

et

$$\forall x \in E, \forall \lambda \in K : f(\lambda.x) = \lambda.f(x) \tag{1.6}$$

Où E et F sont des K-espaces vectoriels.[7]

définition 5 (Espace vectoriel) *un espace vectoriel est un ensemble non vide dont les éléments s'appellent des* ***vecteurs*** *et qui est muni de deux opérations :* la somme de vecteurs *et* la multiplication

d'un scalaire par un vecteur *vérifiant les huit axiomes suivants[7] :*
Étant donné E un K-espace vectoriel

1. ***Associativité** :*

$$\forall x, y, z \in E : (x + y) + z = x + (y + z) \tag{1.7}$$

2. ***Commutativité** :*

$$\forall x, y \in E : x + y = y + x \tag{1.8}$$

3. ***Élément neutre** : E contient un élément 0_E appelé élément neutre pour l'addition tel que :*

$$\forall x \in E : x + 0_E = 0_E + x = x \tag{1.9}$$

4. ***Élément opposé (symétrique)** : $\forall x \in E$, E contient un élément noté $-x$ appelé élément opposé de x, tel que :*

$$x + (-x) = (-x) + x = 0_E \tag{1.10}$$

5. ***Distributivité à gauche***

$$\forall \alpha \in K, \forall x, y \in E : \alpha(x + y) = \alpha x + \alpha y. \tag{1.11}$$

6. ***Distributivité à droite***

$$\forall \alpha, \beta \in K, \forall x \in E : (\alpha + \beta)x = \alpha x + \beta x. \tag{1.12}$$

7.

$$\forall \alpha, \beta \in K, \forall x \in E : \alpha(\beta.x) = (\alpha.\beta)x. \tag{1.13}$$

8. ***Élément absorbant***

$$\forall x \in E : 1x = x. \tag{1.14}$$

Remarque

Dans notre cas $K = \mathbb{R}$.

Citons maintenant la définition de quelques termes utilisés ultérieurement :

définition 6 (Produit scalaire) *Soit E espace vectoriel, l'application $f : E \times E$ dans E est un **produit scalaire** si elle vérifie qu'elle est[4] :*

– ***symétrique** :*

$$\forall (x, y) \in E^2, f(x, y) = f(y, x) \tag{1.15}$$

– ***Bilinéaire*** *:*

$$\forall (x, y, z) \in E^3, f(x + z, y) = f(x, y + z) \text{ et } \forall \alpha \in \mathbb{R}, f(\alpha x, y) = f(x, \alpha y) = \alpha f(x, y) \tag{1.16}$$

– ***Strictement positive*** *:*

$$\forall x \in E - \{0\}, f(x, x) > 0 \ (f(x, x) = 0 \Leftrightarrow x = 0) \tag{1.17}$$

Le produit scalaire est noté : $\langle ., . \rangle$

définition 7 (Espace de Hilbert) *Un espace de Hilbert (ou espace Hilbertien)* $\mathcal{H}$ *est un espace vectoriel complet doté du produit scalaire* $\langle ., . \rangle_{\mathcal{H}}$ *dont la norme découle du produit scalaire.[4]*

1.4.2 La matrice de Gram

C'est une matrice qui contient l'évaluation de la fonction noyau pour toutes paires de données. Elle agit quand un goulot d'étranglement d'informations, le modèle où le bruit sera extrait de la cette matrice.

définition 8 *Étant donné un ensemble de vecteurs,* $S = \{x_1, x_2, ..., x_\ell\}$ *la* ***matrice de Gram*** *est définie comme la matrice carré G d'ordre* ℓ *tel que* $G_{i,j} = \left\langle x_i, x_j \right\rangle$.
Si on utilise la fonction noyau k pour évaluer le produit scalaire dans un espace de description avec une fonction de description Φ*, les entrées de la matrice de Gram associées[13]*

$$G_{i,j} = \left\langle \Phi(x_i), \Phi(x_j) \right\rangle = k(x_i, x_j). \tag{1.18}$$

Dans ce cas, la matrice fait souvent référence à la **matrice noyau** noté :

K	1	2	...	ℓ
1	$k(x_1, x_1)$	$k(x_1, x_2)$	...	$k(x_1, x_\ell)$
2	$k(x_2, x_1)$	$k(x_2, x_2)$	...	$k(x_2, x_\ell)$
⋮	⋮	⋮	⋱	⋮
ℓ	$k(x_\ell, x_1)$	$k(x_\ell, x_2)$	...	$k(x_\ell, x_\ell)$

La matrice est symétrique si $G_{i,j} = G_{j,i}$, et on note $G' = G$[13].

1.4.3 Caractérisation de noyaux

Nous présentons maintenant une caractérisation importante, permet de montrer qu'une fonction candidat est un noyau.

Théorème 1 (Noyau semi-défini positif) *Une fonction noyau $k : X \times X \rightarrow \mathbb{R}$ est dite semi-défini positive sur X si[4] :*

- *k est symétrique, c.à.d.,*

$$\forall (x, y) \in X^2, k(x, y) = k(y, x) \tag{1.19}$$

- *la matrice de Gram K (avec $K_{ij} = k(x_i, x_j)$) associée à la fonction noyau k est semi-défini positive, c.à.d.,*

$$\forall (x_1, \ldots, x_n) \in X^n, \forall (c_1, \ldots, c_n) \in \mathbb{R}^n : \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K_{ij} \geq 0 \tag{1.20}$$

1.4.4 Noyau de Mercer

L'inconvénient de noyau est qu'on doit définir la fonction Φ, citons les conséquences[4] :

1. une bonne définition de Φ exige une connaissance de l'espace de description. Or, dans la majorité des cas, cette connaissance fait défaut ;
2. lorsque l'espace de description est de haute dimension et/ou complexe, il peut être calculatoirement coûteux d'effectuer le passage vers l'espace de description avant d'effectuer le calcul du produit scalaire. Une expression optimisée du produit scalaire sans transformation explicite peut être plus adaptée ;
3. dans le cas d'un espace de description de dimension infini comme celui induit par une fonction gaussienne (RBF-Radial Basis Function-), la définition de Φ est impossible ;

Dans la suite, nous présentons comment définir un noyau sans définir explicitement la fonction Φ[4].

Théorème 2 (Mercer) *Soit X un sous-ensemble compact de $\mathbb{R}^n$, Supposons que k est une fonction symétrique continue tel que l'opérateur d'intégration $T_k : L_2(X) \rightarrow L_2(X)$ -Sachant que $L_2(X) = \{f : \int_X f(x)^2 dx < \infty\}$[13]- :*

$$(T_k f)(.) = \int_X k(., x) f(x) dx \tag{1.21}$$

est positive, à savoir :

$$\int_X \int_X k(x, y) f(x) f(y) dx dy \geq 0 \tag{1.22}$$

pour toute fonction $f \in L_2(X)$. Alors, $k(x, y)$ peut être décomposé en séries uniformément convergentes, $\Phi_j \in L_2(X)$ sur $X \times X$:

$$k(x, y) = \sum_{j=1}^{\infty} \lambda_j \Phi_j(x) \Phi_j(y) \tag{1.23}$$

Avec Φ_j les fonctions propres de T_k tel que $\left\|\Phi_j\right\|_{L_2} = 1$ et $\lambda_j > 0$ les valeurs propres associées.
De plus, les séries $\sum_{j=1}^{\infty} \left\|\Phi_j\right\|_{L_2(X)}$ sont convergentes.

Si une fonction k est continue semi-définie positive, la condition suffisante issue du théorème de Mercer, permet de conclure qu'il existe un espace de Hilbert dans lequel k est un produit scalaire. Par conséquent, les noyaux vérifiant le théorème de Mercer, appelés noyaux de Mercer, sont des noyaux valides.

définition 9 (Noyau valide) *Une fonction $k : X \times X \to \mathbb{R}$ est un noyau valide si et seulement si k est un produit scalaire dans un espace de Hilbert.*

1.4.5 Opérations sur les fonctions noyaux

Dans cette section nous présentons des opérations sur les noyaux qui nous permettent de construire des nouveaux noyaux[13] :

Propriétés de clôture (fermeture)

Étant donné k_1, k_2 deux noyaux de $X \times X$, $X \subseteq \mathbb{R}^n$, $x, y \in X$, $a \in \mathbb{R}^+$, $f(.)$ une fonction à valeurs réelles dans X, $\Phi : X \to \mathbb{R}^N$ avec k_3 un noyau de $\mathbb{R}^N \times \mathbb{R}^N$, et B une matrice $n \times n$ symétrique semi-défini positive. les fonctions suivantes sont des noyaux :

1. $k(x, y) = k_1(x, y) + k_2(x, y)$
2. $k(x, y) = ak_1(x, y)$
3. $k(x, y) = k_1(x, y)k_2(x, y)$
4. $k(x, y) = f(x)f(y)$
5. $k(x, y) = k_3(\Phi(x), \Phi(y))$
6. $k(x, y) = x'By$
7. $k(x, y) = \exp(k_1(x, y))$
8. $k(x, y) = \exp(-\frac{\|x-y\|^2}{2\sigma^2})$ est connue sous le **noyau Gaussien**

1.5 Conclusion

L'apprentissage automatique est une sous discipline de l'intelligence artificielle qui s'intéresse à faire que les ordinateurs apprennent tous seuls à partir des expériences. Les méthodes à noyaux font partie de l'apprentissage automatique, elles sont couramment utilisées aujourd'hui, car elles permettent de résoudre les problèmes classiques non-linéaire de l'apprentissage automatique par le calcul de produit scalaire de projections de deux points de données dans un espace de description.

Les fonctions noyau sont caractérisées par la propriété qu'elles sont semi-définies positives.

Lorsque l'espace de description est de haute dimension et/ou complexe ou infini, il est difficile, voir impossible de définir la fonction de description Φ, donc la nécessité de rechercher des méthodes permettent de calculer le noyau sans calculer cette fonction.

On peut définir des nouveaux noyaux en combinant des noyaux par l'application de propriétés de fermeture.

Chapitre 2

Noyaux sur les mots : algorithmes et analyse

Nous avons présenté dans le premier chapitre un petit tour d'horizon sur l'apprentissage automatique dans lequel s'inscrit notre projet.

Parmi les types de données, le type string (mot) qui est très essentiel dans les applications de bioinformatique. Il est utilisé pour représenter les **protéine** comme une séquence d'acides aminés, **ADN** comme une séquence de nucléotides (**Adénine, Cytosine, Guanine, Thymine**).

Les noyaux calculent le produit scalaire entre les images de mots dans un espace de description de haute dimension.

Ce chapitre consacre une partie pour définir les notions de mot, sous-mots et d'autres termes, sur lesquels nous appliquons la fonction noyau ; et une partie pour citer les algorithmes implémentés dans ce contexte, ainsi que leurs analyses à fin de choisir quelques uns pour la réalisation de notre boite à outils dans le chapitre ci-après.

La majorité des informations présentées dans ce chapitre sont de la référence [13].

2.1 Mot, sous-mot, séquence et sous-séquence

définition 10 (Mot, String en anglais) *Le grand dictionnaire terminologique[1] définit le concept* ***mot*** *comme étant un ensemble d'éléments contigus de même nature, considéré comme un tout. le terme mot ou encore string fait souvent référence à la chaîne de caractère.*

Formellement, un alphabet est un ensemble fini Σ *de symboles. Le nombre d'éléments (longueur) de* Σ *est noté par* $|\Sigma|$.

Un ***mot*** $s = s_1 \dots s_{|s|}$ *est* ***une séquence de symbole de*** Σ*, y compris la séquence vide noté* ε*, le seul mot de longueur* 0. Σ^n *dénote l'ensemble de tous les mots fini de longueur n, et* $\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$ *dénote l'ensemble de tous les mots qui peuvent être formés de* Σ.

On note $[s = t]$ *la fonction qui renvoie 1 si s et t son identique ; 0 sinon.*

La notation $s(i : j)$ *correspond au sous-mot (substring)* $s_i s_{i+1} \dots s_j$ *de s pour* $1 \leq i \leq j \leq |s|$.

Un mot t est un sous-mot de s s'il existe deux mots u, v (éventuellement vide) tel que : $s = utv$. *Les sous-mots de longueur n sont appelés* ℓ***-grams*** *ou* ℓ***-mers***.

définition 11 (Sous-séquence, subsequence en anglais) *Le grand dictionnaire terminologique[1] définit le concept* ***séquence*** *comme étant une suite d'éléments disposés dans un ordre donné.*

Formellement, le mot u est une sous-séquence de s s'il existe une suite d'indices $I = (i_1, \dots, i_{|u|})$ *dans s, avec* $1 \leq i_1 \leq \dots \leq i_{|u|} \leq |s|$ *tel que* $u_j = s_{i_j}$ *pour* $j = 1, \dots, |u|$. *Dans la littérature, on utilise la notation* $u = s(I)$ *si u est une sous-séquence de s dont les positions sont données par I. La longueur de u est le nombre d'indices de u est dénoté par* $|u| = |I|$ *tandis que la longueur de sous-séquence u est* $\ell(I) = i_{|u|} - i_1 + 1$. *Elle désigne le nombre de caractères de s couvertes par la sous-séquence u. Le mot vide* ε *est indexé par le tuple vide.*

2.2 Algorithmes de noyau sur les mots : implémentations et analyses

2.2.1 Noyau de *p-spectre*

Le noyau de spectre (ou spectrum en anglais) est le simple noyau pour comparer deux mots, il consiste à calculer le nombre de sous-mots de longueur p qui ont en commun. En cas des applications où la contiguïté joues un rôle important. La comparaison de ces sous-mots donne des informations important sur leur similarité.

définition 12 (Le noyau *p*-spectre) *L'espace de description* $\mathcal{H}$ *associé au noyau p-spectre est indexé par* $I = \Sigma^p$. *La fonction de description tel que le* $u^{ème}$ *composant du vecteur* $\Phi^p(s)$ *associé à la séquence s avec* $u \in \Sigma^P$ *est défini[13] comme suit :*

$$\Phi_u^p(s) = |\{(v_1, v_2) : s = v_1 u v_2\}| \tag{2.1}$$

Donc le noyau p-spectre est défini comme :

$$k_p(s,t) = \langle \Phi^p(s), \Phi^p(t) \rangle = \sum_{u \in \Sigma^p} \Phi^p_u(s) \Phi^p_u(t) \tag{2.2}$$

Exemple 1 (Le noyau 2-spectre) *Considérons les mots "**bar**", "**bat**", "**car**" et "**cat**". Ses 2-spectres son donnés dans la table suivante :*

ϕ	*ar*	*at*	*ba*	*ca*
bar	1	0	1	0
bat	0	1	1	0
car	1	0	0	1
cat	0	1	0	1

avec toutes autres dimensions indexé avec autres mots de langueur 2 *ont la valeur* 0*, donc la matrice noyau résultat est :*

K	*bar*	*bat*	*car*	*cat*
bar	2	1	1	0
bat	1	2	0	1
car	1	0	2	1
cat	0	1	1	2

Définition récursive de noyau *p-spectre*

Une définition récursive de ce noyau à travers un noyau supplémentaire appelé **noyau de *k*-suffixe** qui est défini par :

définition 13 (noyau de *k*-suffixe)

$$k^S_k(s,t) = \begin{cases} 1 & si\ s = s_1 u,\ t = t_1 u,\ pour\ u \in \Sigma^k \\ 0 & sinon \end{cases} \tag{2.3}$$

Donc le noyau p-spectre sera :

$$k_p(s,t) = \sum_{i=1}^{|s|-p+1} \sum_{j=1}^{|t|-p+1} k^S_p(s(i:i+p), t(j:j+p)) \tag{2.4}$$

Analyse de l'algorithme

La complexité de ce noyau est $O(p\,|s|\,|t|)$, car le noyau 2.3 nécessite $O(k)$ comparaisons donc le noyau p-spectre 2.4 peut être évaluée en $O(p\,|s|\,|t|)$ opérations.

Une optimisation possible en utilisant les méthodes de **la programmation dynamique** [1] et des structures de données appropriées comme **les arbres de suffixes** rend ce coût : $O(p \max(|s|, |t|))$[3]. (Pour plus d'information voir [13]).

2.2.2 Noyau de *toutes les sous-séquences*

Ce noyau considère toutes les sous-séquences contiguës et non-contiguës de toute taille du mot.

définition 14 (Noyau de toutes les sous-séquences) *L'espace de description associé avec le noyau de toutes les sous-séquences est indexé par $I = \Sigma^*$, avec la fonction de description est défini comme :*

$$\phi_u(s) = |\{\boldsymbol{i} : u = s(\boldsymbol{i})\}|, u \in I \tag{2.5}$$

autrement dit, cette fonction de description permet de plonger la séquence s dans un espace vectoriel dans lequel le $u^{ème}$ composant de $\Phi(s)$ indique la fréquence d'occurrence de la séquence u comme sous-séquence dans s. Le noyau associé est défini par :

$$k(s,t) = \langle \phi(s), \phi(t) \rangle = \sum_{u \in \Sigma^*} \phi(s)\phi(t). \tag{2.6}$$

Exemple 2 *Toutes les sous-séquences (non contigus) dans le mots "**bar**", "**baa**", "**car**" et "**cat**" sont donnés dans le tableau ci-dessous :*

1. Les méthodes (techniques) de programmation dynamique consiste à stocker des résultats intermédiaire dans une structure de données et de les utiliser pour construire d'autres résultats intermédiaire qui sont à leur tour stockés et utilisés pour construire des résultats toujours plus complets. Voir : Eric KOW, *Réalisation de surface : ambiguïté et déterminisme*, thèse Doctorat, université Henri Poincaré - Nancy I (France), 2007, page *viii*.

ϕ	*bar*	*baa*	*car*	*cat*
ε	1	1	1	1
a	1	2	1	1
b	1	1	0	0
c	0	0	1	1
r	1	0	1	0
t	0	0	0	1
aa	0	1	0	0
ar	1	0	1	0
at	0	0	0	1
ba	1	2	0	0
br	1	0	0	0
bt	0	0	0	0
ca	0	0	1	1
cr	0	0	1	0
ct	0	0	0	1
bar	1	0	0	0
baa	0	1	0	0
car	0	0	1	0
cat	0	0	0	1

et comme toutes autres coordonnées (infinies) doivent avoir la valeur zéro, la matrice du noyau est :

K	*bar*	*baa*	*car*	*cat*
bar	8	6	4	2
baa	6	12	3	3
car	4	3	8	4
cat	2	3	4	8

Notez qu'on général, les éléments de diagonal de cette matrice noyau peut varier considérablement avec la longueur d'un mot et même comme illustré ici entre des mots de même longueur, quand ils ont des nombres différents de séquences répétées. Bien sûr, cet effet peut être retiré si nous choisissons de normaliser ce noyau.

Définition récursive de noyau de *toutes les sous-séquences*

Il est possible d'évaluer le noyau de *toutes les sous-séquences* plus efficacement. Au lieu de calculer explicitement les vecteurs de description, nous utilisons une définition récursive et des techniques de la **programmation dynamique** afin de compléter un tableau de valeurs avec l'évaluation du noyau est donnée par la dernière entrée dans le tableau. Ces techniques sont illustrées par le biais des algorithmes ci-après.

Remarquons que la fonction de description est indexée par le mot u. La contribution de cette fonction peut être exprimée comme suit :

$$\phi_u(s) = \sum_{\mathbf{i}:u=s(\mathbf{i})} 1 \Longrightarrow \phi_u(s)\phi_u(t) = \sum_{\mathbf{i}:u=s(\mathbf{i})} 1 \sum_{\mathbf{j}:u=t(\mathbf{j})} 1 = \sum_{(\mathbf{i},\,\mathbf{j}):u=s(\mathbf{i})=t(\mathbf{j})} 1. \tag{2.7}$$

par conséquent,

$$k(s,t) = \langle \phi(s), \phi(t) \rangle = \sum_{u \in I} \sum_{(\mathbf{i},\,\mathbf{j}):u=s(\mathbf{i})=t(\mathbf{j})} 1 = \sum_{(\mathbf{i},\,\mathbf{j}):u=s(\mathbf{i})=t(\mathbf{j})} 1 \tag{2.8}$$

La clé de la récursion est de considérer l'ajout d'un symbole supplémentaire a à l'une des chaînes s. Dans la somme finale de l'équation 2.8, il existe deux possibilités pour le tuple i : soit il est entièrement contenu dans s ou son indice final correspond à la finale a. Dans le premier cas, la sous-séquence est une sous-séquence de s, tandis que dans le second son symbole final est a. Autrement dit, toute sous-séquence de s contenant le dernier caractère a de s, tel que $s = s'a$, ne peut apparaître dans t qu'entre le premier caractère de t et la dernière occurrence de a dans t.

Par conséquent, nous pouvons diviser cette somme en deux parties :

$$\sum_{(\mathbf{i},\,\mathbf{j}):sa(\mathbf{i})=t(\mathbf{j})} 1 = \sum_{(\mathbf{i},\,\mathbf{j}):s(\mathbf{i})=t(\mathbf{j})} 1 + \sum_{u:t=uav)} \sum_{(\mathbf{i},\,\mathbf{j}):s(\mathbf{i})=u(\mathbf{j})} 1 \tag{2.9}$$

définition 15 (Définition récursive) *La dernière équation (2.9) mène à la définition récursive suivante :*

$$\begin{aligned} k(s,\varepsilon) &= 1, \\ k(sa,t) &= k(s,t) + \textstyle\sum_{k:t_k=a} k(s,t(1:k-1)) \end{aligned} \tag{2.10}$$

Algorithme de noyau de *toutes les sous-séquences*

Comme le calcul de produit scalaire est très coûteux, un algorithme proposé basé sur une méthode de programmation dynamique permet de calculer le noyau sans calculer la fonction de description Φ et le produit scalaire (le noyau k) à travers l'évaluation d'une matrice nommé DP tel que $DP = K_{ij} = k(s(1..i), t(1..j))$:

DP	ε	t_1	t_2	...	t_m
ε	1	1	1	...	1
s_1	1	k_{11}	k_{12}	...	k_{1m}
s_2	1	k_{21}	k_{22}	...	k_{2m}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
s_n	1	k_{n1}	k_{n2}	...	$k_{nm} = k(s,t)$

Cette matrice est calculé depuis un vecteur P.

Algorithme 1 (Noyau de toutes les sous-séquences non-contigus) *Le noyau de toutes les sous-séquences non-contigus est donné par le pseudo-code 2.1 :*

Entrées	s, t deux mots de longueur n et m respectivement
1	pour j de 1 à m faire
2	$DP(0, j) \leftarrow 1$
3	pour i de 1 à n faire
4	début
5	$last \leftarrow 0$
6	$P(0) \leftarrow 0$
7	pour k de 1 à m faire
8	début
9	$P(k) \leftarrow P(last)$
10	si $t_k = s_i$ alors
11	début
12	$P(k) \leftarrow P(last) + DP(i-1, k-1)$
13	$last \leftarrow k$
14	fin
15	fin
16	pour k de 1 à m faire
17	$DP(i, k) \leftarrow DP(i-1, k) + P(k)$
18	fin
Sorties	l'évaluation du noyau $k(s,t) = DP(n,m)$

Fragment de code 2.1 – Pseudo-code de noyau de *toutes les sous-séquences non-contiguës*.

Exemple 3 *Soit s = "cata", t = "gatta" deux mots. Pour calculer son noyau $k(s,t)$ et le noyau de toutes les sous-séquence sans calculer explicitement les vecteurs de description, nous appliquons l'algorithme donné par le pseudo-code 2.1, nous obtenons :*

DP	ε	g	a	t	t	a	P
ε	1	1	1	1	1	1	*[0, 0, 0, 0, 0, 0]*
c	1	1	1	1	1	1	*[0, 0, 1, 1, 1, 2]*
a	1	1	2	2	2	3	*[0, 0, 0, 2, 4, 4]*
t	1	1	2	4	6	7	*[0, 0, 1, 1, 1, 7]*
a	1	1	3	5	7	14	− − − − − −−

par exemple k("gatt","cat") = 6 donné par $DP(3,4)$.

Analyse de l'algorithme

Cet algorithme a un coût de $O(|s||t|)$, car considérons que l'opération significative est la comparaison (ligne 10) qui se fait $n \times m$ fois : $\sum_{i=1} \sum_{k=m} 1 = n \times m = |s||t|$. Le noyau de **toutes les sous-séquences non-contiguës** permet de calculer toutes les motifs communs de deux mots, mais l'inconvénient majeurs que l'espace de projection est de très haute dimension entraînant un temps de calcul important.

2.2.3 Noyaux des *sous-séquence de longueur fixe* p

L'idée de ce noyau est que l'espace de description peut être réduit pour qu'il accomplit une tache particulière, si en adaptent la relation de récurrence 2.10 de sorte à ne considérer que les sous-séquences non contiguës de taille fixe p.

définition 16 (Noyaux des sous-séquence de longueur fixe p) *L'espace de description associé avec les noyaux des sous-séquence de longueur fixe p est indexé par Σ^p, avec la fonction de description définie comme :*

$$\phi_u^p(s) = |\{\boldsymbol{i} : u = s(\boldsymbol{i})\}|, u \in \Sigma^p \tag{2.11}$$

Le noyau associé est défini par :

$$k(s,t) = \langle \phi^p(s), \phi^p(t) \rangle = \sum_{u \in \Sigma^p} \phi_u^p(s) \phi_u^p(t). \tag{2.12}$$

Définition récursive de noyaux des *sous-séquence de longueur fixe* p

définition 17 *Nous appliquons un calcul similaire de noyau de toutes les sous-séquences sauf que nous devons restreindre les indices de séquences à ceux de longueur p, nous obtenons la définition récursive :*

$$\begin{aligned} k_0(s,t) &= 1, \\ k_p(s,\varepsilon) &= 0, \ for\ p > 0 \\ k_p(sa,t) &= k_p(s,t) + \textstyle\sum_{k:t_k=a} k_{p-1}(s, t(1:k-1)) \end{aligned} \tag{2.13}$$

Algorithme de noyaux des *sous-séquence de longueur fixe* p

L'équation 2.13 montre que nous avons une récursion sur le préfixe des mots en retirant à chaque étape le dernier élément de la séquence, et une autre sur $p = |u|$ la longueur de sous-séquences considérées. L'algorithme suivant inclut cette récurrence, pour calculer le noyau en mémorisant l'évaluation du noyau précédent dans une matrice **DPrec**.

Algorithme 2 (Noyaux des sous-séquences de longueur fixe p) *Les noyaux des sous-séquences de longueur fixe p est calculé par le pseudo-code 2.2 :*

Exemple 4 *En utilisant les mots de l'exemple 3, nous obtenons pour* $p = 1, 2, 3$ *:*

$DP : k_0$	ε	g	a	t	t	a	$DP : k_1$	ε	g	a	t	t	a
ε	1	1	1	1	1	1	ε	0	0	0	0	0	0
c	1	1	1	1	1	1	c	0	0	0	0	0	0
a	1	1	1	1	1	1	a	0	0	1	1	1	2
t	1	1	1	1	1	1	t	0	0	1	2	3	4
a	1	1	1	1	1	1	a	0	0	2	3	4	6
$DP : k_0$	ε	g	a	t	t	a	$DP : k_1$	ε	g	a	t	t	a
ε	0	0	0	0	0	0	ε	0	0	0	0	0	0
c	0	0	0	0	0	0	c	0	0	0	0	0	0
a	0	0	0	0	0	0	a	0	0	0	0	0	0
t	0	0	0	1	2	2	t	0	0	0	0	0	0
a	0	0	0	1	2	5	a	0	0	0	0	0	2

Notons que la somme de quatre matrices $\{DP : k_i \mid i \in \{0, 1, 2, 3\}\}$ *est identique au noyau de toutes les sous-séquences, car les deux mots ne partagent que des séquences de longueur* 3 *de sorte qu'en sommant sur les longueurs* $0, 1, 2, 3$ *qui englobent toutes les sous-séquences communes. Cela suggère aussi que si nous calculons les noyaux des sous-séquences de longueur fixe p que nous pouvons,*

Entrées	s, t deux mots de longueur n et m respectivement et p
1	$DP(0:n, 0:m) = 1$
3	pour l de 1 à p faire
4	début
5	$DPrec \leftarrow DP$
6	pour j de 1 à m faire
7	$DP(0, j) \leftarrow 1$
8	pour i de 1 à $n - p + l$ faire
9	début
10	$last \leftarrow 0$
11	$P(0) \leftarrow 0$
12	pour k de 1 à m faire
13	début
14	$P(k) \leftarrow P(last)$
15	si $t_k = s_i$ alors
16	début
17	$P(k) \leftarrow P(last) + DPrec(i-1, k-1)$
18	$last \leftarrow k$
19	fin
20	fin
21	pour k de 1 à m faire
22	$DP(i, k) \leftarrow DP(i-1, k) + P(k)$
23	fin
Sortie	l'évaluation du noyau $k_p(s, t) = DP(n, m)$

Fragment de code 2.2 – Pseudo-code de noyaux des *sous-séquence de longueur fixe p*.

presque sans aucun coût supplémentaire, de calculer le noyau k_l pour $l \leq p$.
Par conséquent, nous avons la possibilité de définir un noyau qui combine ces différents noyaux des sous-séquences avec des pondérations différentes $a_l \geq 0$:

$$k(s, t) = \sum_{l=1}^{p} a_l k_l(s, t). \tag{2.14}$$

Le seul changement à l'algorithme ci-dessus serait que la borne supérieure de la boucle de la variable i aurait besoin pour devenir n plutôt que n − p + l et nous aurions besoin d'accumuler la somme à la

fin de la boucle de l avec

$$Kern = Kern + a(l)DP(n,m) \tag{2.15}$$

où la variable l est initialisé à 0 *dès le début de l'algorithme.*

Analyse de l'algorithme

La complexité de cet algorithme est $O(p\,|s|\,|t|)$ car à la $p^{ème}$ étape nous devons être capable d'accéder à la ligne au dessus de l'autre dans la matrice courante et dans la matrice $DPrec$ de l'étape précédente, sur l'avant dernière étape lorsque $l = p-1$ il suffit de calculer jusqu'au rang $n-1$, puisque c'est tout ce qui est nécessaire dans la boucle finale avec $l = p$. Dans l'implémentation donnée ci-dessus, nous avons fait l'utilisation de ce fait à chaque étape , de sorte que pour l'étape de $l^{ème}$ nous avons calculé que jusqu'à la rangée $n - p + l$, où $l = 1, ..., p$.

2.3 Conclusion

Les méthodes à noyau sont très importantes car elle permettent de résoudre les problèmes non-linéaire.

Ce chapitre est spécifié à présenter quelques noyaux sur les mots tels le **noyau** p**-spectre** basé sur l'idée intuitive pour comparer deux mots s et t, de calculer le nombre de sous-mots de longueur p qui ont en communs. Le noyau de **toutes les sous-séquences** qui considère toutes les sous-séquences contiguës et non-contiguës de toute taille du mot. Les **noyaux des sous-séquences de longueur fixe** p permettent de réduire l'espace de description de noyau précédent (noyau de toutes les sous-séquences) de sort qu'en ne considérant que les sous-séquences non contiguës de taille fixe p.

Le coût de ces noyaux est de $O(p\,|s|\,|t|)$ pour les noyaux p**-spectres** et **des sous-séquences de taille fixe** p et de $O(|s|\,|t|)$ pour les noyau et **toutes les sous-séquences**.

Chapitre 3

Implémentation d'une boîte à outils de noyaux sur les mots

Dans le chapitre précédent nous avons présenté quelques noyaux sur les mots et ainsi que l'analyse de leur complexité.

Dans ce chapitre nous essayons d'implémenter les algorithmes présentés précédemment sous forme d'une boîte à outils.

3.1 Le langage MATLAB

MATLAB est un langage de calcul scientifique de haut niveau et un environnement interactif pour le développement d'algorithmes, la visualisation de données, analyse des données et le calcul numérique. Il permet de résoudre les problèmes de calcul scientifique plus rapidement qu'avec les langages de programmation traditionnels, tels que java, C, C++ et Fortran[2]...

La figure suivante (3.1) illustre quelque domaines d'application de MATLAB[14] :

MATLAB est utilisé dans une large gamme d'applications, y compris traitement du signal et de l'image, de la communication, la conception du contrôle, de test et mesure, modélisation et analyse financière, et la bio-informatique. Parmi cas caractéristiques[2] :

- Langage de haut niveau pour le calcul scientifique ;
- Environnement de développement pour la gestion du code, des fichiers et des données ;
- Offre des outils interactifs pour l'exploration itérative, la conception et la résolution de problèmes ;

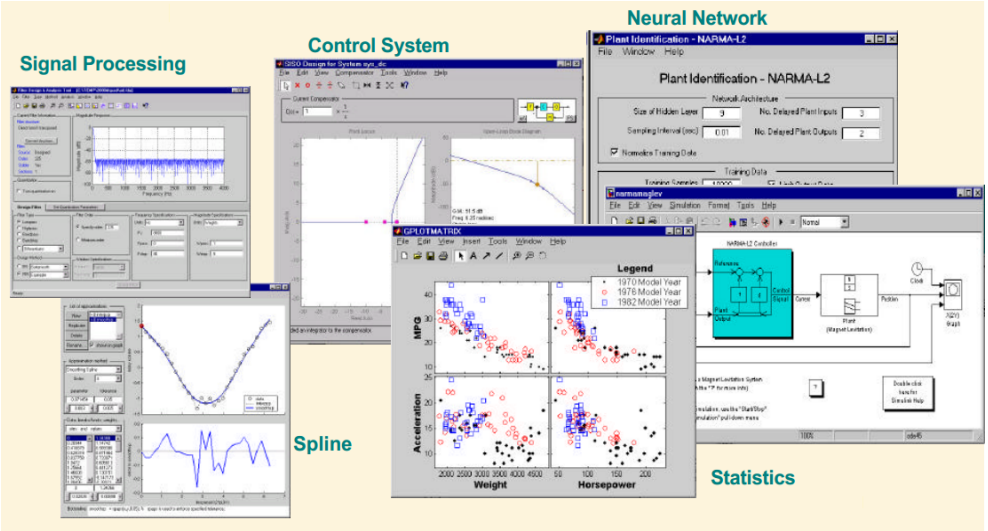

FIGURE 3.1 – Domaines d'application de MATLAB

– Fonctions mathématiques pour l'algèbre linéaire, les statistiques, l'analyse de Fourier, le filtrage, l'optimisation et l'intégration numérique ;
– Le dessin de fonctions graphiques en 2-D et 3-D pour la visualisation des données ;
– Outils pour la création d'interfaces utilisateur graphiques personnalisées ;
– Fonctions pour l'intégration des algorithmes développés en langage MATLAB avec des applications externes et des langages, tels que C, C++, Fortran, Java...

MATLAB offre un certain nombre de fonctionnalités pour la documentation et le partage de travail. Octave est la version libre de MATLAB

3.2 Méthode de construction d'une boîte à outils sous MATLAB

Une boîte à outils (toolbox en anglais) est une collections de fonctions MATLAB à des fins spéciales, disponibles séparément. Elle étend l'environnement MATLAB pour résoudre des catégories particulières de problèmes dans ces domaines d'application. Il est possible d'intégrer des codes MATLAB avec d'autres langages et applications, et les distribuer.

3.2.1 Présentation de notre boîte à outils

Notre boîte à outils est composé de deux fonctions calculent des noyaux selon les besoins (tel le noyau considérant des séquences de taille fixe, les séquences d'ADN de longueur 4, par exemple).

3.2.2 Installation de la boîte à outils

Pour que vous pouvez utiliser notre boîte à outils suiviez les étapes suivantes :

1. d'abord décomposer le fichier zip (rar) **String_kernels_toolbox.zip** ;
2. copier le chemin de l'existence de fichier **String_kernels_toolbox** ;
3. taper la commande : **cd** ′ <**chemin_copier**>′ sous Octave (ou MATLAB) ;
4. Vous pouvez avoir l'aide de chaque fonction à travers la commande :
   help <**nom_de_fonction**>
5. exécuter la fonction avec vos paramètres ;

3.3 Les fonctions implémentées

La boîte à outils **String_kernels_toolbox** contient deux fonctions **assqk** : noyau le toutes les sous-séquences non-contiguës, et **pflssqk** : noyaux des sous-séquences de taille fixe p

3.3.1 Paramétrage de fonctions

Le paramétrage de nos fonctions est très facile, la fonction **assqk** prend deux (02) paramètres : s et t de type string qui est représenté en MATLAB sous forme d'un vecteur ligne. La fonction **pflssqk** prend trois (03) paramètres : deux (02) strings s et t et un entier p représente la longueur de sous-séquences.

3.3.2 Code de fonctions

Cette partie représente le code des fonctions implémentées sous Octave :

Noyau de toutes les sous-séquences non-contiguës

```
function result = assqk (s, t)
% assqk : all non-contiguous subsequences kernel
%compute the kernel (inner products) of tow strings given as inputs
using a dynamic programming implementation.
% It using : val = assqk (s, t), i.e. it return an integer.
% * Exemples : assqk("cata", "gatta") return 14
%                    assqk("cat", "bar") return 2

s = strcat('x', s) ; % x used as epsilon.
t = strcat('x', t) ;
%Obtain lengths of strings
[ls, n] = size(s) ;
[lt, m] = size(t);
%Initialization
DP=ones(n, m) ;
P=zeros(1, m) ;
%Program
for i = 2 : n
        last= 1 ;
        P(1)=0 ;
        for k = 2 : m
                P(k)=P(last) ;
                if t(k)==s(i)
                        P(k)=P(last) + DP(i-1, k-1) ;
                        last=k ;
                endif
        endfor
        for k = 1 : m
                DP(i, k)=DP(i-1, k)+P(k) ;
        endfor
endfor
result = DP(n, m) ;
endfunction
```

Fragment de code 3.1 – Fonction noyau de toutes les sous-séquences non-contiguës

Noyau de toutes les sous-séquences de taille fixe p

```
function result = pflssqk (s, t, p)
% pflssqk : p-fixed length subsequence kernel
%compute the kernel (inner products) of tow strings considering non-contiguous
% substrings that have a given fixed length p using
% a dynamic programming implementation. This strings given as inputs.
% It using : result = pflssqk (s, t, p), i.e. it return an integer.
% * Exemples : pflssqk("cata", "gatta", 1) return 6
%                    pflssqk("cata", "gatt", 2) return 2
s = strcat('x', s) ; % x used as epsilon.
t = strcat('x', t) ;
%Obtain lengths of strings
[ls, n] = size(s) ;     [lt, m] = size(t);
%Error checking statements : - Make sure input vectors are horizontal :
if (ls ~ = 1 | lt ~ = 1)
    error('Error : s and t must be horizontal vectors.') ;
endif ;
%Error checking statements : - If p is less than zero or not a number, program
    should quit due to faulty variable input.
if p < 0 | ischar(p)
    error('Error : p needs to be a number greater than 0.') ;
endif ;
%Initialization
DP=ones(n, m) ;   DPrec=[] ;   P=zeros(1, m) ;
%Program
for l = 1 : p
    DPrec = DP ;
    for j = 1 : m
        DP(1, j) = 1 ;
    endfor
    for i = 2 : n - p + l
        last= 1 ;
        P(1)=0 ;
        for k = 2 : m
            P(k)=P(last) ;
            if t(k)==s(i)
                P(k)=P(last) + DPrec(i-1, k-1) ;
                last=k ;
            endif
        endfor
        for k = 1 : m
            DP(i, k)=DP(i-1, k)+P(k) ;
        endfor
    endfor
endfor
result = DP(n, m) ;
endfunction
```

Fragment de code 3.2 – Fonction noyau de toutes les sous-séquences de taille fixe p

3.4 Conclusion

Une boîte à outils est une collection de fonctions écrites en langage MATLAB permettent d'étendre l'environnement de développement MATLAB (Octave), à fin qu'il peut résoudre plus de problèmes dans différents domaines.

Ce chapitre est consacré à l'implémentation de notre boîte à outils qui contient deux fonctions qui calculent le noyau entre les mots. Ces mots sont représentés en langage MATLAB sous fourme de vecteurs ligne.

Notre implémentation est courte à cause de temps, elle ne contient pas d'interface graphique, et les données sont entrées manuellement (c.à.d., à la ligne de commande).

Conclusion

Les méthodes à noyaux ont été proposées pour faire face aux limitations des méthodes d'apprentissage traditionnels. Notre projet s'intéresse aux méthodes de noyaux sur les mots qui ont un grand intérêt dans l'apprentissage automatique. Ils trouvent leur intérêt dans différentes domaines d'application comme traitement automatique de la langue, la bio-informatique...

Nous avons décrits quelques algorithmes de noyaux sur les mots ainsi que leurs analyses de complexité, à savoir le *noyau de p-spectre* qui a un coût à d'ordre de $O(p\,|s|\,|t|)$ o*ù*, s et t deux mots à calculer son noyau et p est la longueur de sous-mots, le *noyau de toutes les sous-séquences* dont le coût est à l'ordre de $O(|s|\,|t|)$, si nous considerons toutes les sous-séquences non-contiguës, et est d'ordre $O(p\,|s|\,|t|)$ si nous considerons que les sous-séquences de longueur fixe p, pour s et t deux mots à calculer son noyau. Cette étude est couronnée par l'ouverture d'un grand projet de développement d'une boite à outils des méthodes à noyaux. Notre contribution se résume en l'implémentation, avec Octave, de deux algorithmes suscités qui nécessite encore des tests pour raffiner l'implémentation.

Parmi les limitations que nous avons rencontré, le manque de références en français, sachant que nous nous sommes basés sur une référence anglaise.

Fondé sur les résultats obtenus citées ci-dessus, nous proposons d'approfondir l'étude de ce sujet pour étudier la panoplies des algorithmes restants, tel l'algorithme de *noyau de sous-séquences* noté généralement **ssk** qu'il faut implémenter afin d'enrichir la boite à outils et par conséquent, participer au développement de ce domaine de recherche.

Références bibliographiques

[1] Le grand dictionnaire terminologique est une banque de fiches terminologiques rédigées par l'Office québècois de la langue française ou des partenaires de l'Office. Chaque fiche renseigne sur un concept lié à un domaine d'emploi spécialisé et présente les termes qui le désignent en français, en anglais et, parfois, dans d'autres langues.`http://gdt.oqlf.gouv.qc.ca/index.aspx`.

[2] *MATLAB Getting Started Guide*, mathworks édition, Septembre 2011.

[3] Sujeevan ASEERVATHAM et Emmanuel VIENNET : Méthodes à noyaux appliquées aux textes structurés. *Revue des Nouvelles Technologies de l'Information, RNTI-A2*, pages 185–205, 2008.

[4] Younès BENNANI : *Apprentissage à base de Noyaux Sémantiques pour le Traitement de Données Textuelles*. Thèse de doctorat, Université Paris 13, Decembre 2007.

[5] Johan COLLIEZ : *Une approche basée sur la régression par les machines à vecteurs supports : application au suivi d'objets en mouvement dans les séquences d'images*. Thèse de doctorat, l'Université du Littoral Cote d'Opale, france, 2008.

[6] Antoine CORNUÉJOLS et Laurent MICLET : *Apprentissage artificiel, Concepts et algorithmes*. Editions Eyrolles, 2003. Avec la participation d'Yves Kodratoff.

[7] Anne DENMAT et François HEAULME : *Algèbre linéaire*. Edition Dunod.

[8] Mathieu GLADE : *Modélisation des coûts de cycle de vie : prévision des coûts de maintenance et de la fiabilité Application à l'aéronautique*. Thèse de doctorat, L'école centrale de doctorat de Lion, Janvier 2005.

[9] Hugo LAROCHELLE : *Étude de techniques d'apprentissage non-supervisé pour l'amélioration de l'entraînement supervisé de modèles connexionnistes*. Thèse de doctorat, Université de Montréal - Canada, Décembre 2008.

[10] Radjà MAAROUF : Amélioration de l'apprentissage d'un Modèle Neuronal pour la reconnaissance des anomalies cardiaques. Mémoire de Magistère, Université Abou Bakr BELKAID Tlemcen, Juillet 2010.

[11] Nadia MARREF : Apprentissage Incrémental et Machines à Vecteurs Supports. Mémoire de Magistère, Université de Hadj Lakhdar Batna - Algérie, Décembre 2013.

[12] Bernhard SCHOLKOPF et Alexander J.SMOLA : *Learning with Kernels, Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[13] John SHAWE-TAYLOR et Nello CRISTIANINI : *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[14] Peter WEBB : *MATLAB : Platform Architecture How the MathWorks puts the "Tower of Power" to work for us*, mathworks édition, 2000.