RAPPORT DE STAGE
CLASSIFICATION ET ATTRIBUTION DES PICS DE RÉSONANCE DES G-QUADRUPLEXES

ZORODDU Lucas
Année ARIA S1, ENS Paris-Saclay, Université Paris-Saclay
Encadré par M. Brahim Heddi, LBPA, ENS Paris-Saclay

Septembre 2021 – Janvier 2022

Sommaire

Résumé

La formation de G-quadruplex (structures secondaires de l'ARN) a été corrélée à diverses maladies génétiques humaines et ces structures sont donc devenues des cibles attrayantes pour les médicaments. Une des méthodes pour étudier ces structures est la spectroscopie RMN, plutôt simple à réaliser expérimentalement mais conduisant à des résultats plus complexes à analyser.
Une base de données en ligne (Protein Data Bank) recense plusieurs centaines de G-quadruplex et met à disposition leur structure 3D ainsi que leur spectre RMN. Cela motive l'introduction d'algorithmes d'apprentissage pour automatiser certaines étapes de l'étude des G-quadruplex.
Lors de ce stage je me suis intéressé dans un premier temps à la classification des G-quadruplex selon leur topologie à l'aide des spectres RMN. J'ai ensuite étudier le problème d'attribution des pics de résonances d'un spectre RMN 1D.
Dans ce rapport je compare et discute les résultats obtenus avec trois types d'algorithmes différents pour la classification, puis je présente deux variantes d'une méthode probabiliste et statistique pour l'attribution des pics.

1 Introduction

1.1 Contexte

Les acides nucléiques sont d'une importance fondamentale chez tous les êtres vivants, en étant le support de leur information génétique. Ils sont des assemblages de macromolécules dont l'unité de base est appelé un nucléotide (Adénine, Cytosine, Guanine, Thymine). On distingue deux type d'acides nucléiques, l'ADN (acide désoxyribonucléique) qui contient le génome et qui reste dans le noyau, et l'ARN (acide ribonucléique) qui joue le rôle de messager en copiant l'ADN ou qui a un rôle catalytique.

La structure la plus connue et la plus représentée de l'ADN est l'hélice B, découverte par Watson et Crick en 1953 [13]. On trouve également deux autre types d'ADN en hélice : l'ADN-Z et l'ADN-A.

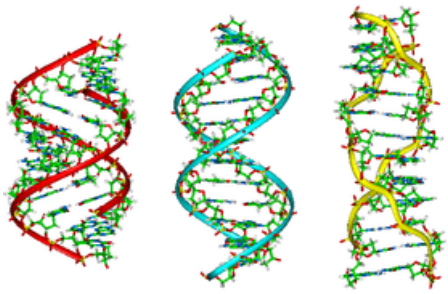
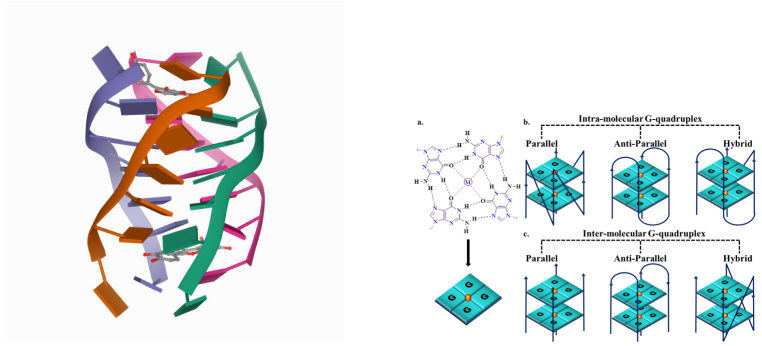


Figure 1: Structures en hélice de l'ADN (gauche : B , milieu : A , droite : Z).

Dans les zones riches en guanines, le repliement d'un brin d'acides nucléiques peut permettre la formation d'une structure secondaire d'ADN, le G-quadruplex. Les guanines entre elles se disposent de façon à former un plan de quatre guanines à l'aide de liaisons hydrogènes, appelé quartet. L'empilement d'au moins deux quartets forme un G-quadruplex (Figure 2) et ces

structures peuvent être classées en 3 catégories selon leur topologie : parallèle, antiparallèle ou hybride (cf Figure (2)).



Représentation d'un G-quadruplex Topologies des G-quadruplex

Figure 2

Initialement étudiées in vitro, des études ont ensuite montré que ces structures secondaires existaient in vivo et ont suggéré qu'il pourrait y avoir plus de 350 000 séquences formatrices de G-quadruplexes dans le génome humain [4]. La formation de G-quadruplex a été corrélée à diverses maladies génétiques humaines et au cancer et ces structures sont donc devenues des cibles attrayantes pour les médicaments.
Une description complète de la structure et de la dynamique des G-quadruplexes fournirait des informations fondamentales pour comprendre leur existence et leur fonction. Il existe deux principales méthodes pour étudier ces structures : la cristallographie et la spectroscopie RMN. Une des limites de la cristallographie est qu'il peut être difficile de créer un cristal et dans notre cas précis il semble presque impossible d'observer un G-quadruplex avec une topologie hybride ou antiparallèle (la topologie parallèle serait physiquement privilégiée lors de la création d'un cristal). Il est alors intéressant d'utiliser la spectroscopie RMN, qui permet non seulement la détermination haute résolution de la structure, mais aussi l'étude cinétique et dynamique ainsi que les interactions moléculaires [3]. La spectroscopie RMN repose sur les propriétés magnétiques de certains noyaux atomiques : lorsqu'on excite des noyaux de spin non nuls avec un champ magnétique on peut observer une fréquence de résonance, liée au champ magnétique, au noyau et à son environnement.

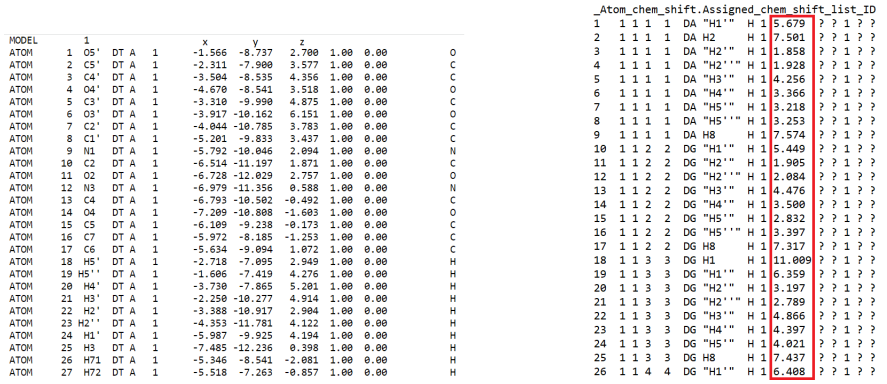
1.2 Objectif du stage

Lors de l'analyse d'un G-Quadruplex par spectroscopie RMN, la tâche d'attribution des pics de résonance (trouver les atomes responsables de chaque pic) est complexe et très chronophage (de l'ordre de plusieurs mois). Le but de ce stage est d'aider à cette attribution, que ce soit de manière "exacte" et automatique ou bien probabiliste dans le but de donner des indices aux biologistes. Ce projet est exploratoire du fait de la faible bibliographie sur la seule utilisation d'un spectre RMN 1D. En effet, les seules méthodes utilisées sont le deep learning, avec notamment l'utilisation de réseaux convolutifs [6], mais ce type de méthode peut montrer quelques limites avec peu de données. En pratique on peut également disposer de spectres 2D ou 3D afin d'établir des corrélations spatiales entre différents pics et des méthodes de machine learning existent déjà pour traiter ces types de spectres [9].

L'objectif sera donc de répondre aux questions : est-il possible de classifier de manière automatique les différentes topologies de G-Quadruplexes en utilisant seulement un spectre RMN 1D ? Peut on aider à l'attribution des pics de résonance ?

2 Création de la base de données

Depuis quelques années, une base de donnée en ligne nommée *Protein Data Bank* (*PDB*) [2] recense un grand nombre de protéines analysées par les biologistes et met en libre accès leur séquence, leur structure 3D, leurs déplacements chimiques ainsi que d'autres observations (cf Figure 3).
On trouve sur cette base de données un peu plus de 200 G-Quadruplexes et on peut donc importer toutes les données nécessaires.



```
MODEL         1                                                    
ATOM      1  O5'  DT A   1      -1.566  -8.737   2.700  1.00  0.00           O
ATOM      2  C5'  DT A   1      -2.311  -7.900   3.577  1.00  0.00           C
ATOM      3  C4'  DT A   1      -3.504  -8.535   4.356  1.00  0.00           C
ATOM      4  O4'  DT A   1      -4.670  -8.541   3.518  1.00  0.00           O
ATOM      5  C3'  DT A   1      -3.310  -9.990   4.875  1.00  0.00           C
ATOM      6  O3'  DT A   1      -3.917 -10.162   6.151  1.00  0.00           O
ATOM      7  C2'  DT A   1      -4.044 -10.785   3.783  1.00  0.00           C
ATOM      8  C1'  DT A   1      -5.201  -9.833   3.437  1.00  0.00           C
ATOM      9  N1   DT A   1      -5.792 -10.046   2.094  1.00  0.00           N
ATOM     10  C2   DT A   1      -6.514 -11.197   1.871  1.00  0.00           C
ATOM     11  O2   DT A   1      -6.728 -12.029   2.757  1.00  0.00           O
ATOM     12  N3   DT A   1      -6.979 -11.356   0.588  1.00  0.00           N
ATOM     13  C4   DT A   1      -6.793 -10.502  -0.492  1.00  0.00           C
ATOM     14  O4   DT A   1      -7.209 -10.808  -1.603  1.00  0.00           O
ATOM     15  C5   DT A   1      -6.109  -9.238  -0.173  1.00  0.00           C
ATOM     16  C7   DT A   1      -5.972  -8.185  -1.253  1.00  0.00           C
ATOM     17  C6   DT A   1      -5.634  -9.094   1.072  1.00  0.00           C
ATOM     18  H5'  DT A   1      -2.718  -7.095   2.949  1.00  0.00           H
ATOM     19 H5''  DT A   1      -1.606  -7.419   4.276  1.00  0.00           H
ATOM     20  H4'  DT A   1      -3.730  -7.865   5.201  1.00  0.00           H
ATOM     21  H3'  DT A   1      -2.250 -10.277   4.914  1.00  0.00           H
ATOM     22  H2'  DT A   1      -3.388 -10.917   2.904  1.00  0.00           H
ATOM     23 H2''  DT A   1      -4.353 -11.781   4.122  1.00  0.00           H
ATOM     24  H1'  DT A   1      -5.987  -9.925   4.194  1.00  0.00           H
ATOM     25  H3   DT A   1      -7.485 -12.236   0.398  1.00  0.00           H
ATOM     26  H71  DT A   1      -5.346  -8.541  -2.081  1.00  0.00           H
ATOM     27  H72  DT A   1      -5.518  -7.263  -0.857  1.00  0.00           H
```

Fichier positions 3D

```
_Atom_chem_shift.Assigned_chem_shift_list_ID
1   1 1 1  1  DA "H1'"  H 1 5.679 ? ? 1 ? ?
2   1 1 1  1  DA H2     H 1 7.501 ? ? 1 ? ?
3   1 1 1  1  DA "H2'"  H 1 1.858 ? ? 1 ? ?
4   1 1 1  1  DA "H2''" H 1 1.928 ? ? 1 ? ?
5   1 1 1  1  DA "H3'"  H 1 4.256 ? ? 1 ? ?
6   1 1 1  1  DA "H4'"  H 1 3.366 ? ? 1 ? ?
7   1 1 1  1  DA "H5'"  H 1 3.218 ? ? 1 ? ?
8   1 1 1  1  DA "H5''" H 1 3.253 ? ? 1 ? ?
9   1 1 1  1  DA H8     H 1 7.574 ? ? 1 ? ?
10  1 1 2  2  DG "H1'"  H 1 5.449 ? ? 1 ? ?
11  1 1 2  2  DG "H2'"  H 1 1.905 ? ? 1 ? ?
12  1 1 2  2  DG "H2''" H 1 2.084 ? ? 1 ? ?
13  1 1 2  2  DG "H3'"  H 1 4.476 ? ? 1 ? ?
14  1 1 2  2  DG "H4'"  H 1 3.500 ? ? 1 ? ?
15  1 1 2  2  DG "H5'"  H 1 2.832 ? ? 1 ? ?
16  1 1 2  2  DG "H5''" H 1 3.397 ? ? 1 ? ?
17  1 1 2  2  DG H8     H 1 7.317 ? ? 1 ? ?
18  1 1 3  3  DG H1     H 1 11.009 ? ? 1 ? ?
19  1 1 3  3  DG "H1'"  H 1 6.359 ? ? 1 ? ?
20  1 1 3  3  DG "H2'"  H 1 3.197 ? ? 1 ? ?
21  1 1 3  3  DG "H2''" H 1 2.789 ? ? 1 ? ?
22  1 1 3  3  DG "H3'"  H 1 4.866 ? ? 1 ? ?
23  1 1 3  3  DG "H4'"  H 1 4.397 ? ? 1 ? ?
24  1 1 3  3  DG "H5'"  H 1 4.021 ? ? 1 ? ?
25  1 1 3  3  DG H8     H 1 7.437 ? ? 1 ? ?
26  1 1 4  4  DG "H1'"  H 1 6.408 ? ? 1 ? ?
```

Fichier déplacements chimiques annotés

Figure 3: Exemples de fichiers PDB.

2.1 Observation des structures 3D et création d'un algorithme automatique de classification

Une première difficulté se présente: les fichiers PDB des G-Quadruplexes ne font pas mention de leur topologie, sauf pour quelques cas où on peut la retrouver dans le titre. Il nous faut donc trouver un moyen de créer les labels de façon automatique. La topologie étant décidable à l'œil en observant la structure 3D du G-Quadruplexe, il est donc naturel d'importer les fichiers contenant les positions 3D de tous les atomes et de créer un algorithme à partir de ces données. Une première idée a été d'utiliser des algorithmes de *Topological Data Analysis*, mais ceux-ci se sont avérés peu pertinents dans cette tâche du fait de la grande variabilité de structure des G-Quadruplexes (nombre de nucléotides différents, colonnes de longueur et forme différentes etc). Je suis rapidement arrivé à la conclusion qu'utiliser la position de tous les atomes n'était pas pertinent sachant que seuls les nucléotides **G** et le sens des colonnes apportent de l'information sur la topologie.
Un article récent utilisant du machine learning pour classifier les G-Quadruplexes à partir des spectres RMN annotés [12] montre la pertinence de l'observation des **G** dans les tétrades. En effet, les nucléotides **G** existent dans 2 conformations : *syn* et *anti*. En calculant le ratio $\frac{n_{syn}}{n_{anti}}$, on peut classifier (à l'aide d'un arbre de décision et d'un SVM comme montré dans [12]) les G-Quadruplexes selon leur topologie.

Cependant, l'article soulève quelques exemples pour lesquels cette méthode ne fonctionne pas du fait de l'existence d'autres **G** en dehors des tétrades ou du fait de l'absence d'un ou plusieurs **G** dans les tétrades.
Une manière de résoudre ce problème est dans un premier temps d'identifier les tétrades en utilisant les positions 3D des atomes de chaque nucléotide **G**. En observant les G-Quadruplexes en 3D, on remarque que la distance $\overline{O6\ H1}$ entre deux **G** est discriminante: pour deux **G** voisins dans la même tétrade cette distance sera courte. On voit en Figure (4) qu'il existe deux distributions séparées représentant les distances entre **O6** d'un **G** et **H1** d'un autre. On dispose donc d'un seuil permettant de construire chaque tétrade.

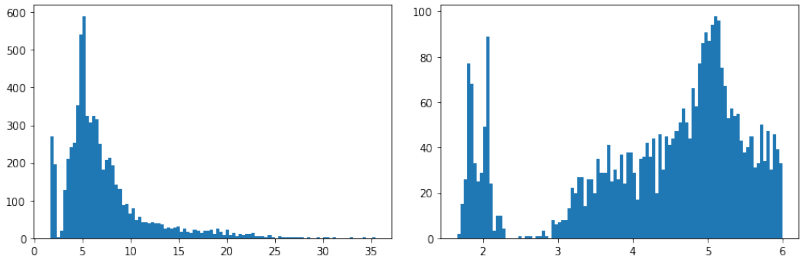


Figure 4: Distribution des distances $\overline{O6\ H1}$.

Maintenant que nous pouvons trouver les tétrades, il s'avère que l'on peut directement trouver la topologie en observant le sens de lecture des **G** dans chaque tétrade. En effet, les nucléotides étant numérotés selon leur ordre d'apparition dans la séquence ADN, en observant les numéros dans chaque tétrade on peut savoir si une colonne est orienté vers le "haut" ou vers le "bas". Par exemple, supposons que nous identifions les tétrades suivantes : Tétrade 1 : $\{1, 9, 13, 21\}$, Tétrade 2: $\{2, 8, 14, 20\}$, Tétrade 3: $\{3, 7, 15, 19\}$. L'ordre de lecture des **G** en partant de la tétrade 1 jusqu'à la tétrade 3 est : $1 \rightarrow 2 \rightarrow 3$; $9 \rightarrow 8 \rightarrow 7$; $13 \rightarrow 14 \rightarrow 15$; $21 \rightarrow 20 \rightarrow 19$. On observe que 2 lectures se font dans l'ordre croissant (i.e vers le haut) et les 2 autres dans l'ordre décroissant (i.e vers le bas). On est donc ici dans une topologie anti-parallèle, et on peut trancher pour chaque G-Quadruplexes de cette façon. On dispose donc d'un algorithme déterministe, sans utilisation de machine learning, pour créer le label de nos données.

Connaître la conformation des **G** dans les tétrades n'était finalement pas nécessaires, mais ces informations pourront nous aider pour la suite car l'article précité [12] montre que la conformation a une influence sur les pics de résonance. Il peut donc être intéressant de rajouter la conformation de chaque **G** dans notre label. En observant de nouveau la structure 3D des **G** *anti* et *syn*, on remarque que la distance $\overline{H1'\ H8}$ permet de séparer les 2 conformations, comme montré en Figure (5).

Nous disposons donc d'un algorithme de création automatique de label basé sur la position 3D des atomes, qui nous renvoie ce genre d'information:

Tétrade 1 : 1 (syn) 9 (anti) 13 (syn) 21 (anti)
Tétrade 2 : 2 (anti) 8 (syn) 14 (anti) 20 (syn)
Tétrade 3 : 3 (anti) 7 (syn) 15 (anti) 19 (syn)

$\Rightarrow$ Anti-parallèle

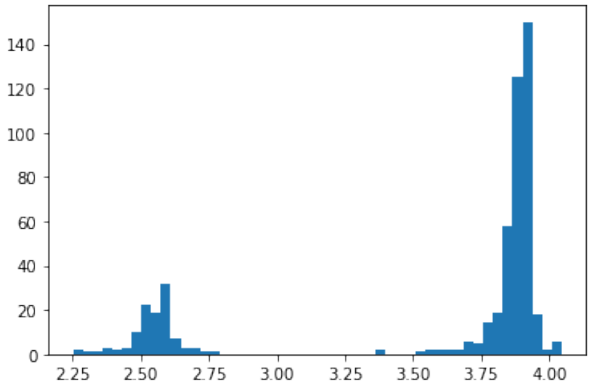


Figure 5: Distribution des distances $\overline{H1' \; H8}$.

2.2 Difficultés

Une fois que nous avons attribué un label à nos données, nous pouvons importer le spectre RMN 1D pour construire notre data set. Nous rencontrons ici un réel problème: le nombre initial de G-Quadruplexes recensés sur PDB était déjà relativement faible (dans le but d'entraîner un algorithme d'apprentissage automatique), mais de plus nous disposons des spectres RMN 1D de seulement la moitié de ces G-Quadruplexes. Nous tombons donc à 120 données ce qui rendra la tâche d'autant plus difficile. De plus, les données ne sont évidemment pas parfaite et on observe que certains spectres ne sont pas complets: certains pics ne sont pas renseignés ce qui endommage fortement le spectre.
De plus, la classe des G-Quadruplexes parallèles est majoritaire et représente 50% des données comme on peut le voir sur la Figure (6).

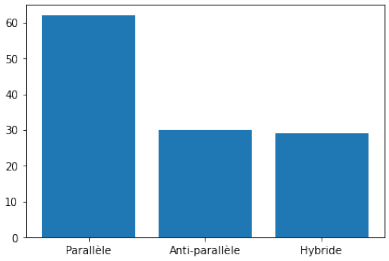


Figure 6: Nombre de spectres pour chaque classe.

3 Classification des G-Quadruplexes à l'aide des spectres RMN 1D

3.1 Représentation des données

Les spectres RMN 1D sont des ensembles de pics de fréquences (que nous observerons entre 0 et 12 ppm). Expérimentalement nous obtenons des spectres avec des pics d'intensités et de largeurs différentes (cf Figure 7) et la première étape (qui ne nous intéresse pas ici) est d'en extraire des pics.

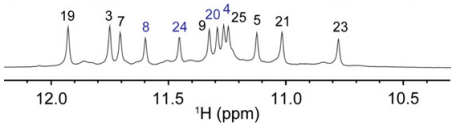


Figure 7: Exemple de spectre RMN.

Chaque pic correspond à la fréquence de résonance d'un atome. Nous devons dans un premier temps décider de la représentation informatique de ce spectre. Deux principales solutions s'offrent à nous :

- Représenter un spectre comme une liste de fréquences.
- Séparer le spectre en n intervalles et représenter un spectre comme l'histogramme sur ces intervalles.

Chacune de ces deux solutions a ses avantages et inconvénients.

- Représenter un spectre comme une liste de fréquence a l'avantage de conserver une information complète mais pose 2 principaux problèmes. Premièrement le nombre de pics varie selon les spectres (séquences d'ADN plus longues, ou spectres incomplets) et cela est dérangeant notamment lors de l'utilisation de réseaux de neurones. On pourrait résoudre cela en faisant du "padding", c'est à dire en rajoutant des 0 à notre liste pour que tous les spectres aient la même taille. Le deuxième problème réside dans le fait que le nombre de pics influe sur la position d'un pic dans la liste. En effet, un pic à 8ppm qui peut être lié au **G** d'une tétrade pourrait se retrouver en 50ème position pour un spectre et en 110ème pour un autre (en supposant que les listes soient triées) rendant difficile l'apprentissage pour un réseau de neurones.
- L'avantage de l'histogramme est qu'il permet de créer des vecteurs d'entrée de même taille en préservant une structure spatiale dans le domaine des fréquences : le i-ème élément du vecteur d'entrée sera toujours responsable du même intervalle de fréquences. Le problème de cette représentation réside dans le choix du nombre d'intervalles. En effet, si des écarts de l'ordre du centième de ppm sont pertinents pour la classification, nous devons donc séparer en des intervalles de cet ordre là. Cependant nous aurions un vecteur d'entrée de taille $12 \times 100 = 1200$, ce qui est beaucoup trop grand lorsqu'on dispose seulement de 120 spectres. A contrario, un échantillonnage moins précis pourra peut-être nous faire passer à côtés de features discriminantes. Il y a donc un juste milieu à trouver.

Nous travaillerons principalement avec des histogrammes, excepté lorsque nous calculerons des features.

3.2 Observations

Commençons par observer nos données pour voir si des différences dans les spectres de G-Quadruplexes de topologie différentes sautent aux yeux. Nous utilisons la représentation par histogramme afin de voir plus facilement si des zones sont denses en pics. Les spectres obtenus en Figure (8) ont été tracés en séparant l'intervalle $[0, 12]$ en 1000 intervalles de même longueur.

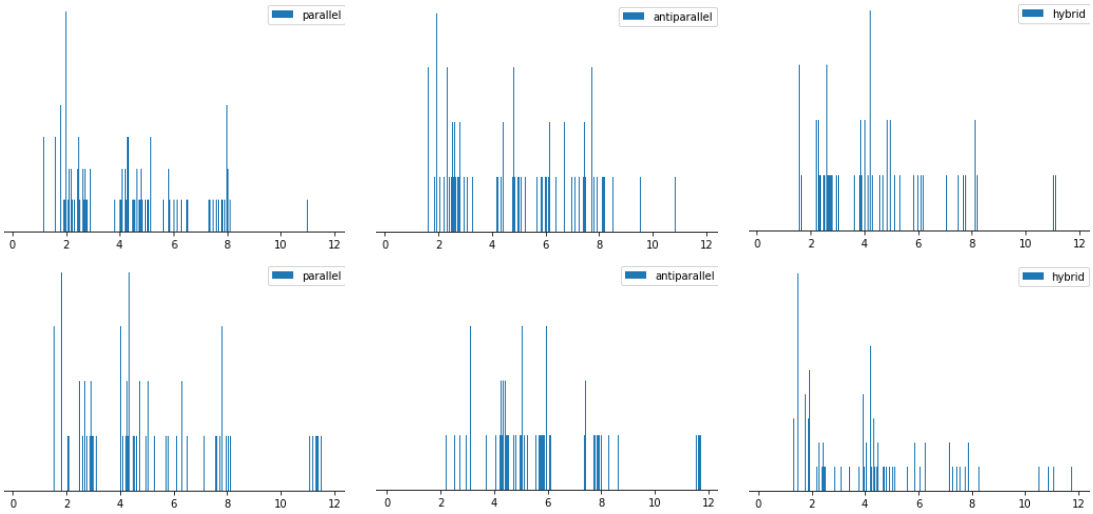


Figure 8: Deux exemples de spectres RMN 1D pour chaque topologie.

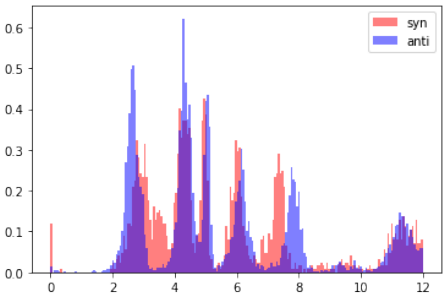
A première vue, il n'y a pas d'allure générale pour chaque topologie et au sein d'une même classe de topologie les spectres peuvent énormément varier. Cette observation a été renforcée par une analyse plus approfondie du data set. Nous avons précédemment vu que les **G** de la tétrade nous apportent les informations nécessaires pour la classification des G-Quadruplexes. Il est donc intéressant de regarder les spectres en différenciant les pics issus d'un nucléotide **G** et les autres. On remarque immédiatement que la zone $[10, 12]$ ppm contient des pics de fréquences exclusivement issus des nucléotides **G**. Ces pics sont dûs aux imino protons présent dans les guanines, il peut donc être intéressant de se focaliser sur cette partie du spectre.
Cependant, une fois de plus, aucune allure générale propre à chaque topologie n'apparaît. On pourra éventuellement essayer d'utiliser un réseau de neurones sur cette partie du spectre pour voir si une classification est possible. Cependant il faut raffiner les observations et ne pas seulement observer une allure générale.

Peu (voire pas) de bibliographie existe sur ce type de classification à partir de spectre RMN 1D. Le seul papier existant est celui dont nous avons déjà parlé et qui utilisait les conformations *syn* et *anti*. La classification est donc possible si l'on connaît le nombre de **G** avec chaque conformation. Cependant, contrairement à cet article, je ne dispose pas des annotations des pics, le but final étant justement l'attribution des pics de résonance. De plus, la séparation *syn*/*anti* était rendue possible par la connaissance des fréquences de résonance du carbone 13, fréquences dont nous ne disposons pas.

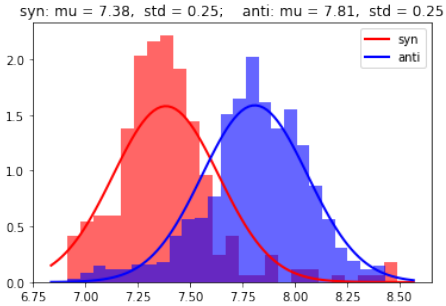
Observons les contributions des **G** *syn* et *anti* de toutes nos données pour savoir si il est possible de les différencier.

On voit sur la Figure (9) qu'il existe principalement 2 zones qui pourraient nous permet-

tre de connaître le nombre de *syn* et *anti*. En effet on voit une nette séparation des contributions des syn et anti sur les intervalles $[2, 4]$ et plus particulièrement $[7, 8.5]$. De plus la zone $[6, 10]$ est intéressante car elle correspond aux protons aromatiques qui sont simples à étudier expérimentalement.



Contributions des G syn et anti



Contributions des G syn et anti sur l'intervalle $[7, 8.5]$

Figure 9: Attributions pour 3 exemples (parallèle , antiparallèle , hybride).

Nous faisons alors face à un dernier problème: les contributions des autres nucléotides peuvent se mêler dans ces intervalles et donc jouer le rôle de bruit. Malheureusement ce "bruit" représente plus de 50% de nos données comme illustré en Figure (10). Il sera donc nécessaire de traiter ce bruit avant de pouvoir utiliser les contributions des **G** pour la classification.

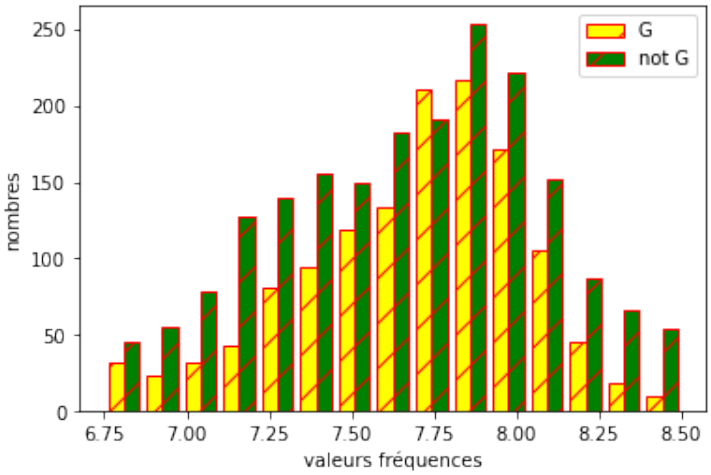


Figure 10: Contributions des G et des autres nucléotides sur $[7, 8.5]$.

Nous allons maintenant voir, discuter et comparer plusieurs méthodes de classification.

3.3 Calcul de features et classification

Une première idée pour la classification est l'utilisation de l'expertise d'un biologiste. En effet, mon maître de stage est familier avec ce type de données et est donc capable d'émettre des hypothèses en faveur d'une classe lors de l'analyse d'un spectre. Un bon moyen pour traduire cette expertise en algorithme de machine learning est le calcul des features. Par exemple un spectre aux pics resserrés ou écartés feront pencher vers une classe ou une autre, idem pour la largeur de bande des fréquences. Après discussion avec mon maître de stage, les premières idées de features portaient donc sur les écarts entre les pics (écart maximum/minimum ou écart moyen) ainsi que sur des densités relatives de pics dans certaines zones.

On peut donc séparer l'intervalle de fréquence $[0, 12]$ en n intervalles de même longueur afin de calculer les énergies relatives de chaque intervalle. En pratique je séparerai en 24 intervalles car lors de l'observation des données j'ai pu constater que des zones discriminantes avaient une variance de l'ordre de 0.25, donc observer des intervalles de taille 0.5 peut être pertinent.
Je rajoute à ces features la densité sur l'intervalle $[\mu - \sigma^2/5, \mu + \sigma^2/5]$ avec $\mu = 7.38$ et $\sigma^2 = 0.25$ (cf Figure(9)).

En affichant les boxplot on se rend compte qu'un grand nombre de nos features de densité relative ne sont pas pertinentes (zones sans présence de pics dûs aux nucléotides **G**), mais certaines semble permettre de différencier certaines classes, notamment les G-quadruplexes parallèles des autres topologies comme on peut le voir sur la Figure (11).

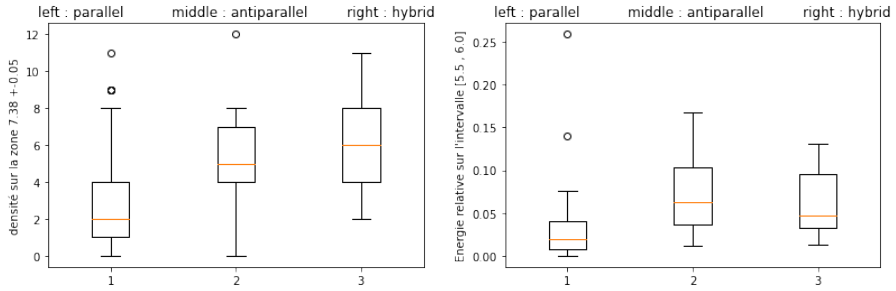


Figure 11: Exemples de boxplot pour 2 features.

3.3.1 Random Forest

Un premier outil de classification est l'arbre de décision. Ce type d'algorithme permet de classifier nos données en plusieurs compartiments à l'aide de features. L'algorithme repose sur le procédé récursif suivant :

- On cherche la feature qui sépare le mieux les données.
- On sépare notre base d'entraînement en utilisant cette feature.
- On réitère ce procédé sur chaque sous ensemble obtenu tant que l'on peut classifier.

En pratique on essaie de minimiser l'indice de diversité de Gini ou l'entropie pour choisir la feature qui sépare le mieux les données.

Supposons que l'on ait m classes numérotées de 1 à m et que nous avons séparés l'ensemble S de toutes les entrées en 2 sous-ensembles S_G et S_D en utilisant une feature, l'indice de diversité est alors donné par la formule:

$$ID(S) = \frac{|S_G|}{|S|}E(S_G) + \frac{|S_D|}{|S|}E(S_D) \tag{1}$$

où $E(X)$ est donné par:

- Indice de diversité de Gini: $E(X) = \sum_{i=1}^{m} p_i(1 - p_i)$
- Entropie: $E(X) = \sum_{i=1}^{m} p_i log_2(p_i)$

où p_i est le ratio entre les éléments de la classe i dans X et tous les éléments de X.

Les algorithmes "Random Forest" [5] sont une amélioration des arbres de décision et sont très efficaces pour la classification.
Ces algorithmes consistent à créer plusieurs arbres de décisions indépendants (chacun ayant une vision restreinte du problème : nombre restreint d'échantillons et de features) puis à classifier en procédant à un vote majoritaire parmi tous les arbres.

Après plusieurs simulations afin de trouver les hyper paramètres les plus efficaces (nombre d'abres, profondeur d'arbre maximale), on peut voir les résultats d'accuracy en Figure (12). Pour obtenir ces résultats, j'ai utilisé 90 données pour l'entraînement et 30 pour le test, en séparant aléatoirement. Au vu du faible nombre de données dont nous disposons, le test a été réalisé 1000 fois afin de constater la sensibilité du modèle aux données d'entraînement. Ces test nous permettent d'avoir une idée du comportement général de cet algorithme et de ses limites.

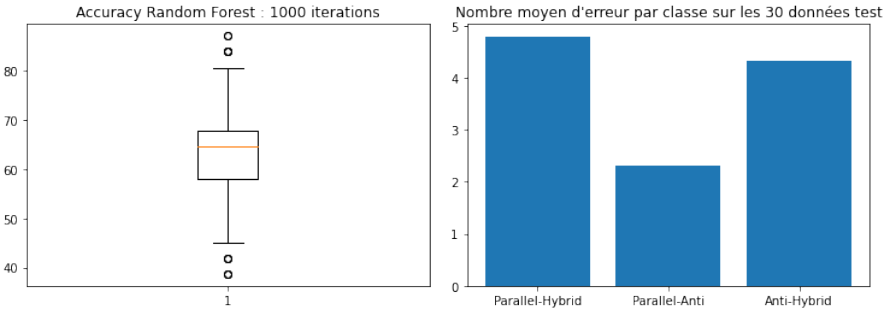


Figure 12: Accuracy et erreurs Random Forest.

On constate que l'accuracy moyenne est de 65% et qu'en moyenne les Random Forest séparent bien les topologies parallèles et anti-parallèles. La difficulté semble provenir des hybrides, ce qui semble logique puisque cette topologie (3 colonnes vers le haut) est intermédiaire entre parallèle (4 colonnes vers le haut) et anti-parallèles (2 colonnes vers le haut).
On remarque également que cette méthode est sensible aux données dans le set d'entraînement.

En effet, on peut passer d'une accuracy de presque 90% à seulement 40%.

Un avantage de cette méthode est que la sortie est une distribution de probabilités sur chaque classe. On peut donc s'intéresser aux probabilités en sortie de l'algorithme pour voir dans quelle mesure il se trompe lors d'une fausse classification ou inversement si il est sûr de lui quand il réussit.

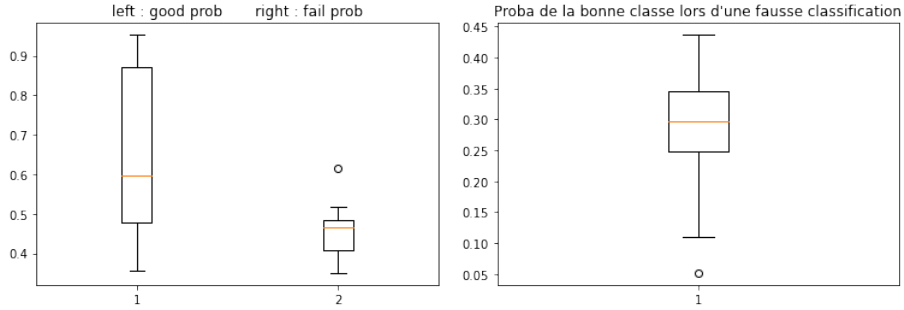


Figure 13: Probabilité en sortie de la Random Forest. A gauche on compare les probabilités lors d'une fausse ou bonne classification. A droite on regarde la probabilité de la bonne classe lors d'un échec.

Les résultats présentés en Figure (13) montrent que l'algorithme fonctionne plutôt bien, puisque l'algorithme donne une probabilité élevée à la bonne classe lorsqu'il classifie bien (0.6 en moyenne) et est plus réservé lorsqu'il se trompe (moins de 0.5). De plus, on peut voir que la probabilité de la bonne classe quand il se trompe reste relativement haute puisqu'elle est de l'ordre de 0.3 en moyenne.
On présente en Figure (14) l'évolution de l'accuracy et du ratio d'échantillons classifiés lorsqu'on décide de classifier si et seulement si la probabilité maximale est supérieure à un certain seuil.

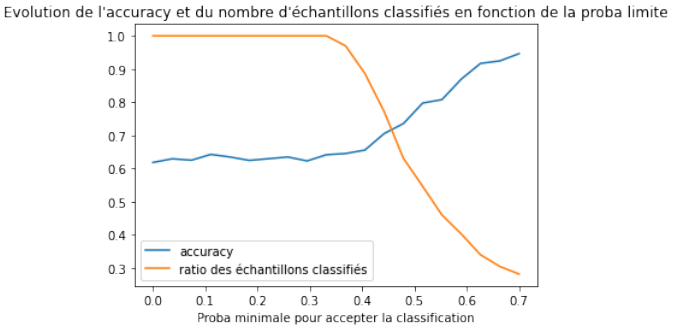


Figure 14: Proportion d'échantillons classifiés et accuracy sur ces échantillons en fonction du seuil de probabilité maximale.

Une classification aléatoire donne une accuracy de 33%, cette méthode est donc en moyenne 2 fois plus performante qu'une classification aléatoire. Cependant, il est nécessaire de rappeler

que la topologie parallèle représente 50% des données et qu'un algorithme renvoyant toujours parallèle aurait donc en moyenne 50% d'accuracy.
65% d'accuracy en moyenne n'est donc pas un très bon résultat, nous devons donc essayer d'autres méthodes pour améliorer les résultats.

3.4 Méthodes statistiques

3.4.1 Rappels des notions principales

Le problème est le suivant : on dispose d'observation de variables aléatoires et on cherche la loi de ces variables. En pratique, on cherchera cette loi parmi un certain ensemble de lois particulières.

Definition 3.4.1. Un modèle statistique est la donnée d'un ensemble de distribution $\mathcal{M} = \{p(.|\theta)\ ;\ \theta \in \Theta\}$. Un modèle sera dit paramétrique si $\Theta \subset \mathbb{R}^d$.
Dans le cas de distributions discrètes, $p(x|\theta) = \mathbb{P}_\theta(X = x)$.

Definition 3.4.2. Soit $X_1, ..., X_n$ des variables aléatoires iid, la vraisemblance en θ d'une observation $(x_1, ..., x_n)$ est la fonction :

$$l_{(x_1,...,x_n)}(\theta) = \prod_{i=1}^{n} p(x_i|\theta)$$

Le but est de trouver θ^* qui maximise la vraisemblance. Dans ce cas, θ^* est appelé le *maximum de vraisemblance*.

3.4.2 Modélisation de notre problème

On dispose de quatre bases **A**, **C**, **G**, **T** qui contribuent au spectre RMN. D'après ce qui précède, l'information essentielle réside dans les pics de fréquences des nucléotides **G**. On veut donc "débruiter" les données en séparant les pics de chaque nucléotides.

Un spectre RMN sera représenté comme un élément de $\mathbb{N}^n$ où n est le nombre d'intervalles qui partitionnent la bande $[0, 12]$.

On considère qu'un G-quadruplexe g est complètement déterminé par sa structure 3D $S(g) \in (\mathbb{R}^3)^{n_{atomes}}$.
En notant $\mathcal{G}4$ l'ensemble des G-Quadruplexes, il existe une fonction qui à chaque G-quadruplexe associe son spectre RMN :

$$\begin{aligned} RMN : \mathcal{G}4 &\longrightarrow \mathbb{N}^n \\ g &\mapsto RMN(g) \end{aligned}$$

Plus précisément, il existe quatre fonctions RMN_A, RMN_C, RMN_G et RMN_T à valeurs dans $\mathbb{N}^n$ vérifiant :

$$RMN(g) = \sum_{J \in \{A,C,G,T\}} RMN_J(g).$$

Ces quatre fonctions représentent les contributions de chaque base au spectre RMN.

De plus, il existe des fonctions $T : \mathcal{G}4 \longrightarrow \{0, 1, 2\}$ donnant la topologie d'un G-quadruplexe et n_A, n_C, n_G, n_T donnant respectivement le nombre de nucléotides **A**, **C**, **G** et **T**).

Ici, étant donné un G-quadruplexe g, on souhaite obtenir sa topologie $T(g)$ mais nous avons seulement accès à $RMN(g)$, $n_A(g)$, $n_C(g)$, $n_G(g)$ et $n_T(g)$.

Nous ferons par la suite l'hypothèse que pour chaque t dans $\{0, 1, 2\}$ (parallèle, hybride ou antiparallèle), pour chaque J dans $\{A, C, G, T\}$, il existe des distributions $(p_{J,t}(i))_{i\in[|1,n|]}$ qui représentent la distribution des pics de fréquence pour chaque base dans chaque classe de topologie.
La donnée de $RMN_J(g)$ sera donc vu comme des observations de variables aléatoires iid suivant la loi donnée par $(p_{J,t}(i))_{i\in[|1,n|]}$ pour un certain t.

Finalement, on suppose que chaque nucléotide J est responsable d'un nombre fixe $f(J)$ de pics dans le spectre RMN.

La donnée de $RMN(g)$ sera donc vu comme N observations de variables aléatoires iid suivant la distribution mélangée suivante :

$$(p_t(i))_{i\in[|1,n|]} = \left(\frac{\sum\limits_{J\in\{A,C,G,T\}} n_J(g) f(n_J) p_{J,t}(i)}{\sum\limits_{J\in\{A,C,G,T\}} n_J(g) f(n_J)} \right)_{i\in[|1,n|]}$$

pour un certain $t \in \{0, 1, 2\}$, avec $N = \sum\limits_{J\in\{A,C,G,T\}} n_J(g) f(n_J)$.

Notre modèle statistique est donc $\mathcal{M} = \{p_0, p_1, p_2\}$.

3.4.3 Classification basée sur le modèle statistique.

D'après la définition du modèle précédent, nous devons connaître les distributions $p_{J,t}$ pour $J \in \{A, C, G, T\}$ et $t \in \{0, 1, 2\}$. Nous allons séparer nos données en 2 parties: une partie pour l'entraînement et une pour le test.

Le sous ensemble destiné à l'entraînement nous servira à approcher les distributions $p_{J,t}$ en considérant les distributions empiriques $\hat{p}_{J,t}$. Ensuite, pour chaque spectre test, nous calculerons sa vraisemblance pour les distributions empiriques $\hat{p}_t$, $t \in \{0, 1, 2\}$. La classe assignée sera celle pour laquelle la vraisemblance est maximale.

Les distributions empiriques calculées à partir d'un data set d'entraînement de 90 spectres sont présentés en Figure (17).

Les résultats d'accuracy avec cette méthode sont nettement meilleurs qu'avec les Random Forest. En effet, on constate en Figure (16) qu'on atteint une accuracy de 75% en moyenne, avec une bonne précision pour les topologies parallèles et antiparallèles. Une fois de plus, c'est la topologie hybride qui est la plus difficile à classifier.

Contrairement à la méthode précédente pour laquelle le principal problème venait du fait que les features calculées ne séparaient pas convenablement les différentes topologies, ici le principal

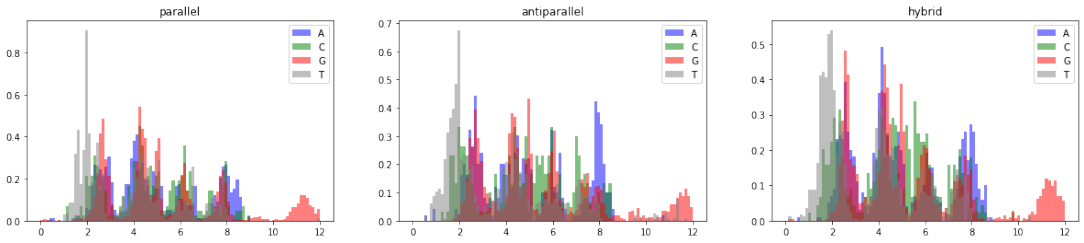


Figure 15: Distributions empiriques des pics de chaque nucléotide pour les différentes topologies.

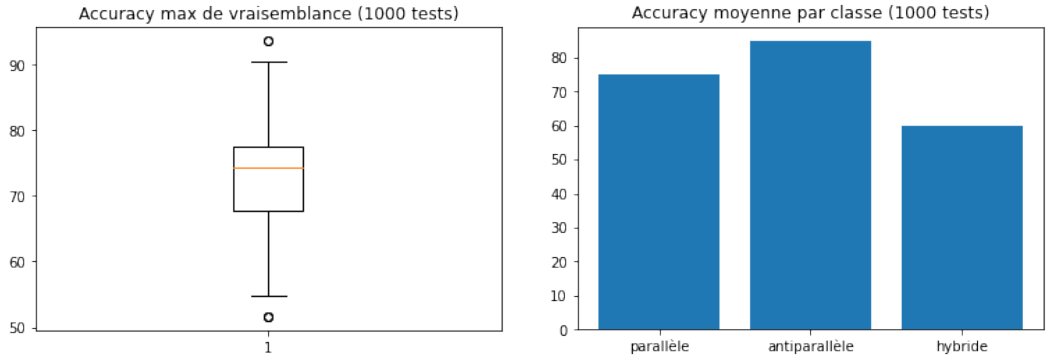


Figure 16: Accuracy moyenne de la méthode du maximum de vraisemblance.

frein est le nombre de données. En effet les distributions pour lesquelles sont calculés les vraisemblances sont obtenus empiriquement avec seulement 90 spectres. En comptant une moyenne d'environ 200 pics par spectres, on dispose de 18000 observations pour construire 12 distributions (une distribution par nucléotide et par topologie, soit entre 1000 et 2000 observations par distribution (le nombre d'observation n'est pas équitable entre les différentes classes et les différents nucléotides) ce qui est relativement faible quand on considère des distributions discrètes sur 50 ou 100 éléments.
Cette méthode peut donc avoir du potentiel si l'on a un jeu de données plus grand. Afin de savoir si la sensibilité du modèle (de 50% à 90% d'accuracy comme indique la Figure (16)) vient de l'entraînement (prise en compte ou non de la variété des données dans le data set d'entraînement) ou du test (plus ou moins de chance dans le test avec des données plus ou moins propres), nous allons quantifier la différence entre les distributions calculées pour différents data set d'entraînement.
Nous allons également calculer la différence des distributions entre les différentes topologies.
Nous allons pour cela calculer une distance sur l'ensemble des distributions : la distance de Wasserstein.

Definition 3.4.3. Soit X, Y deux variables aléatoires de lois respectives P et Q. Notons $\mathcal{J}(P,Q)$ l'ensemble des lois jointes pour (X,Y) qui ont pour marginales P et Q.
La distance de Wasserstein d'ordre p entre P et Q est alors définie par:

$$W_p(P,Q) = \left(\inf_{J \in \mathcal{J}(P,Q)} \int ||x-y||^p dJ(x,y) \right)^p$$

Cette distance est souvent décrite par analogie avec des tas de sables. Si nos deux distributions sont décrites par des tas de sable, cette distance correspond au travail minimal nécessaire pour passer d'un tas à un autre.
L'avantage de cette distance est qu'elle tient compte de la géométrie de l'espace. En effet, supposons que l'on ait trois mesures sur $\mathbb{N}$, par exemple un dirac en 0, un dirac en 1 et un dirac en 10. Il semble naturel que δ_1 soit plus proche de δ_0 que de δ_{10}, pourtant de nombreuses distances (distance L^2, variation totale, Hellinger) n'en tiennent pas compte et renverrons que ces trois distributions sont équidistantes.

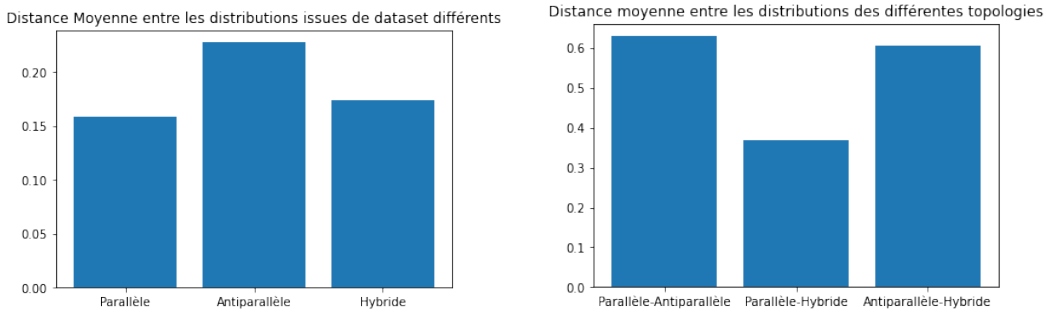


Figure 17: Distances de Wasserstein entre les différentes distributions.

De façon plutôt étonnante, la distribution variant le plus avec le dataset d'entraînement est celle pour la topologie antiparallèle, alors que c'est cette même topologie qui est la mieux classifiée par la méthode du maximum de vraisemblance. Cependant, la Figure (17) montre aussi que la distribution de la topologie antiparallèle est très éloignée des distributions des autres topologies ce qui compense la variance élevée issue de la sensibilité aux données d'entraînement.
A contrario, les distributions des topologies parallèles et hybrides varient moins pour différents entraînements, mais ont des distributions globalement assez proches, ce qui explique le grand nombre d'erreurs de classification sur ces deux classes.

En augmentant le nombre de données, on peut espérer accroître le score d'accuracy pour la topologie antiparallèle, mais la séparation parallèle-hybride restera sûrement plus complexe.

3.5 Réseaux de neurones

3.5.1 Rappels

Nous avons jusqu'ici utilisé deux types de méthodes : les Random Forest (machine learning) et le Maximum de Vraisemblance (méthode statistique). Il existe une dernière méthode que nous pouvons tester ici: les réseaux de neurones profonds [8] [11].
Ce type de méthode est largement utilisée dans tous les domaines de la data science et à juste titre puisqu'elles sont l'état de l'art pour presque tous les problèmes. Le prix à payer est une faible (voire inexistante) interpretabilité du modèle et la nécessité d'avoir un grand nombre de données.

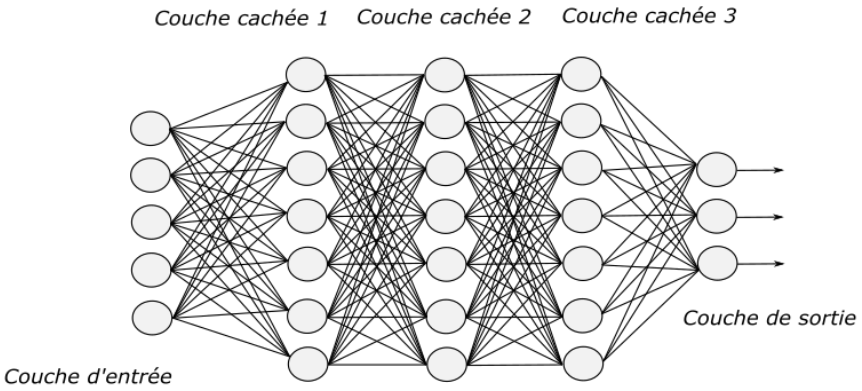


Figure 18: Structure d'un réseau de neurones profond.

Definition 3.5.1. Un réseau de neurones à propagation avant (feedforward neural network) est composé de plusieurs couches de neurones et défini une fonction $F : \mathbb{R}^{n_0} \to \mathbb{R}^m$ de la forme

$$F(x, \theta) = f_{out} \circ g_L \circ f_L \circ ... \circ g_1 \circ f_1(x)$$

où les f_i sont des fonctions linéaires de pré-activation et les g_i sont des fonctions d'activations non linéaires et θ contient les matrices de poids W_l et les vecteurs de biais b_l pour chaque couche. La sortie de la l-ième couche est un vecteur $x_l = (x_{l,1}, ..., x_{l,n_l})$.
Étant donné x_{l-1}, on a $f_l(x_{l-1}) = W_l x_{l-1} + b_l$ où f_l est un vecteur de pré-activation linéaire $(f_{l,1}, ..., f_{l,n_l})$. On a alors:

$$x_{l,i} = g_{l,i}(f_{l,i}(x_{l-1}))$$

Remark 3.5.2. *Nous utiliserons ici la fonction d'activation ReLU :* $g(x) = \max(0, x)$.

L'étape d'apprentissage se fait par propagation arrière du gradient. On dispose d'une fonction de perte $\mathcal{L}$ que l'on veut minimiser : étant donné nos entrées $(x^i)_i$ et leur étiquette de sortie réelle $(y^i)_i$, on veut minimiser la "différence" $\mathcal{L}\left((y^i)_i, (F(x^i, \theta))_i\right)$ entre ces étiquettes et les prédictions du réseau de neurones $(F(x^i, \theta))_i$, par rapport à θ.

On réalise alors simplement une descente de gradient (gradient stochastic descent) sur tous les poids de notre réseau de neurones en utilisant la règle de la chaîne (dérivation de fonctions composées).

Nous allons ici nous intéresser à un modèle plus complexe de réseaux de neurones : les réseaux convolutifs.

3.5.2 Réseaux de neurones convolutifs (CNN)

Un réseau de neurones convolutif [7] se décompose en deux parties : une première dédiée à l'extraction de features, et une deuxième dédiée à la classification. La deuxième partie prend en entrée les features renvoyées par l'extracteur et est composée de plusieurs couches de neurones (entièrement connectées), comme vu précédemment. C'est la première partie qui est originale et qui rend ce type de réseaux si efficace. L'idée est la suivante : on fait glisser un noyau de convolution (une fenêtre de n valeurs en $1D$ ou de (n, m) valeurs en $2D$) le long de notre signal pour créer un nouveau signal. On applique ensuite un "MaxPooling" de paramètre p qui consiste à couper le signal en sous-signaux de taille p et ne conserver que la valeur maximale afin de réduire la dimension.
On peut appliquer ce processus plusieurs fois à la suite, et à chaque étape on peut utiliser différents noyaux de convolution simultanément (cf Figure (19)).

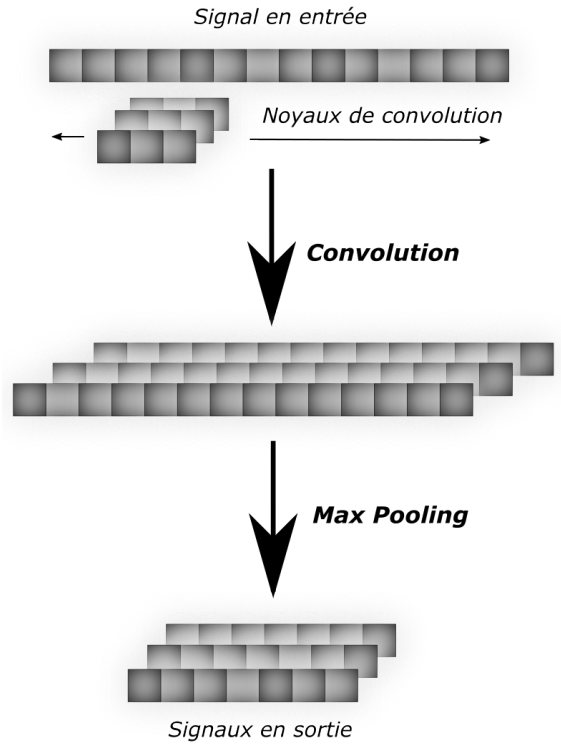


Figure 19: Principe d'un CNN.

Les principaux avantages de ce type de réseaux sont la robustesse à la translation et un nombre réduit de paramètres. En effet les paramètres à entrainer sont les poids dans les noyaux de convolutions, donc pour un signal de 1000 points, on peut par exemple se restreindre à une fenêtre glissante de 10 points et donc diviser par 100 le nombre de paramètres par rapport à un réseau classique.

3.5.3 Application à notre problème.

Dans notre cas l'entrée sera un histogramme des pics (l'intervalle des fréquences est séparé en 100 sous intervalles) et le réseau de neurones retournera 3 valeurs en sortie correspondant aux probabilités d'appartenance à chaque classe.
Afin d'obtenir des probabilités on utilisera la fonction Softmax σ qui à un vecteur (x_1, x_2, x_3) associe :

$$\sigma(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{3} e^{x_j}} \quad , \quad \forall i \in \{1, 2, 3\}.$$

L'entraînement se fera par descente de gradient stochastique et nous minimiserons la "Cross Entropy", définie ci-dessous.

Definition 3.5.3. Etant donné un problème de classification en K classes $\{1, 2, ..., K\}$. Soit x une entrée et c sa classe, pour tout i dans $\{1, ..., K\}$, on définit $y_i = \delta_{i=c}$ (1 si i est la classe de x et 0 sinon). Soit p_i les prédictions de probabilités d'appartenance à la classe i, pour i dans $\{1, ..., K\}$, alors la Cross Entropy est la fonction de perte définie par :

$$\mathcal{L}_{CE}\left(x, (p_i)_i\right) = -\sum_{j=1}^{K} y_j log(p_j).$$

Remark 3.5.4. *La Cross Entropy est seulement égale à un terme, les autres étant tous nuls. En effet, en notant c la classe de x on a :*

$$\mathcal{L}_{CE}\left(x, (p_i)_i\right) = -log(p_c).$$

La fonction à minimiser est donc la moyenne sur toutes les données d'entraînement. Cependant, notre dataset est déséquilibré (60 parallèles, 30 antiparallèles et 30 hybrides) et l'optimisation est biaisée. En effet le réseau de neurones sera plus sensible aux données parallèles et négligera les autres classes.

Dans le but de corriger ce biais, la fonction de perte sera modifiée afin de tenir compte de ce déséquilibre. On attribue donc des poids w_c à chaque classe c et la fonction à minimiser s'écrit donc :

$$\sum_{c=1}^{K} \frac{1}{n_c} \sum_{x \text{ de classe c}} w_c \mathcal{L}_{CE}\left(x, (p_i)_i\right) \tag{2}$$

où $w_c = \frac{n}{K n_c}$, avec n le nombre d'échantillons total et n_c le nombre d'échantillons de classe c, afin de conserver la même masse totale.

Les deux principaux problèmes lorsqu'on utilise des réseaux de neurones pour la classification sont le sur-apprentissage et le sous-apprentissage.
On dit qu'on est en sur-apprentissage lorsque le réseau apprend trop précisément les données d'entraînement et perd donc en pouvoir de généralisation, ce qui implique de mauvais score du les données tests.
Inversement, on est en sous-apprentissage lorsque le réseau s'adapte mal aux données d'entraînement, ce qui implique aussi une mauvaise généralisation et de mauvais scores sur les données de tests.

Au vu du faible nombre de données, on procèdera par validation croisée, c'est à dire qu'on séparera les données d'entraînement en k parties, et alternativement on entraînera sur $k-1$ sous

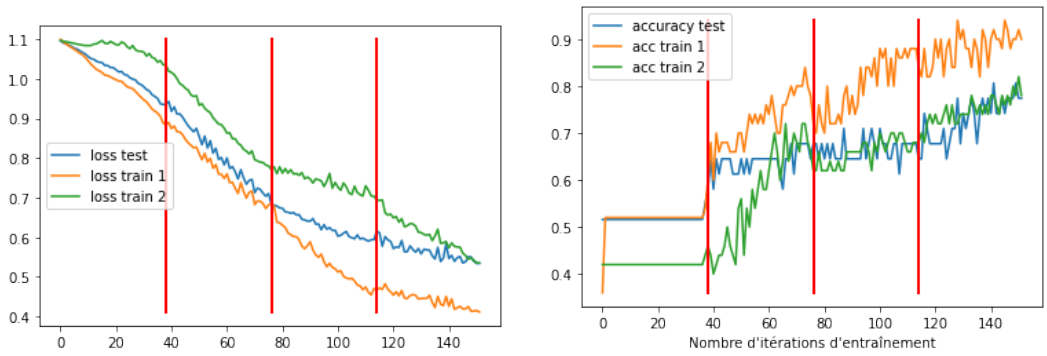


Figure 20: Exemple de bon entraînement.

parties et on validera sur l'autre. Ce procédé permet de vérifier si on est en sur-apprentissage ou en sous-apprentissage et d'avoir une estimation plus fiable de la performance du modèle.

On peut voir en Figure (20) un exemple de bon entraînement. Les droites verticales séparent les entraînements sur les différentes sous parties de la base d'entraînement (2 parties distinctes), et l'on peut voir que la fonction de perte diminue et que le score augmente même lorsqu'on entraîne sur d'autres données : ici le réseau a réussi à trouver de bonnes features et peut généraliser. On obtient un score proche de 75% sur le test ce qui est prometteur, dans l'optique où nous aurions beaucoup plus de données.

Il faut cependant être prudent avec l'utilisation de réseaux de neurones (surtout avec si peu de données) car on peut rapidement faire du sur-apprentissage.

3.5.4 Utilisation de la séquence

Le réseau de neurones utilise comme seule information les spectres RMN pour classifier les G-quadruplexes contrairement à la méthode statistiques qui prend également en compte les nombres de nucléotides.
Une idée pour améliorer l'algorithme de deep learning est d'intégrer au modèle les nombres de nucléotides **A**, **C**, **G** et **T**.

Afin de ne pas noyer l'information apportée par la séquence dans celles apportées par le spectre (4 features contre 100), il est nécessaire de travailler au préalable sur chaque type de données de manière séparée. L'architecture utilisée sera celle présentée en Figure (21) avec n du même ordre de grandeur que m.

Nous procédons encore par validation croisée pour l'entraînement, avec un batch size égal à 3.
Cette architecture est capable de classifier nos données comme le montrent les résultats obtenus ci-dessous.

Les résultats de l'entraînement en validation croisée sont les suivants :

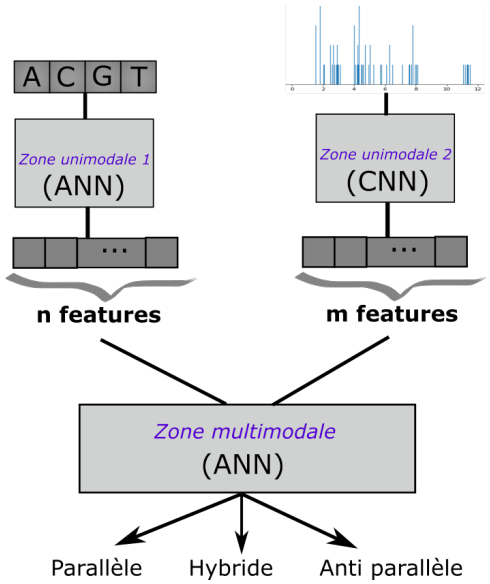


Figure 21: Architecture pour traiter deux types de données.

	Fold 1	Fold 2	Fold 3
Accuracy	89.9%	89.9%	100%
Loss	0.324179	0.307947	0.136527

On atteint sur un entraînement un score de 90% sur les données test (cf Figure (22)) ce qui améliore considérablement les résultats obtenus précédemment.
On constate sur la Figure (23) que le réseau classifie bien les 3 topologies (sur le train et le test) ce qui implique que le minimum obtenu est candidat pour être un minimum global. En effet durant l'entraînement on atteint souvent des minimums locaux consistant à envoyer toutes les entrées sur une seule ou deux classes.

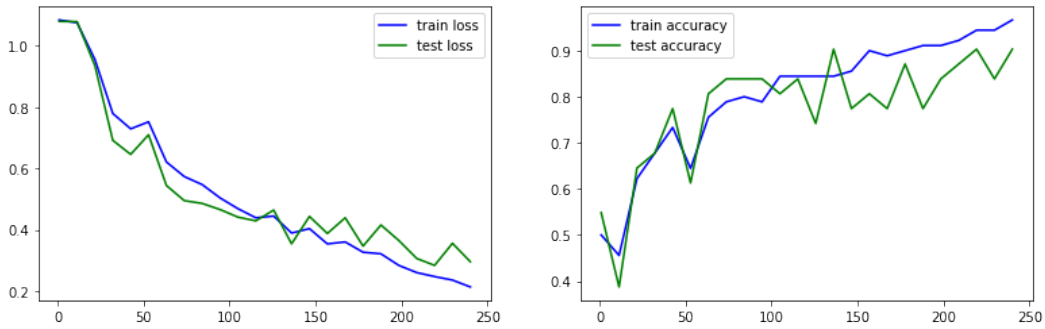


Figure 22: Apprentissage du réseau multimodal.

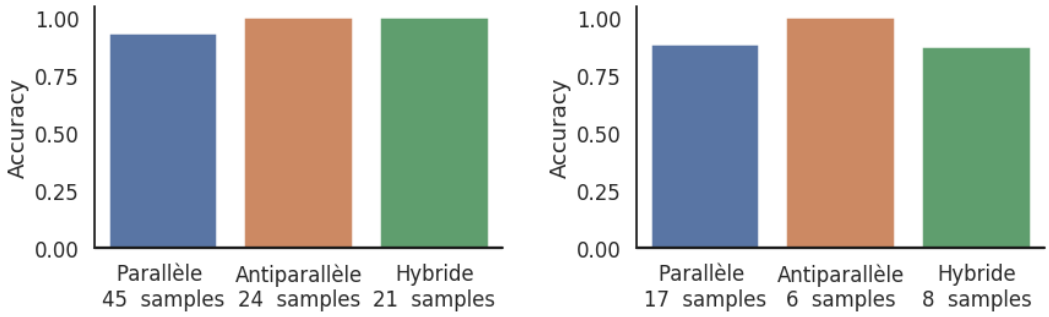


Figure 23: Scores par topologie.

Le score n'est pas la seule donnée intéressante, en effet il est important de connaître les probabilités attribuées à chaque classe pour savoir si la séparation est nette ou si le réseau de neurones est hésitant (i.e distribution proche de l'uniforme sur les 3 classes).
Ici encore (Figures (24) et (25)) les résultats obtenus sur le train et le test sont similaires et on constate une nouvelle fois que les topologies parallèles et antiparallèles sont clairement séparées, contrairement à la topologie hybride qui semble partager des caractéristiques avec les deux autres topologies.

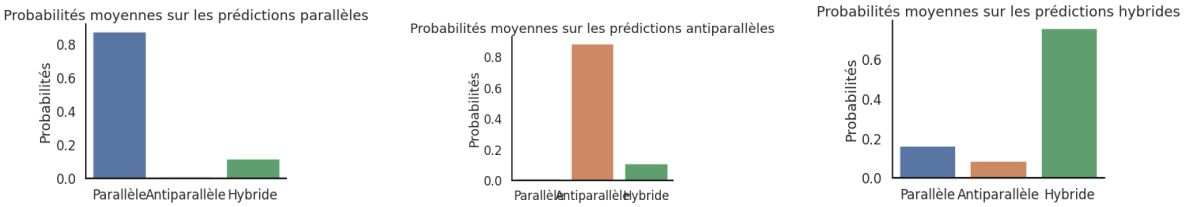


Figure 24: Distributions moyennes en sortie du réseau sur le train.

D'un point de vue biologique, on peut mettre en avant l'argument des conformations des **G** (*syn* et *anti*) : dans la plupart des cas 0 *syn* pour les parallèles, 5 *syn* pour les hybrides et 6 pour les antiparallèles. Plus globalement, la topologie hybride contient deux motifs de "loops" propres aux parallèles et deux autres propres aux antiparallèles, d'où la possible confusion. En effet, le pic d'un atome dans un spectre RMN est influencé par son environnement (spatial) et le type de "loops" a donc une influence sur le spectre. La taille des loops et leurs conformations (diagonal, propeller, loop) expliquent en partie la variété des données et la difficulté à les classifier.

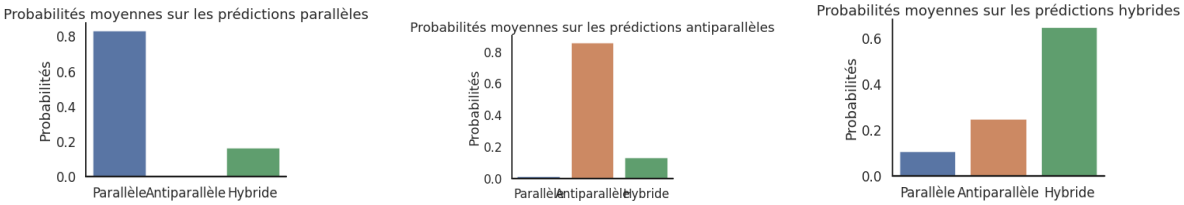


Figure 25: Distributions moyennes en sortie du réseau sur le test.

3.6 Discussion

Trois différents type d'algorithmes IA ont été essayés. Un premier basé sur le calcul de features, un autre basé sur des méthodes statistiques et un dernier utilisant des réseaux de neurones convolutifs.

- On constate que la méthode basée sur le calcul de features (Random Forest) est la moins efficace avec une accuracy moyenne de 65%. De plus cette méthode ne dépend pas vraiment du nombre de données et les résultats ne seraient pas forcément meilleurs avec une plus grande base de données. L'avantage de cette méthode est qu'elle mesure ses réponses et se trompe très peu lorsqu'elle est renvoie une forte probabilité (> 0.75). On pourrait améliorer cette méthode en cherchant et calculant de nouvelles features .

- La méthode statistique est une méthode obtenant de bons résultats, autour de 75% d'accuracy en moyenne et pouvant aller jusqu'à 90% sur certaines base de données. L'avantage de cette méthode est qu'elle est sensible au nombre de données car on utilise les distributions empiriques des pics pour chaque nucléotide de chaque topologie. On peut donc raisonnablement penser que le score augmenterait ou se stabiliserait avec un plus grand nombre de données. L'inconvénient principal est le risque fort de sur-apprentissage, du fait que la méthode soit statistique.

- La dernière méthode testée utilise des réseaux de neurones convolutifs et peut retourner de bons scores selon l'entraînement. Il est difficile de mesurer l'efficacité d'une telle méthode avec notre faible nombre de données. Cependant les résultats sont encourageants car j'ai pu constater sur certains entraînements que l'algorithme peut extraire des features pertinentes et bien classifier ensuite. On peut donc imaginer qu'avec un jeu de données conséquent cet algorithme deviendrait le plus efficace.
  L'inconvénient de ce type d'algorithme est le manque d'interprétabilité. En effet, dans l'optique où il serait très efficace, il agirait comme une boîte noire et nous serions incapable de comprendre ce qui lui permet de trancher, contrairement au Random Forest par exemple.

Le principal problème est la variance des données : après discussion avec mon maître de stage et un autre élève travaillant sur des données biologiques, on comprend qu'il y a presque autant d'exceptions que de règles et qu'il est donc difficile de classifier et d'entraîner un modèle quand des outliers se mêlent aux données plus "propres".
Les réseaux de neurones sont très sensibles à ce genre de phénomène et on ne peut pas savoir comment il va interpréter une exception, je pense donc que la meilleure méthode reste la méthode

statistique utilisant le maximum de vraisemblance car elle obtient de bons score et est plus robuste aux outliers. En effet j'ai sélectionné 10 spectres correspondant à des "exceptions", non inclues dans le dataset de départ, et le score du réseau de neurones tombe à 20% quand la méthode statistique conserve 60% d'accuracy.

4 Attribution des pics de résonance

4.1 Objectif

Afin de pouvoir établir la structure 3D d'un G-Quadruplex, il est nécessaire d'attribuer les pics de résonance. C'est cette tâche qui est fastidieuse et qui peut prendre plusieurs mois. L'objectif ici est d'aider à la décision en renvoyant une attribution de façon automatique ou en donnant des indices comme des probabilités d'attribution.
Pour chaque pic, on doit décider à quel nucléotide l'atome associé correspond (A,C,G,T) puis le nom de l'atome (H1,H1',H2" ...). On doit donc prédire un couple (J, H).

J'ai pu répertorier 45 labels possibles, les algorithmes devront donc choisir une attribution parmi ces 45 couples.

En pratique, le seul spectre RMN 1D n'est pas suffisant pour procéder à l'attribution. En effet, les spectres 2D et 3D [14] donnent des informations précieuses sur des corrélations entre différents pics. On peut donc ici essayer de construire un algorithme le plus précis possible en utilisant seulement un spectre RMN 1D, en imaginant une amélioration qui imposerait des contraintes en analysant des spectres 2D ou 3D, comme on peut le voir dans certaines méthodes réalisant l'état de l'art [9].

4.2 Analyse statistique

Ici nous allons séparer notre jeu de données selon la topologie des G-quadruplexes.
Pour chaque topologie t dans {parallèle, antiparallèle, hybride}, pour chaque base b dans {A,C,G,T}, pour chaque atome d'hydrogène h dans la base b, on va modéliser les pics dûs à cet atome par une distribution normale $\mathcal{N}(\mu_{t,b,h}, \sigma^2_{t,b,h})$.
On rappelle que étant donnés n observations $x_1, ..., x_n$, suivant une loi normale de paramètre (μ, σ), l'estimateur du maximum de vraisemblance est donné par :

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$$
$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})^2$$

On voit sur la Figure (26) que les distributions se chevauchent énormément, rendant l'attribution très compliquée sans information supplémentaire.

De plus, on fait ici implicitement une hypothèse forte : les pics sont mutuellement indépendants. En effet, on considère chaque distribution pour chaque atome indépendante du reste, mais rien ne nous assure que c'est réellement le cas, les pics de différents atomes au sein du même nucléotide pourraient être corrélés et nous pourrions étudier ce point plus en détails.

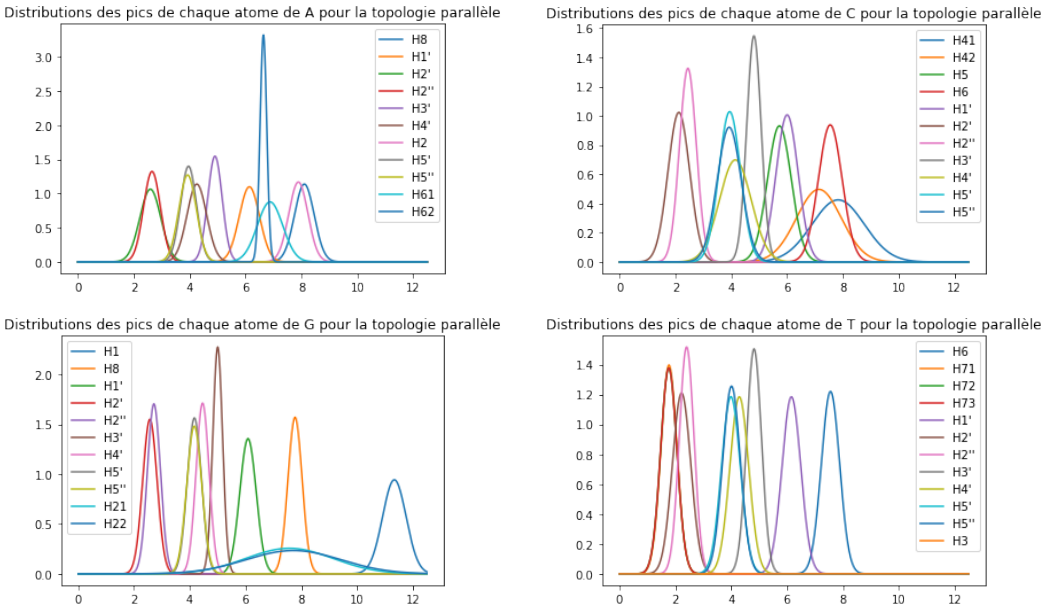


Figure 26: Distribution des pics pour chaque atome.

4.3 Première méthode inspirée de DP4-AI

Un algorithme existant de traitement automatique de spectres RMN 1D nommé DP4-AI [10] effectue une tâche d'attribution basée sur la comparaison d'un spectre avec un spectre créé par un prédicteur (GIAO). Cet algorithme calcule une matrice qui contient les probabilités que le pic i du premier spectre soit lié au pic j du deuxième spectre. Ensuite un problème d'optimisation est résolu afin de maximiser la probabilité de l'attribution en respectant certaines contraintes.
Les premières méthodes que je présente sont inspirées de cet algorithme.

4.3.1 Transport optimal

La comparaison de mesures et le transport d'une distribution à une autre sont des problèmes initialement introduits par Monge au 18è siècle. D'importants développements en théorie du transport sont dûs à Léonid Kantorovitch qui a généralisé le problème de Monge. Nous allons ici formaliser les versions discrètes de ces problèmes que nous appliquerons ensuite.

Definition 4.3.1. (Monge)
Soit une matrice de coût $(C_{i,j})_{i,j\in[|1,n|]}$, le problème de Monge est défini par :

$$\min_{\sigma\in Perm(n)} \sum_{i=1}^{n} C_{i,\sigma(i)} \tag{3}$$

où $Perm(n)$ est l'ensemble des permutations de $\{1, ..., n\}$.

La problème d'optimisation précédent suppose que la matrice de coût soit carré.
Kantorovitch a généralisé ce problème afin de pouvoir résoudre des problèmes d'attribution entre

deux nuages de points de tailles éventuellement différentes.

Nous prenons ici le formalisme probabiliste et considérons 2 mesures $\alpha = \sum_{i=1}^{n} a_i \delta_{x_i}$ et $\beta = \sum_{j=1}^{m} b_j \delta_{y_j}$ avec $\sum_i a_i = a$ et $\sum_j b_j = b$.

Definition 4.3.2. On note $U(a,b) = \left\{ P \in \mathbb{R}^{n \times m} \ | \ P\mathbb{1}_m = a \ , \ P^T \mathbb{1}_n = b \right\}$ l'ensemble des matrices d'attributions conservant les masses.

Definition 4.3.3. Le problème de transport optimal de Kantorovitch s'écrit:

$$L_C(a,b) = \min_{P \in U(a,b)} \sum_{i,j} C_{i,j} P_{i,j}. \tag{4}$$

Proposition 4.3.4. *Dans le cas 1D, si la fonction de coût est de la forme $c(x,y) = |x-y|^p$ avec $P \geq 1$, alors une application du transport optimal est donnée par une fonction croissante.*

4.3.2 Méthode générative

Une première idée est de générer aléatoirement (selon les distributions calculées) des spectres complètement annotés puis de les comparer aux spectres non annotés. L'idée est donc de trouver une correspondance pic par pic entre 2 spectres, l'un annoté et l'autre pas. On pourrait aussi comparer un spectre inconnu à des spectres annotés de notre base de données, mais on perdrait en généralité. Le coté probabiliste et statistique ici permet d'éviter le sur-apprentissage et est moins sensible au bruit.

La création aléatoire d'un spectre se fait en parcourant la liste des nucléotides présents dans la séquence, et pour chaque nucléotide on génère aléatoirement un pic pour chaque atome du nucléotide selon la distribution empirique calculée.

On dispose donc de deux spectres RMN :

$$RMN_{reel} = [f_1, ..., f_{n_1}]$$
$$RMN_{alea} = \left[\hat{f}_1, ..., \hat{f}_{n_2}\right]$$

On supposera dans la suite $n_1 \leq n_2$ car les données de spectre RMN obtenus sur Protein Data Bank sont généralement incomplètes.

On veut donc attribuer chaque pic du spectre réel à un pic du spectre généré, sachant que chaque pic du spectre généré aléatoirement ne peut être utilisé qu'une fois. On modélise cette attribution par le problème d'optimisation suivant :

$$\min_{1 \leq i_1 < i_2 < ... < i_{n_1} \leq n_2} \sum_{k=1}^{n_1} \left| f_k - \hat{f}_{i_k} \right|^{\alpha} \tag{5}$$

avec $\alpha > 0$.

L'attribution découle de la solution du problème précédent en assignant au pic f_k le label du

pic $\hat{f}_{i_k}$ pour tout k.

La résolution d'un tel problème d'optimisation n'est pas triviale. En théorie, le nombre d'attributions est fini donc on peut toutes les tester et sélectionner la meilleure. Cependant le nombre de combinaisons est trop grand (croissance exponentielle en fonction du nombre de pics) et cette méthode devient inutilisable en pratique. En effet, le nombre total d'attributions est donné par :

$$N_{attributions} = \frac{n_2!}{(n_2 - n1)!} \tag{6}$$

Ce problème est similaire aux problèmes de transport optimal (Monge et Kantorovitch) mais diffère dans le sens où $n_1 < n_2$ et on ne sépare pas la masse d'un pic en plusieurs pics : on doit trouver n_1 pics parmi les n_2 et résoudre le problème de Monge.
On pourra aussi résoudre le problème de Kantorovitch associé afin d'avoir plusieurs labels pour chaque pic, ce qui peut être utile en pratique si l'attribution initiale est fausse et que l'on veut la corriger.

Réécrivons le problème sous forme matricielle.
Notons $C = \left(|f_i - \hat{f}_j|^\alpha\right)_{i,j} \in \mathcal{M}_{n_1,n_2}(\mathbb{R})$, et soit $i_1, ..., i_{n_1}$ une attribution. On note $P \in \mathcal{M}_{n_1,n_2}(\mathbb{R})$ la matrice définie par $P_{p,q} = \delta_{q=i_p}$, c'est à dire que $P_{p,q}$ vaut 1 si on attribue p à q et 0 sinon.
Alors :

$$\sum_{k=1}^{n_1} \left| f_k - \hat{f}_{i_k} \right|^\alpha = \sum_{i,j} P_{i,j} C_{i,j}$$

On est donc ramené au problème d'optimisation suivant:

$$\min_{\substack{P \in \mathcal{M}_{n_1,n_2}(\mathbb{Z}/2\mathbb{Z}) \\ \sum_{j=1}^{n_2} P_{i,j} = 1 \ \forall i \\ \sum_{i=1}^{n_1} P_{i,j} \leq 1 \ \forall j}} \sum_{i,j} P_{i,j} C_{i,j}.$$

qui peut se réécrire à l'aide du problème de Monge :

$$\min_{\substack{\tilde{C} \in \mathbb{R}^{n_1 \times n_1} \\ \text{extraite de } C}} \min_{\sigma \in Perm(n)} \sum_{i=1}^{n} \tilde{C}_{i,\sigma(i)}. \tag{7}$$

.

Il existe des librairies python qui résolvent ce type de problème, nous utiliserons donc le package *mip* [1] pour le résoudre.

Les résultats obtenus sont présentés en Figure (27) et on constate une efficacité plutôt faible puisqu'on prédit la bonne attribution seulement dans 30% des cas en moyenne. Cependant on peut noter que l'attribution du nucléotide est bonne une fois sur deux (2 fois mieux que le hasard) et on obtient une efficacité intéressante sur l'attribution de l'atome.
Ces résultats peuvent être encourageants dans l'optique où nous disposerions de spectres 2D afin d'imposer des contraintes et d'améliorer ces résultats.

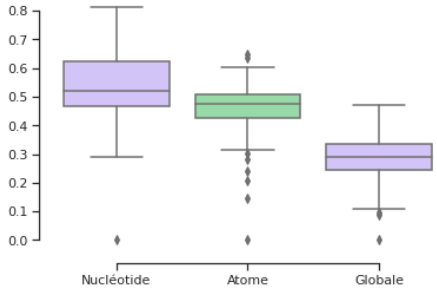


Figure 27: Résultats de la première méthode d'attribution.

4.4 Probabilités Bayesiennes

4.4.1 Attribution par maximisation de la vraisemblance

Une variante de la méthode utilisée précédemment est de résoudre le problème d'optimisation en remplaçant la matrice des distances par la matrice des log-vraisemblances [9]. Dans ce cas, on n'a pas besoin de générer aléatoirement un spectre et l'algorithme est déterministe.

Comme précédemment, on peut voir un spectre RMN comme des réalisations des variables aléatoires $(X_1, ..., X_n)$ qui suivent une loi de mélanges gaussiens. En effet les labels des pics que nous noterons $Y_1, ..., Y_n$ peuvent être vus comme des variables latentes et le modèle est décrit par :

$$X|Y \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$$

Les contraintes portent justement sur ces labels : en théorie, pour chaque nucléotide $J \in \{A, C, G, T\}$, pour chaque proton H dans ce nucléotide, on devrait avoir n_J pics du label (J, H) (où n_J est le nombre de nucléotides J dans la séquence). Ces contraintes s'écrivent :

$$\sum_{i=1}^{n} \mathbb{1}_{Y_i=(J,H)} = n_J \quad , \quad \text{pour tout atome } H \text{ de tout nucléotide } J.$$

Nous avons déjà discuté du fait que les spectres de la base de données ne sont pas complets, et qu'en pratique $n < \sum_J \sum_H n_J$. Les contraintes se réécrivent donc :

$$\sum_{i=1}^{n} \mathbb{1}_{Y_i=(J,H)} \leq n_J \quad , \quad \text{pour tout atome } H \text{ de tout nucléotide } J. \tag{8}$$

Comme nous voulons ici résoudre une tâche d'attribution de labels, nous sommes intéressés par les quantités $p(y|x)$.

On veut résoudre un problème d'optimisation de log-vraisemblance en imposant les contraintes (8). Afin de ne pas rajouter de redondance dans les contraintes on considère le mélange de distributions $\sum_{z \text{ label}} \pi_z \mathcal{N}(\mu_z, \sigma_z^2)$ avec $\pi_z = \frac{1}{N}$ où N est le nombre de labels. On obtient alors :

$$p(y|x) = \frac{\mathcal{N}(x; \mu_y, \sigma_y^2)}{\sum_{z \text{ label}} \mathcal{N}(x; \mu_z, \sigma_z^2)} \tag{9}$$

où $\mathcal{N}(x; \mu, \sigma^2)$ désigne la densité de probabilité de la loi $\mathcal{N}(\mu, \sigma^2)$ évaluée en x.

On veut donc maximiser la somme $\sum_{i=1}^{n} log\left(\mathcal{N}(x_i; \mu_{y_i}, \sigma^2_{y_i})\right)$.
En notant $(l_1, ..., l_N)$ les labels possibles, et en posant $P \in \mathbb{R}^{n \times N}$ la matrice définie par $P_{i,j} = 1$ si on attribue le label l_j au pic x_i et 0 sinon, on cherche donc à résoudre le problème d'optimisation suivant :

$$\max_{\substack{P \in \mathcal{M}_{n,N}(\mathbb{Z}/2\mathbb{Z}) \\ \sum_{j=1}^{N} P_{i,j} = 1 \ \forall i \\ \sum_{i=1}^{n} P_{i,j} \leq n_J \ \forall j \text{ tq } l_j = (J,h)}} \sum_{i,j} P_{i,j} log\left(\mathcal{N}(x_i; \mu_{l_j}, \sigma^2_{l_j})\right). \tag{10}$$

On obtient de meilleurs résultats que pour la méthode précédente comme l'illustre la Figure (28), avec presque 40% de bonnes attributions en moyenne et des scores pouvant aller jusqu'à 70%.

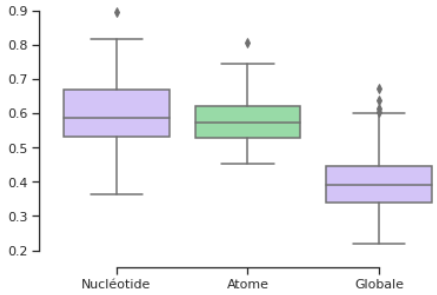


Figure 28: Résultats de la deuxième méthode d'attribution.

4.4.2 Listes d'attributions de pics individuels.

Nous avons pu constaté que les deux méthodes précédentes conduisent à des résultats encourageant pour des premiers essais avec si peu de données, mais que les scores obtenus ne permettent pas en général de vraiment guider le biologiste pour l'attribution. Il peut donc être intéressant de fournir en plus d'une attribution complète des listes d'attributions possibles pour chaque pic, indépendamment des autres.
Pour cela, on regarde juste les probabilités de chaque label pour un pic ($p(y|x)$) et on sélectionne les plus hautes.
On obtient donc en sortie des listes de m propositions de labels pour les n pics.

Les résultats présentés Figure (29) sont très encourageants car on voit que 3 choix est un bon compromis : le bon label fait partie des 3 propositions dans 70% des cas en moyenne.
En supposant qu'on dispose d'autres informations (autres spectres RMN 1D ou spectres RMN 2D ou 3D), on pourrait recouper ces informations pour améliorer l'attribution complète et ainsi presque automatiser totalement la tâche d'attribution.

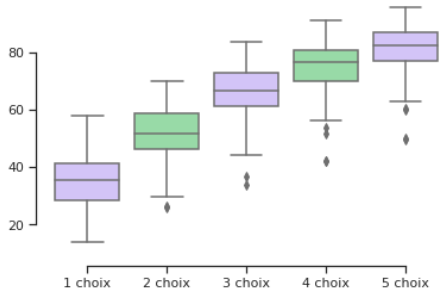


Figure 29: Résultats avec plusieurs choix possibles d'attributions.

4.4.3 Observation des erreurs

Il est intéressant d'observer les pics bien ou mal attribués pour connaître les zones qui posent problèmes et sur lesquelles on doit se concentrer pour améliorer l'algorithme.

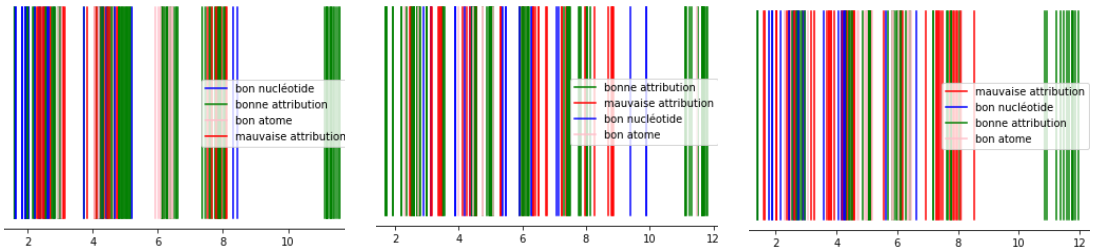


Figure 30: Attributions pour 3 exemples (parallèle , antiparallèle , hybride).

On observe sur la Figure (30) plusieurs zones qui semblent poser problème dans les 3 exemples, notamment autour de 2 ppm, 4 ppm, 7 ppm et 8 ppm. Ces résultats ne sont pas étonnants car ces zones correspondent à des chevauchements des distributions de plusieurs atomes comme on peut voir en Figure (26).
L'utilisation de spectres 2D ou des résonances des atomes de carbone pourraient aider à résoudre ce problème car cela permettrait de réduire l'overlapping des fréquences entre certains atomes.

5 Conclusion

Durant ce projet plusieurs thématiques ont été abordées : la compréhension d'un sujet biologique et la familiarisation avec le vocabulaire et les données, le pré-traitement des données, la recherche bibliographique, la recherche d'algorithmes spécifiques pour ce projet et leur implémentation.
Je pense que les deux tâches initiales du projet ont été en partie complétées : pour chaque tâche plusieurs méthodes ont été testées, comparées et discutées. Les résultats obtenus sont encourageant dans le sens où des scores relativement bons ont été obtenus avec une connaissance assez faible du sujet biologique, laissant donc l'opportunité d'améliorer les algorithmes proposés en utilisant une meilleure expertise des données.

J'ai trouvé ce projet très intéressant du fait de la découverte d'un domaine nouveau. Je remercie mon maître de stage Brahim Heddi pour ses explications claires du sujet et du contexte, ainsi que pour nos discussions qui m'ont permit de comprendre les biais, les limites et les potentielles améliorations des algorithmes que je proposais. En effet la recherche d'interprétabilité dans les algorithmes du point de vue biologique était enrichissante puisqu'elle guidait la recherche de nouveaux algorithmes plus spécifiques à nos données.

Ce projet a renforcé mon idée de travailler dans l'IA appliquée à la biologie/santé plutôt que dans la recherche purement mathématique. En effet j'ai trouvé très stimulant le fait de travailler sur des données issues d'un domaine dans lequel le machine learning est apparu depuis peu, et donc de pouvoir réfléchir et créer des modèles mathématiques spécifiques à ce type de données.

J'ai également apprécié l'organisation de ce semestre partitionné entre ce projet et le suivi de deux cours du MVA qui m'ont été très utiles. En effet j'ai suivi les cours de Computational Optimal Transport et Probabilistic Graphical Models qui m'ont donné des idées d'algorithmes pour ce projet et ont donc permis des avancées progressives durant ces 4 mois.

Je remercie mes encadrants pour le temps qu'ils m'ont accordé ainsi que les responsables du parcours IA pour avoir rendu possible ce projet.
J'encourage vivement les futurs élèves du parcours IA à effectuer leur stage au sein du LBPA, les possibilités en machine learning étant immenses du fait de l'existence d'un grand nombre de données encore traitées entièrement à la main aujourd'hui. Lors de mes recherches bibliographiques j'ai pu constater qu'un grand nombre d'articles appliquant de l'IA aux données biologiques sont très récents (entre 2018 et 2021), signe de son potentiel et des nouvelles avancées possibles en biologie.

References

[1] *Mip package*, https://www.python-mip.com/.

[2] *Protein data bank*, https://www.rcsb.org/.

[3] Michael Adrian, Brahim Heddi, and Anh Tuân Phan, *Nmr spectroscopy of g-quadruplexes*, Methods **57** (2012), 11–24.

[4] Todd AK, Johnston M, and Neidle S., *Highly prevalent putative quadruplex sequence motifs in human dna*, Nucleic Acids Research **33** (2005), 2901–2907.

[5] Leo Breiman, *Random forests*, Machine Learning (2001).

[6] D Chen, Z Wang, D Guo, V Orekhov, and X QU, *Review and prospect: Deep learning in nuclear magnetic resonance spectroscopy*, (2020).

[7] Dan Cireşan, Ueli Meier, and Juergen Schmidhuber, *Multi-column deep neural networks for image classification*, 2012.

[8] G Cybenko, *Approximation by superpositions of a sigmoidal function*, Mathematics of Control, Signals and Systems **2(4)** (1989), 303–314.

[9] Jose F.S, Bravo-Ferreira, Cowburn D., Khoo Y., and Singer A., *Nmr assignment through linear programming*, arXiv:2008.03641 (2020).

[10] Alexander Howarth, Kristaps Ermanis, and Jonathan M. Goodman, *Dp4-ai automated nmr data analysis: straight from spectrometer to structure*, Chem. Sci. **11** (2020), 4351–4359.

[11] G Montufar, R Pascanu, K Cho, and Y Bengio, *On the number of linear regions of deep neural networks*, NIPS **27** (2014).

[12] Rajesh Kumar Reddy Sannapureddi, Manish Kumar Mohanty, Anoop Kumar Gautam, and Bharathwaj Sathyamoorthy, *Characterization of dna g-quadruplex topologies with nmr chemical shifts*, The Journal of Physical Chemistry Letters **11** (2020), no. 23, 10016–10022, PMID: 33179931.

[13] J. D. WATSON and F. H. C. CRICK, *Molecular structure of nucleic acids: A structure for deoxyribose nucleic acid*, Nature **171** (1953), 737–738.

[14] Jianguo Xia, Trent C. Bjorndahl, Peter Tang, and David S. Wishart, *Metabominer – semi-automated identification of metabolites from 2d nmr spectra of complex biofluids*, BMC Bioinformatics (2008).