Robin Fuchs

ENSAE 3ème année
Stage de fin d'études
Année scolaire 2018-2019

DGMM et classification de communautés phytoplanctoniques

CREST
Palaiseau

Maitre de stage : Nicolas Chopin
Du 20 mai au 25 août 2019

Table des matières

1 Introduction

Le début des années 2010 marque la prise de conscience par le grand public des possibilités offertes par les réseaux de neurones artificiels profonds au travers des systèmes de recommandation, la victoire de AlphaGo contre Lee Sedol ou encore la reconnaissance des visages par le réseau DeepFace de Facebook.
Depuis les travaux précurseurs de Rosenblatt (1957) [19], plusieurs décennies se sont pourtant écoulées avant que les performances des réseaux de neurones ne deviennent réellement impressionnantes et dépassent celles des algorithmes plus traditionnels. Au-delà des avancées théoriques survenues au cours des années 1980-1990 grâce notamment aux travaux de LeCun [15] ou plus récemment, c'est aussi l'augmentation des capacités de calcul qui a finalement permis la suprématie des réseaux de neurones et de l'apprentissage profond dans nombre de contextes.

Les réseaux de neurones artificiels sont des modèles structurés en une succession de couches constituées d'un certain nombre de neurones. Un nombre de couches plus élevé permet au réseau de capter des structures plus complexes au sein des données qui pourront être réutilisées dans un second temps pour faire des prédictions. Ce processus d'apprentissage-prédiction à l'aide d'un réseau de neurones comportant un grand nombre de couches (typiquement au moins supérieur à une dizaine) est appelé apprentissage profond. Le processus d'apprentissage profond nécessite cependant une puissance de calcul qui était prohibitive jusqu'à peu.

Les développements récents en Apprentissage Profond se sont davantage concentrés sur des tâches d'apprentissage supervisé, c'est-à-dire des problèmes pour lesquels l'algorithme "apprend" à partir d'exemples annotés afin de réaliser ensuite des prédictions. La très forte émulation autour du cadre supervisé par rapport au cadre non-supervisé, dans lequel on ne donne pas d'informations auxiliaires sur le phénomène à prédire à l'algorithme, s'explique par plusieurs raisons. Les principales semblent être que les nouveaux domaines d'application ouverts par l'apprentissage supervisé, comme la reconnaissance d'images ou de textes, semblent très prometteurs et que les tâches d'apprentissage non-supervisé (profond ou non) sont souvent considérées comme plus difficiles à étudier.

Certains développements ont cependant eu lieu dans le cadre de l'apprentissage profond non-supervisé. C'est le cas notamment des *Deep Gaussian Mixture Model* (DGMMs) utilisés pour faire du *clustering*, c'est-à-dire identifier un nombre *a priori* inconnu de groupes d'observations homogènes au sein des données.
Pour ce faire, les DGMMs généralisent le principe de mélanges gaussiens très fréquemment utilisés en *clustering*. Ces derniers ont été introduits par Mclachlan & Peel (2000) [18] et Fraley & Raftery (2002)[11] et supposent que chaque groupe présent dans les données a été

généré par une distribution gaussienne distincte. Les DGMMs sont une généralisation de ce principe au sens où chaque couche du réseau prise isolément peut-être considérée comme une généralisation de mélange gaussien.
Mclachlan & Viroli (2017) [23] dressent dans un article récent un bilan de l'état de l'art en ce qui concerne les DGMMs sur lequel cette thèse prendra appui. Si dans beaucoup d'applications d'aprentissage non-supervisé, les DGMMs obtiennent une performance supérieure aux techniques traditionnelles, il n'en reste pas moins que les DGMMs souffrent de certains problèmes d'identifiabilité, ce qui contraint fortement les spécifications envisageables pour ces modèles. La procédure de sélection de modèles par validation croisée rend également difficile d'identifier les architectures les plus performantes. Ainsi, introduire une version bayésienne des DGMMs basée sur les méthodes variationnelles permettrait d'adresser les deux problèmes simultanément et constitue le premier axe de développement de ce stage puis de cette thèse. En effet, cette période de stage d'environ trois mois au CREST a été l'occasion de démarrer les travaux de ma thèse.

Au-delà des développements en apprentissage entrepris, cette dernière s'attelera à appliquer une approche statistique à des problématiques océanographiques. Ces problèmes, intrinséquemment dynamiques, spatialisés et en grande dimension se prêtent en effet particulièrement bien à l'analyse statistique, l'éclairage apporté étant relativement complémentaire avec les modèles physiques, chimiques et biologiques en vigueur.

De manière plus précise le phénomène océanographique étudié au cours de cette thèse sera la structure des populations de phytoplanctons, c'est-à-dire la distribution jointe observée en un point donné de groupes de phytoplanctons homogènes. Le phytoplancton désigne une grande variété d'organismes autotrophes (qui synthétisent leur matière organique principalement à partir d'éléments minéraux) dont la taille varie de quelques dizaines à quelques centaines de micromètres. La contribution du phytoplancton à la production primaire, c'est-à-dire à l'assimilation du CO_2 dissout par unité de temps, est équivalente à l'ensemble de la production primaire terrestre. De ce fait, la compréhension des réactions du phytoplancton à son environnement est de première importance afin de comprendre la façon dont l'océan continuera à stocker le CO_2 au cours du processus de réchauffement climatique. La mer Méditerranée, du fait de ses caractéristiques particulières comme la faible profondeur, des conditions météorologiques très changeantes, la forte pression anthropique et la forte activité économique dans laquelle elle est impliquée, est un terrain d'étude privilégié. Cette thèse s'appliquera ainsi à éclaircir l'impact des variables environnementales sur les communautés de phytoplanctons, que cela soit à l'aide de méthodes statistiques existantes comme les champs aléatoires gaussiens, la classification supervisée de courbes ou en adaptant les développements théoriques réalisés autour des DGMMs. Avant de pouvoir mener cette étude d'impact, il est néanmoins nécessaire de pouvoir automatiser la reconnaissance de groupes

fonctionnels du phytoplancton (espèces se comportant de façon similaire), ce que ce stage m'a permis de commencer à faire.

L'ensemble des éléments précédemment évoqués nous amène à définir ce qu'a été le sujet de ce stage : "Développements bayésiens et implémentations des DGMMs - applications aux groupes fonctionnels du phytoplancton".

2 Développements autour des DGMMs

2.1 Présentation du modèle

Un Deep Gaussian Mixtures Model (DGMM) est un modèle multicouche dans lequel chaque couche est constitué d'un mélange de modèles de facteurs (MFA) gaussiens. Le nombre d'unités (aussi appelées composantes ou neurones) sur chaque couche, K_l, est choisi par l'utilisateur comme pour le nombre de composantes d'un mélange Gaussien. Soit y la matrice des données de taille $N \times p$ avec N le nombre d'observations et p la dimension. On définit $z^{(l)}$ la variable latente de la couche $l \in [1, L]$ de dimension r_l et on pose $y = z^{(0)}$. De cette façon, on peut écrire le modèle sous la forme suivante :

$$\begin{cases} y_n = \eta_{k_1}^{(1)} + \Lambda_{k_1}^{(1)} z_n^{(1)} + u_n^{(1)} \text{ avec proba } \pi_{n,(k_o,k_1)}^{(1)} \text{ pour } k_0 \in \{1\} \text{ et } k_1 \in [1, K_1] \\ z_n^{(1)} = \eta_{k_2}^{(2)} + \Lambda_{k_2}^{(2)} z_n^{(2)} + u_n^{(2)} \text{ avec proba } \pi_{n,(k_1,k_2)}^{(2)} \text{ pour } k_1 \in [1, K_1] \text{ et } k_2 \in [1, K_2] \\ \dots \\ z_n^{(L-1)} = \eta_{k_L}^{(L)} + \Lambda_{k_L}^{(L)} z_n^{(L)} + u_n^{(L)} \text{ avec proba } \pi_{n,(k_{l-1},k_L)}^{(L)} \text{ pour } k_{L-1} \in [1, K_{L-1}] \text{ et } k_L \in [1, K_L] \\ z_n^{(L)} \sim \mathcal{N}(0, I_{r_L}) \end{cases}$$

$(u^{(l)})_{l \in [1,L]}$ est un terme gaussien centré de matrice de variance-covariance $\Psi^{(l)}$.
En posant $r_0 = p$, $\Lambda_{k_l}^{(l)}$ est de dimension $r_{l-1} \times r_l$, $\eta_{k_l}^{(l)}$ et $u^{(l)}$ sont des vecteurs de taille r_l et $\Psi^{(l)}$ est de dimension $r_l \times r_l$. Pour limiter la taille des expressions on pose $\Theta^{(l)} = (\eta_{k_l}^{(l)}, \Lambda_{k_l}^{(l)}, \Psi^{(l)})_{k_l \in [1,K_l]}$ A chaque couche, on définit la variable latente $s^{(l)}$, représentée pour chaque observation par un vecteur binaire de taille K_l valant 1 si l'observation a été générée par la k_l composante de la couche. Ainsi, $s^{(l)}$ est de dimension $N \times k_l$. La probabilité que $s_n^{(l)}$ vaille 1 en k-ème position est donc bien $\pi_{n,(k_{l-1},k_l)}^{(l)}$. Afin de simplifier les notations, on omettra souvent l'indice k_{l-1} des expressions.

La figure 1 est adaptée de McLachlan et Viroli (2017) [23] et permet d'illustrer simplement l'architecture d'un DGMM à deux couches, dont la première couche possède $k_1 = 3$ neurones et dont la deuxième couche possède $k_2 = 2$ neurones.

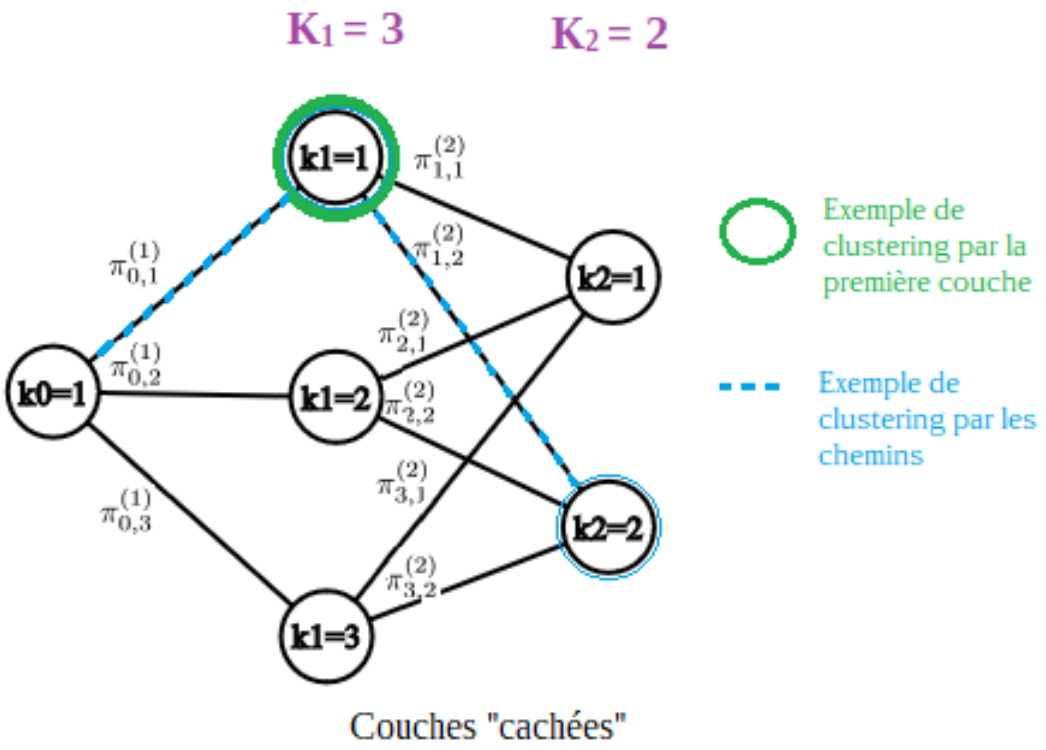

FIGURE 1 – Exemple de clustering d'une observation conduite grâce à la première couche ou aux chemins

L'idée de ce modèle est donc d'approximer les données, qui ne sont pas forcément gaussiennes, par plusieurs MFA successifs. Comme le montre la figure 1, le *clustering* peut être réalisé de deux façons. Tout d'abord en associant chaque observation à la composante présentant la plus forte probabilité $\pi^{(1)}_{n,(k_0,k_1)}$. De manière alternative, il est également possible de discrimer les groupes à l'aide des chemins possédant la plus forte probabilité de réalisation. Afin d'illustrer le propos, plaçons nous dans le cadre du réseau répresenté dans la figure 1. En clusterant à l'aide de la première couche, on place chaque observation dans l'une des $k_1 = 3$ composantes. Dans le cas du partitionnement à l'aide des chemins, chaque observation est associée au chemin $s = (k_1, k_2)$ parmi les $3^2 = 9$ chemins qui présentent la probabilité de réalisation $\pi^{(1)}_{n,(k_0,k_1)} \times \pi^{(2)}_{n,(k_1,k_2)}$ la plus forte. Le *clustering* par chemin permet donc pour le même prix computationnel de considérer davantage de groupes.

Peu importe l'option choisie, la distribution de l'observation n sachant que celle-ci a été générée par l'unité k_1 de la première couche ou par le chemin s suivent des distributions Gaussiennes. Plus précisemment, on a $f(y_n|z^{(1)}, s^{(1)}_n = k_1, \Theta^{(1)}) \sim \mathcal{N}(\eta^{(1)}_{k_1} + \Lambda^{(1)}_{k_1} z^{(1)}_n, \Psi^{(1)}_{k_1})$ et

$f(y_n|z^{(1)}, s_n = (k_1, k_2), \Theta^{(1)}, \Theta^{(2)}) \sim \mathcal{N}(\mu_s, \Sigma_s))$, avec

$$\mu_s = \eta_{k_1}^{(1)} + \sum_{l=2}^{L} \left(\prod_{m=1}^{l-1} \Lambda_{k_m}^{(m)} \right) \eta_{k_l}^{(l)} \text{ et } \Sigma_s = \Psi_{k_1}^{(1)} + \sum_{l=2}^{L} \left(\prod_{m=1}^{l-1} \Lambda_{k_m}^{(m)} \right) \Psi_{k_l}^{(l)} \left(\prod_{m=1}^{l-1} \Lambda_{k_m}^{(m)} \right)^T$$

2.1.1 Méthode d'entrainement

L'entrainement du modèle permet donc d'identifier les chemins les plus prometteurs (valeurs des $\pi_{k_{l-1},k_l}^{(l)}$ les plus élevées) en même temps qu'il estime les paramètres du modèle de facteur de la première couche $(\eta_{k_1}^{(1)}, \Lambda_{k_1}^{(1)}, \Psi_{k_1}^{(1)})$, ce qui permet de réaliser le clustering de la manière que l'on souhaite.

Du fait de la structure en couches du DGMM, on peut décomposer la vraisemblance complétée du modèle de façon Markovienne (les indices sont ici juste inversés par rapport à la convention) :
$L(y, z, s, \Theta) = L(y|z^{(1)}, s, \Theta)L(z^{(1)}|z^{(2)}, s, \Theta)...L(z^{(h-1)}|z^{(h)}, s, \Theta)$
avec $\Theta = (\Lambda_{k_l}^{(l)}, \Psi_{k_l}^{(l)}, \Sigma_{k_l}^{(l)})_{k_l \in [1,K_l], l \in [1,L]}$.
On peut donc maximiser la vraisemblance conditionnelle sur chaque couche afin de maximiser la vraisemblance totale du modèle.
Plus précisement, l'entrainement consiste à maximiser sur chaque couche :

$$E_{z^{(l)}, s^{(l+1)}|z^{(l-1)}, \Theta} \left[\sum_{i=1}^{n} \log f(z_i^{(l-1)}|z_i^{(l)}, s, \Theta) \right] = \sum_{i=1}^{n} \int f(z_i^{(l)}, s|z_i^{(l-1)}, \Theta') \log f(z_i^{(l-1)}|z_i^{(l)}, s, \Theta) dz_i$$

Les intégrales présentes dans la fonction objectif la rende difficile à maximiser. Les auteurs proposent donc de recourir à une version stochastique de l'algorithme Expectation - Maximisation (SEM) [7].

Ce dernier se déroule de la façon suivante :

Algorithm 1: Entrainement du DGMM par SEM

Paramètres : M le nombre d'observations à générer à l'étape S;

Initialisation : $z^{(l)}, \Lambda^{(l)}, \eta^{(l)}$ par analyse factorielle ou ppca sur chaque couche. $s^{(l)}$ par k-means sur chaque couche.

for *t=1,2,...* **do**

Etape S

Générer M observations à partir de $f(z_i^{(l)}|z_i^{(l-1)}, s, \Theta)$ (on connait $z_i^{(l-1)}$ par construction).

Etape E : On approxime ensuite $\mathbb{E}[z_i^{(l)}|z_i^{(l-1)}, s, \Theta]$ par $\frac{\sum_{m=1}^{M} z_{i,m}^{(l)}}{M}$

et $\mathbb{E}[z_i^{(l)} z_i^{(l)T}|z_i^{(l-1)}, s, \Theta]$ par $\frac{\sum_{m=1}^{M} z_i^{(l)} z_i^{(l)T}}{M}$

Etape M :

$$\hat{\Lambda}_{s_l}^{(l)} = \frac{\sum_{i=1}^{n} p(s|z_i^{(l-1)})(z_i^{(l-1)} - \eta_{s_l}^{(l)})\mathbb{E}[z_i^{(l)T}|z_i^{(l-1)}, s]\mathbb{E}[z_i^{(l)} z_i^{(l)T}|z_i^{(l-1)}, s]^{-1}}{\sum_{i=1}^{n} p(s|z_i^{(l-1)})}$$

$$\hat{\Psi}_{k_l}^{(l)} = \frac{\sum_{i=1}^{n} p(s|z_i^{(l-1)})(z_i^{(l-1)} - \eta_{k_l}) \left[(z_i^{(l-1)} - \eta_{k_l})^T - \mathbb{E}[z_i^{(l)T}|z_i^{(l-1)}, s]\hat{\Lambda}_{k_l}^T\right]}{\sum_{i=1}^{n} p(s|z_i^{(l-1)})}$$

$$\hat{\eta}_{s_l}^{(l)} = \frac{\sum_{i=1}^{n} p(s|z_i^{(l-1)}) \left[z_i^{(l-1)} - \Lambda_{k_l}^{(l)}\mathbb{E}[z_i^{(l)T}|z_i^{(l-1)}, s]\right]}{\sum_{i=1}^{n} p(s|z_i^{(l-1)})}$$

$$\hat{\pi}_{k_l}^{(l)} = \sum_{n=1}^{N} f(s^{(l)}|y)$$

end

Retourne : $\hat{\Lambda}_{k_l}^{(l)}, \hat{\eta}_{k_l}^{(l)}, \hat{\Psi}_{k_l}^{(l)}, \hat{\pi}_{k_l}^{(l)}$ $\forall k_l \in [1, K_l]$ et $\forall l \in L$

2.1.2 Problèmes d'identifiabilité

Cependant, sans restrictions supplémentaires sur la paramétrisation du modèle, il apparait que le modèle n'est pas identifiable sans ajouter d'hypothèses supplémentaires. Tout d'abord, dans le cas du *clustering* à l'aide des chemins, plusieurs architectures conduisent à un nombre total de chemins possibles à travers le réseau identique. McLachlan & Viroli (2017) [23] donnent l'exemple de deux réseaux à deux couches dont la première aurait 3 neurones sur la première couche et 2 sur la deuxième. Le deuxième réseau aurait quant à lui 2 neurones sur la première et 3 sur la deuxième couche. Dans les deux cas, on aboutit à un nombre total de chemins égal à $2 \times 3 = 6$ (il y a trois chemins possibles à partir de chacun des

deux neurones de la couche précédente pour le deuxième réseau et inversement), ce qui rend le modèle non identifiable. Afin de résoudre ce problème les auteurs proposent que la dimensionalité des variables latentes décroisse strictement au fil des couches : $p > r_1 > ... > r_L$. Intuitivement cela signifie que l'information doit être représentée dans des espaces de plus en plus faible dimension. Ainsi, les paramètres d'une couche ne peuvent comporter une information parfaitement similaire à celle contenue dans les paramètres d'une autre couche et le modèle redevient identifiable.

Le deuxième problème d'identifiabilité des DGMMs se situe cette fois non pas au niveau de l'architecture du réseau mais au niveau de la couche individuelle. On se place sur la première couche après les données et on suppose qu'elle ne comporte qu'une unité ($k_1 = 1$). On rappelle que y désigne les données observées de dimension (n, p), η un vecteur de taille r_1, Λ une matrice de chargement de facteur de dimension (p, r_1), que z est une gaussienne multivariée centrée de dimension (n, r_1), et que u est le terme d'erreur gaussien multivarié de covariance Ψ. On peut alors écrire y_n comme

$$y_n = \eta + \Lambda z_n + u$$

Cependant, la matrice Λ n'est pas identifiable dans ce modèle car pour toute matrice A de dimension (r_1, r_1), ce modèle ne peut être distingué de

$$y_n = \eta + \Lambda A A^{-1} z_n + u_n$$

McLachlan & Viroli proposent d'imposer une condition sur les matrices régissant la variance du modèle en imposant que $\Lambda^T \Psi \Lambda$ soit diagonale et que ses coefficients soient classés par ordre décroissant comme dans [17]. Cette spécification s'applique à la première couche présentée ici aussi bien qu'aux couches suivantes et permet de revenir à un modèle strictement identifié. Cependant, on peut s'interroger à propos de l'influence de cette spécification sur la faculté du modèle à capturer certains phénomènes.

2.1.3 Le DGMM comme généralisation de modèles pré-existants

Ces problèmes d'identifiabilité dérivent en réalité directement des modèles que le DGMM généralisent tels que les mélanges gaussiens, les modèles d'analyse factorielle, les mélanges de modèles de facteurs (MFA) et les modèles factorielles de mélange (FMA). En effet, en prenant un DGMM avec $l = 1$ et $\Psi_{s_1}^{(1)}$ diagonales $\forall k_1 \in [1, K_1]$, on obtient un MFA

$$\begin{cases} y_n = \eta_{k_1}^{(1)} + \Lambda_{k_1}^{(1)} z_n^{(1)} + u_n^{(1)} \text{ avec proba } \pi_{n,(k_0,k_1)}^{(1)} \\ z_n^{(1)} \sim \mathcal{N}(0, I_{r_1}) \end{cases}$$

De même en fixant $l = 2$, $K_1 = 1$, $\Psi_{k_1}^{(1)}$ diagonale et $\Lambda_{k_2}^2 = \{0\}$ $\forall k_2 \in [1, K_2]$ on obtient un FMA de la forme suivante :

$$\begin{cases} z_n^{(0)} = \eta^{(1)} + \Lambda^{(1)} z_n^{(1)} + u_n^{(1)} \\ z_n^{(1)} = \eta_{k_2}^{(2)} + u_n^{(2)} \text{ avec proba } \pi_{n,(k_1,k_2)}^{(2)} \end{cases}$$

2.2 Développements entrepris

La version du modèle présentée par McLachlan & Viroli constitue donc une généralisation élégante de plusieurs autres modèles. On peut cependant lui apporter des améliorations d'un point de vue analytique et computationnel.

2.2.1 Développements analytiques

A l'heure actuelle, la sélection de la meilleure spécification du modèle se fait par validation croisée à l'aide de critères d'information du type AIC ou BIC. Ces critères pénalisent le pouvoir explicatif du modèle par sa complexité, que l'on peut mesurer dans le cas présent par le nombre d'unité et la dimensionalité des variables des différentes couches. Sélectionner le "meilleur" modèle nécessite donc de faire tourner plusieurs fois chaque version du modèle afin de déterminer empiriquement la valeur optimale de ses hyper-paramètres (nombre de couches et de neurones par couche, méthode d'initialisation des paramètres etc...). Cette procédure peut se montrer très coûteuse en termes computationnels.

Relancer plusieurs fois l'algorithme est d'autant plus nécessaire qu'il n'existe pas de résultats *a priori* sur la stabilité de l'algorithme. Comme l'évoque Bishop (2006) [4], il existe une singularité dans l'expression de la vraisemblance des mélanges Gaussiens qui apparait lorsqu'un des clusters ne tend à être constitué que d'une observation. Ce problème prend ici d'autant plus d'importance que le modèle est constitué de mélanges gaussiens imbriqués. Afin de pallier cela, les auteurs réaffectent alors aléatoirement certains points au groupe constitué d'une seule observation.

Devant ces limitations, introduire une version bayésienne du modèle présente plusieurs avantages. Dans le cas des mélanges gaussiens introduire une loi *a priori* sur les paramètres résoud le problème de la singularité en agissant comme un régulariseur [4]. De plus, cette version bayésienne permettrait d'identifier les chemins à travers le réseau ayant une probabilité postérieure proche de 0. Ces chemins pourraient donc être supprimés "à la volée" et

ainsi permettre une sélection de modèle sans coût computationnel supplémentaire. Enfin, cela permettrait d'expérimenter d'autres méthodes d'entrainement pour le modèle comme les méthodes variationnelles ou les méthodes d'Expectation-Propagation. Nous nous sommes concentrés sur les méthodes variationnelles au cours du stage. Nous viendrons certainement à l'entrainement grâce à l'algorithme EP dans un second temps.

Dans cette sous-partie, on rappellera donc tout d'abord le principe général des méthodes variationnelles. Puis seront définies les distributions intervenant dans la vraisemblance et les lois *a priori* choisies. Enfin, nous dériverons les lois *a posteriori* des différents paramètres intervenant dans l'entrainement du modèle. L'ensemble de ces étapes est adapté des chapitres 9 et 10 de l'ouvrage "Machine Learning and Pattern Recognition" (2006) de Bishop [4].

Principe des méthodes variationnelles

Le cadre variationnel peut s'appliquer à notre modèle, où l'entrainement se fait couche par couche. En effet, toutes les densités conditionnelles sont connues et ont pour forme :

$$\begin{cases} L(z^{(l)}|z^{(l+1)},\Theta) = \sum\limits_{k_{l+1}}^{K_{l+1}} \pi_{k_{l+1}}^{(l+1)} \mathcal{N}(z^{(l)}|\eta_{k_{l+1}}^{(l+1)} + \Lambda_{k_{l+1}}^{(l+1)} z^{(l+1)}, \Psi_{k_{l+1}}^{(l+1)}) \ \forall l \in \{0,...,L-1\} \\ L(z^{(L)}) \sim \mathrm{N}(0, \mathrm{I}_{\mathrm{r_L}}) \end{cases}$$

avec $\Theta^{(l)} = \{\Lambda_{k_l}^{(l)}, \eta_{k_l}^{(l)}, \Psi_{k_l}^{(l)}\}_{(\forall l \in [1,..,L])}$.

En dénotant $q(.)$ les distributions variationnelles des différents paramètres, on peut décomposer la densité des couches 0 à $L-1$ de la manière suivante :

$$L(z^{(l)}|z^{(l+1)},\Theta^{(l+1)}) = \mathrm{L}(q) + KL(q||p)$$

$$\text{où} \begin{cases} \mathrm{L}(q) = \int q(s^{(l+1)}) \ln \frac{p(z^{(l)},s^{(l+1)}|z^{(l+1)},\Theta^{(l+1)})}{q(s^{(l+1)})} ds^{(l+1)} \\ KL(q||p) = -\int q(s^{(l+1)}) \ln \frac{p(s^{(l+1)}|z^{(l)},z^{(l+1)},\Theta^{(l+1)})}{q(s^{(l+1)})} ds^{(l+1)} \end{cases}$$

$\mathrm{L}(q)$ est une borne inférieure de la vraisemblance, souvent appelée ELBO pour *Evidence Lower Bound*, et $KL(q||p)$ est la divergeance de Kullback-Liebler entre la distribution variationnelle proposée et la "vraie" distribution conditionnelle des données. Remarquons que l'approche Expectation-Propagation se base elle sur la divergeance de Kullback-Liebler inversée $K(p||q)$.
Chercher à maximiser la vraisemblance, revient donc à maximiser $\mathrm{L}(q)$ et à minimiser $KL(q||p) \geq 0$ [5]. Sans restriction sur les distributions variationnelles, on maximise la vraisemblance en choisissant $q(s^{(l+1)}) = p(s^{(l+1)}|z^{(l)}, z^{(l+1)}, \Theta^{(l+1)})$ et on retombe sur l'algorithme EM.

L'idée des méthodes variationnelles est donc de se restreindre à des distributions plus simple (soit parce que la vraie distribution conditionnelle n'est pas calculable soit pour simplifier le modèle). On se restreint en général à des distributions jointes des paramètres qui sont factorisables par bloc. Il y a donc indépendance entre plusieurs blocs de variables. Dans notre cas, on suppose que $s^{(l+1)}$ est indépendant de $\pi^{(l+1)}, \Theta^{(l+1)}$. On peut ensuite montrer [4] que les distributions variationnelles factorisables par bloc qui maximisent la vraisemblance sont :

$$\begin{cases} \ln q^*(s^{(l+1)}) = \mathrm{E}_{\pi^{(l+1)},\Theta^{(l+1)}}[\ln p(z^{(l)}, \pi^{(l+1)}, s^{(l+1)}, \Theta^{(l+1)}|z^{(l+1)})] + cst \\ \ln q^*(\pi^{(l+1)}, \Theta^{(l+1)}) = \mathrm{E}_{s^{(l+1)}}[\ln p(z^{(l)}, \pi^{(l+1)}, s^{(l+1)}, \Theta^{(l+1)}|z^{(l+1)})] + cst' \end{cases} \tag{1}$$

avec cst et cst' deux constantes.

Rappel des termes relatifs à la vraisemblance de chaque couche

Pour chaque observation $n \in N$ de la couche $l \in [0, L-1]$, on rappelle que la variable latente $s_n^{(l+1)}$ est un vecteur binaire de taille K_l valant 1 en k-ème position si l'observation n appartient au k-ème groupe, 0 sinon. π_{k_l} étant la probabilité que l'observation n appartienne au groupe k_l

On peut donc écrire les distributions totales et conditionnelles des variables latentes :

$$p(s^{(l)}) = \prod_{k_l}^{K_l} \pi_{k_l}^{s_{k_l}}$$

$$p(s^{(l)}|\pi^{(l)}) = \prod_{n=1}^{N} \prod_{k_l=1}^{K_l} \pi_{k_l}^{s_{n,k_l}^{(l)}} \ \forall l \in [1, .., L]$$

.

Les lois conditionnelles des données "observées" à chaque couche ($z^{(l)}\ \forall l \in [0, .., L-1]$) sont quasiment les mêmes que dans un mélange gaussien classique à l'exception de l'expression de la moyenne :

$$p(z^{(l)}|s^{(l+1)}, z^{(l+1)}, \Theta^{(l+1)}) = \prod_{k_{l+1}=1}^{K_{l+1}} \mathcal{N}(\eta_{k_{l+1}}^{(l+1)} + \Lambda_{k_{l+1}}^{(l+1)} z^{(l+1)}, \Psi_{k_{l+1}}^{(l+1)})^{s_{k_{l+1}}^{(l+1)}}$$

$$p(z^{(l)}|z^{(l+1)}, s^{(l+1)}, \Theta^{(l+1)}) = \prod_{n=1}^{N} \prod_{k_{l+1}=1}^{K_{l+1}} \mathcal{N}(z^{(l)}|\eta_{k_{l+1}}^{(l+1)} + \Lambda_{k_{l+1}}^{(l+1)} z^{(l+1)}, \Psi_{k_{l+1}}^{(l+1)})^{s_{n,k_{l+1}}^{(l+1)}}$$

Enfin, d'après la répresentation graphique du modèle gouvernant chaque couche présentée

en figure 2, on peut décomposer la loi jointe de la façon suivante :

$$
\begin{aligned}
p(z^{(l)}, \pi^{(l+1)}, s^{(l+1)}, \Theta^{(l+1)} | z^{(l+1)}) & \\
&= p(z^{(l)} | \pi^{(l+1)}, s^{(l+1)}, \Theta^{(l+1)} | z^{(l+1)}) p(s^{(l+1)} | \pi^{(l+1)}, \Theta^{(l+1)} | z^{(l+1)}) \\
&\times p(\pi^{(l+1)} | \Theta^{(l+1)}, z^{(l+1)}) p(\Theta^{(l+1)} | z^{(l+1)}) \\
&= p(z^{(l)} | z^{(l+1)}, s^{(l+1)}, \Theta^{(l+1)} | \pi^{(l+1)}) p(\pi^{(l+1)} | z^{(l+1)}) p(\Theta^{(l+1)}) \\
&:= p(z^{(l)} | z^{(l+1)}, s^{(l+1)}, \Theta^{(l+1)} | \pi^{(l+1)}) p(\pi^{(l+1)} | z^{(l+1)}) p(\eta^{(l+1)}, \Lambda^{(l+1)}, \Psi_{k_{l+1}}^{(l+1)})
\end{aligned}
\tag{2}
$$

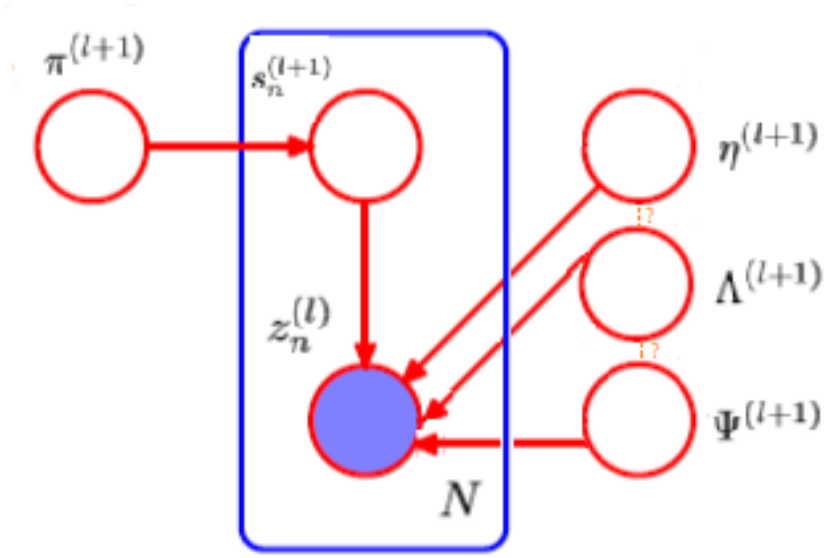

FIGURE 2 – Représentation graphique du modèle bayésien réécrit pour chaque couche

Choix des lois a priori

On définit tout d'abord le prior sur les coefficients de mélange, qui suit classiquement une loi de Dirichlet : $p(\pi^{(l)}) = Dir(\pi^{(l)}|\alpha_0) = C(\alpha_0) \prod_{k_l=1}^{K_l} \pi_{k_l}^{\alpha_0 - 1}$ avec $C(\alpha_0)$ la constante de normalisation de la loi de Dirichlet. Plus α_0 est élevé plus la loi *a posteriori* sera influencée par les données plutôt que par la loi *a priori* et inversement.

On aimerait maintenant définir un prior sur $\eta^{(l)}, \Lambda^{(l)}, \Psi^{(l)}$. L'indépendance entre ces paramètres nous semble peu adaptée dans le sens où ces derniers sont intimement liés au sein de la vraisemblance au travers des termes en $\mathcal{N}(\eta_{k_{l+1}}^{(l+1)} + \Lambda_{k_{l+1}}^{(l+1)} z^{(l+1)}, \Psi_{k_{l+1}}^{(l+1)})$ $\forall k_{l+1} \in [1, K_{l+1}]$. On aimerait de plus trouver un prior qui nous permette de satisfaire la deuxième contrainte d'identifiabilité. Autrement dit, on cherche des lois *a priori* conjuguées sur $\eta^{(l)}, \Lambda^{(l)}, \Psi^{(l)}$, telles que les réalisations de $\Lambda^T \Psi \Lambda$ soient des matrices diagonales dont les coefficients soient

classés par ordre décroissants.
Nous ne sommes pas arrivés à identifier de tels priors pour le moment et nous nous référerons donc à la loi jointe de $\Theta^{(l+1)}$ sans spécifier son expression. Même sans l'expression explicite de la loi de $\Theta^{(l+1)}$, il est possible de dériver les lois *a posteriori* de $s^{(l+1)}$ et de $\pi^{(l+1)}$.

Détermination des lois variationnelles a posteriori

On se restreint à une distribution variationnelle jointe qui se factorise de la façon suivante :

$$q(\pi^{(l+1)}, s^{(l+1)}, \Theta^{(l+1)}) = q(s^{(l+1)})q(\pi^{(l+1)}, \Theta^{(l+1)})$$

D'après l'équation (1), on sait que :

$$\ln q^*(s^{(l+1)}) = \mathrm{E}_{\pi^{(l+1)},\Theta^{(l+1)}}\big[\ln p(z^{(l)}, \pi^{(l+1)}, s^{(l+1)}, \Theta^{(l+1)}|z^{(l+1)})\big] + cst \tag{3}$$

avec *cst* une constante qui contient tous les termes ne dépendant pas de $s^{(l+1)}$.

On développe (3) en utilisant la décomposition (2) et en plaçant tous les termes qui ne dépendent pas de $s_{k_{l+1}}^{(l+1)}$ dans la constante :

$$\begin{aligned}
\ln q^*(s^{(l+1)}) &= \mathrm{E}_{\pi^{(l+1)},\Theta^{(l+1)}}\Big[\ln p(z^{(l)}|s^{(l+1)}, z^{(l+1)}, \Theta^{(l+1)})] \\
&\quad + \mathrm{E}_{\pi^{(l+1)},\Theta^{(l+1)}}[\ln p(s^{(l+1)}|\pi^{(l+1)})\Big] + cst \\
&= \mathrm{E}_{\Theta^{(l+1)}}[\ln p(z^{(l)}|s^{(l+1)}, z^{(l+1)}, \Theta^{(l+1)}] \\
&\quad + \mathrm{E}_{\pi^{(l+1)}}[\ln p(s^{(l+1)}|\pi^{(l+1)})] + cst
\end{aligned}$$

En utilisant les expressions des lois données plus haut, il vient que :

$$\begin{aligned}
\ln q^*(s^{(l+1)}) = -\frac{1}{2}\mathrm{E}_{\Theta^{(l+1)}}\Big[&\sum_{n=1}^{N}\sum_{k_{l+1}=1}^{K_{l+1}} s_{n,k_{l+1}}^{(l+1)}[\ln|\Psi_{k_{l+1}}^{(l+1)}| \\
&+ \big(z_n^{(l)} - (\eta_{k_{l+1}}^{(l+1)} + \Lambda_{k_{l+1}}^{(l+1)} z_n^{(l+1)}\big)^T \Psi_{k_{l+1}^{(l+1)}}^{-1}\big(z_n^{(l)} - (\eta_{k_{l+1}}^{(l+1)} + \Lambda_{k_{l+1}}^{(l+1)} z_n^{(l+1)})\big) + \frac{r_{l+1}}{2}\ln(2\pi)]\Big] \\
&+ \mathrm{E}_{\pi^{(l+1)}}\left[\sum_{n=1}^{N}\sum_{k_{l+1}=1}^{K_{l+1}} s_{n,k_{l+1}}^{(l+1)} \ln \pi_{k_{l+1}}^{(l+1)}\right] + cst
\end{aligned}$$

D'où :

$$\ln q^*(s^{(l+1)}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{k_{l+1}=1}^{K_{l+1}} s_{n,k_{l+1}}^{(l+1)} \mathrm{E}_{\Theta^{(l+1)}}\Big[\ln|\Psi_{k_{l+1}}^{(l+1)}| + (z_n^{(l)} - (\eta_{k_{l+1}}^{(l+1)} + \Lambda_{k_{l+1}}^{(l+1)} z_n^{(l+1)})^T \Psi^{-1}_{k_{l+1}^{(l+1)}} \big(z_n^{(l)} - (\eta_{k_{l+1}}^{(l+1)} + \Lambda_{k_{l+1}}^{(l+1)} z_n^{(l+1)})\big) + \frac{r_{l+1}}{2}\ln(2\pi)\Big] + \sum_{n=1}^{N}\sum_{k_{l+1}=1}^{K_{l+1}} s_{n,k_{l+1}}^{(l+1)} \mathrm{E}_{\pi^{(l+1)}}\Big[\ln \pi_{k_{l+1}}^{(l+1)}\Big] + cst$$

$$\iff \ln q^*(s^{(l+1)}) = \sum_{n=1}^{N}\sum_{k_{l+1}=1}^{K_{l+1}} s_{n,k_{l+1}}^{(l+1)}\Big[-\frac{1}{2}\mathrm{E}_{\Theta^{(l+1)}}\big[\ln|\Psi_{k_{l+1}}^{(l+1)}| + (z_n^{(l)} - (\eta_{k_{l+1}}^{(l+1)} + \Lambda_{k_{l+1}}^{(l+1)} z_n^{(l+1)})^T \Psi^{-1}_{k_{l+1}^{(l+1)}} \big(z_n^{(l)} - (\eta_{k_{l+1}}^{(l+1)} + \Lambda_{k_{l+1}}^{(l+1)} z_n^{(l+1)})\big) + \frac{r_{l+1}}{2}\ln(2\pi)] + \mathrm{E}_{\pi^{(l+1)}}[\ln \pi_{k_{l+1}}^{(l+1)}]\Big] + cst$$

Que l'on peut réécrire comme

$$\ln q^*(s^{(l+1)}) = \sum_{n=1}^{N}\sum_{k_{l+1}=1}^{K_{l+1}} s_{n,k_{l+1}}^{(l+1)} \ln \rho_{n,k_{l+1}} + cst$$

avec $\ln \rho_{n,k_{l+1}} = -\frac{1}{2}\mathrm{E}_{\Theta^{(l+1)}}\big[\ln|\Psi_{k_{l+1}}^{(l+1)}| + (z_n^{(l)} - (\eta_{k_{l+1}}^{(l+1)} + \Lambda_{k_{l+1}}^{(l+1)} z_n^{(l+1)})^T \Psi^{-1}_{k_{l+1}^{(l+1)}} \big(z_n^{(l)} - (\eta_{k_{l+1}}^{(l+1)} + \Lambda_{k_{l+1}}^{(l+1)} z_n^{(l+1)})\big) + \frac{r_{l+1}}{2}\ln(2\pi) + \mathrm{E}_{\pi^{(l+1)}}[\ln \pi_{k_{l+1}}^{(l+1)}]$

On obtient donc en prenant l'exponentielle à droite et à gauche et en spécifiant la constante de normalisation que :

$$q^*(s^{(l+1)}) = \prod_{n=1}^{N}\prod_{k_{l+1}=1}^{K_{l+1}} \left[\frac{\rho_{n,k_{l+1}}}{\sum_{k_{l+1}=1}^{K_{l+1}} \rho_{n,k_{l+1}}}\right]^{s_{k_{l+1}}^{(l+1)}} \tag{4}$$

On dérive maintenant la loi *a posteriori* de $\pi^{(l+1)}, \eta^{(l+1)}, \Lambda^{(l+1)}, \Psi^{(l+1)}$ à partir de la deuxième équation de (1) :

$$\begin{aligned}\ln q^*(\pi^{(l+1)},\Theta^{(l+1)}) &= \mathrm{E}_{s^{(l+1)}}[\ln p(z^{(l)},\pi^{(l+1)},s^{(l+1)},\Theta^{(l+1)}|z^{(l+1)})] + cst' \\ &= \mathrm{E}_{s^{(l+1)}}[\ln p(z^{(l)}|s^{(l+1)},\Theta^{(l+1)}|z^{(l+1)})] \\ &+ \mathrm{E}_{s^{(l+1)}}[\ln p(s^{(l+1)}|\pi^{(l+1)})] + \ln p(\pi^{(l+1)}) \\ &+ \sum_{k_{l+1}=1}^{K_{l+1}} \ln(p(\eta_{k_{l+1}}^{(l+1)})p(\Lambda_{k_{l+1}}^{(l+1)})p(\Psi_{k_{l+1}}^{(l+1)})) + cst' \end{aligned} \tag{5}$$

On constate qu'il n'existe pas de termes impliquant à la fois $\pi^{(l+1)}$ et $\Theta^{(l+1)}$, ce qui signifie que la distribution variationnelle peut se factoriser de la façon suivante :

$$q(\pi^{(l+1)},s^{(l+1)},\Theta^{(l+1)}) = q(s^{(l+1)})q(\pi^{(l+1)}) \prod_{k_{l+1}=1}^{K_{l+1}} \left(p(\eta_{k_{l+1}}^{(l+1)})p(\Lambda_{k_{l+1}}^{(l+1)})p(\Psi_{k_{l+1}}^{(l+1)})\right)$$

On remplace à nouveau les distributions contenues dans l'équation (5) par leurs expressions :

$$\begin{aligned}\ln q^*(\pi^{(l+1)},\Theta^{(l+1)}) &= \sum_{n=1}^{N}\sum_{k_{l+1}=1}^{K_{l+1}} \mathrm{E}_{s^{(l+1)}}\left[s_{n,k_{l+1}}^{(l+1)}\right] \ln \mathcal{N}(z^{(l)}|s^{(l+1)},z^{(l+1)},\Theta^{(l+1)}) \\ &+ \sum_{n=1}^{N}\sum_{k_{l+1}=1}^{K_{l+1}} \mathrm{E}_{s^{(l+1)}}\left[s_{n,k_{l+1}}^{(l+1)}\right] \ln \pi_{k_{l+1}}^{(l+1)} + (\alpha_0-1)\sum_{k_{l+1}=1}^{K_{l+1}} \ln \pi_{k_{l+1}}^{(l+1)} + \ln p(\Theta^{(l+1)}) + cst''\end{aligned}$$

Avec cst'' une constante contenant tous les termes ne dépendant pas de $\pi^{(l+1)}$.

On identifie les termes en $\pi^{(l+1)}$:

$$\begin{aligned}\ln q(\pi^{(l+1)}) &= \sum_{n=1}^{N}\sum_{k_{l+1}=1}^{K_{l+1}} \mathrm{E}_{s^{(l+1)}}[s_{n,k_{l+1}}^{(l+1)}] \ln \pi_{k_{l+1}}^{(l+1)} + (\alpha_0-1)\sum_{k_{l+1}=1}^{K_{l+1}} \ln p(\pi_{k_{l+1}}^{(l+1)}) + cst'' \\ &= \sum_{k_{l+1}=1}^{K_{l+1}} \ln \pi_{k_{l+1}}^{(l+1)} \left[(\alpha_0-1) + \mathrm{E}_{s^{(l+1)}}\left(\sum_{n=1}^{N} s_{n,k_{l+1}}^{(l+1)}\right)\right] + cst''\end{aligned}$$

D'où en prenant l'exponentielle, on aboutit à : $\pi^{(l+1)} \sim Dir\left(\pi^{(l+1)}|\alpha_0 + \mathrm{E}_{s^{(l+1)}}[\sum_{n=1}^{N} s_{n,k_{l+1}}^{(l+1)}]\right)$

On peut à partir de cette distribution "seuiller" à 0 les réalisations de $\pi_{k_{l+1}}^{(l+1)}$ inférieures par exemple au quantile de niveau 5% de la loi de Dirichlet. Il devient donc inutile de continuer à estimer les paramètres des chemins dont le noeud original a une probabilité nulle de jouer

un rôle dans l'explication des données. Plus précisemment, seuiller $\pi_{k_{l+1}}^{(l+1)}$ à 0 entraine la suppression de $\prod_{j=l}^{L} K_j$ chemins à estimer.

Enfin, une fois le prior sur la loi jointe de $\Theta^{(l+1)}$ déterminé, On identifiera les termes afférents dans (5) pour trouver la loi *a posteriori* :

$$\begin{aligned}\ln q^*(\eta^{(l+1)}, \Lambda^{(l+1)}, \Psi^{(l+1)}) = \mathrm{E}_{s^{(l+1)}}[\ln p(z^{(l)}|s^{(l+1)}, \Theta^{(l+1)}|z^{(l+1)})] \\ + \sum_{k_{l+1}=1}^{K_{l+1}} [\ln p(\eta_{k_{l+1}}^{(l+1)}) + \ln p(\Lambda_{k_{l+1}}^{(l+1)}) + \ln p(\Psi_{k_{l+1}}^{(l+1)})] + cst'\end{aligned}$$

Notes sur l'algorithme variationnel local On notera qu'il est aussi possible d'accélérer le modèle en utilisant un sous-ensemble d'observations pour mettre à jour les paramètres des lois utilisées. Ces méthodes appelées méthodes variationnelles locales reposent sur la même idée que la descente de gradient stochastique.

Dans de futurs travaux, réaliser l'entrainement du modèle par Expectation-Propagation pourrait être intéressant. En effet comme l'indique Chopin et Cottet [1], il est possible de paralléliser EP sans trop de pertes d'information grâce à l'algorithme de Cseke et Heskes [8], ce qui accélèrerait considérablement l'entrainement du modèle. Cependant, nous craignons qu'EP, du fait du recours à une divergence de Kullback-Liebler inversée n'ai tendance à trop sur-estimer la variance du modèle. Cette crainte trouve un écho chez Bishop [4] qui juge EP peu adapté aux modèles multimodaux.

2.2.2 Implémentations

Les auteurs ont publié leur implémentation du DGMM l'année dernière. Ce dernier est écrit en R et est disponible à cette adresse : `https://github.com/suren-rathnayake/deepgmm`. Cette implémentation se révèle imparfaite sous deux rapports principaux. Premièrement, il existe un script différent pour chaque nombre de couches défini par l'utilisateur, le nombre total de couche étant limité à 3 (ce qui rend le nom *Deep Gaussian Mixture Models* légérement abusif). Il y a ainsi beaucoup de code dupliqué, ce qui rend l'organisation du package relativement obscur et la création d'extensions du modèle quasiment impossible. Deuxièmement, les opérations ayant lieu sur chacun des chemins du réseau sont traitées séquentiellement alors qu'elles pourraient être parallélisées pour tous les noeuds d'une même couche.

Keras est la plus grande librairie de Deep Learning sur Python à l'heure actuelle. Porter le modèle sur Keras (avec Tensorflow comme *back-end*) permettrait une standardisation appréciable et de profiter des possibilités de parallélisation sur CPU, GPU (ou TPU) facilitées par l'utilisation des tenseurs. Cela rendrait également le modèle accessible à une nouvelle

communauté. J'ai donc dans un premier temps traduit le code de leur package R pour créer un package Python qui se base principalement sur numpy. Plus précisemment, j'ai traduit uniquement la partie du package relative à l'architecture DGMM à 3 couches, dans la mesure où elle généralisait celles des DGMMs avec un nombre de couches inférieur. Une fois ceci fait, j'ai commencé la mise aux normes Keras.

Traduction du package existant de R à Python

Cette première étape bien que laborieuse m'a semblé nécessaire afin de réellement comprendre l'organisation et les détails de fonctionnement de l'algorithme.

En effet, l'article et la documentation du package donnent peu d'indications sur les détails d'implémentation. Par exemple, la façon de gérer le stockage des données correspondant à l'ensemble des chemins, l'implémentation des contraintes d'identification du modèle ou encore les détails de la stabilité numérique de l'algorithme ne sont pas explicités.

Si la traduction littérale du code a été relativement aisée, le débuggage du code a, lui, été très laborieux (plus de deux semaines) du fait des différences de traitement entre R et Python. L'implémentation des algorithmes comme l'ACP ou encore l'analyse de facteurs ne renvoit par exemple pas toujours les mêmes résultats entre les deux langages, que cela soit en termes de formats comme de valeurs. Afin de résoudre ces problèmes, j'ai fini par utiliser le package R2py, qui permet de convertir des objets R pour les importer en Python. Pour chaque opération, j'ai donc pu injecter les mêmes valeurs en entrée et comparer les sorties renvoyées par les deux implémentations.

Au terme de ce travail de traduction, les résultats obtenus par la version Python et R sont quasiment identiques. Les différences sont dues principalement à la façon d'assurer la stabilité numérique de l'algorithme. En effet, les matrices de variance-covariance générées par l'algorithme ne sont pas toujours semi-définies positives ou de plein rang. Il semble que la fonction *rmvnorm* de R soit moins sensible à cela que la function *multivariate_normal* de Scipy (Python) car elle accepte d'évaluer la densité pour des matrices de variance-covariance que Python considère comme singulières.

On peut tout de même forcer Python à procéder à cette évaluation, mais les estimations de densité deviennent alors très différentes de celles réalisées par R (tandis qu'elles sont strictement identiques pour les matrices semi-definies positives). Une autre alternative est de chercher la matrice semi-définie positive la plus "proche" de celle générée par l'algorithme pour pouvoir évaluer la densité de probabilité associée. J'ai pour cela utilisé un algorithme de Nick Higham : `https://nickhigham.wordpress.com/2013/02/13/the-nearest-correlation-matrix/`.

Les estimations changent encore une fois beaucoup par rapport à celle de R en utilisant cette option.

Le but de ces manipulations est de gérer les cas extrêmes/limites générés par l'algorithme. A ce titre, je pense qu'il est difficile d'affirmer qu'il existe une "meilleure façon" de procéder. Il faudrait néanmoins évaluer l'impact de chacune de ces options sur les prédictions.

De manière plus générale, la complexité globale de l'algorithme semble pousser à l'émergence de ces cas extrêmes. c'est par exemple le cas pour les matrices $(\Psi_{k_l}^{(l)})_{k_l \in [1,K_l], l \in [1,L]}$ qui comportent souvent certaines valeurs diagonales proches de la précision machine (aux environs de 10E-40). Une fois inversées et multipliées avec d'autres termes (ce qui arrive souvent étant donné que la plupart des variables du modèle suivent des Gaussiennes) on aboutit donc à des termes très variables d'une exécution à l'autre et dont la norme est très grande.

Une fois la version Python du script opérationnelle, j'ai réalisé un premier test pour me faire une idée de la capacité du DGMM à passer à l'échelle, c'est-à-dire à traiter de gros flux de données en un temps raisonnable. J'ai donc lancé plusieurs fois l'algorithme en faisant croître le nombre d'observations.
J'ai comparé le temps d'exécution moyen d'un DGMM à trois couches avec respectivement 4 neurones, 2 neurones et 2 neurones (noté DGMM 4-2-2), d'un DGMM 4-4-4 et celui du K-means en faisant varier le nombre d'observations.
Les observations sont des données synthétiques générées par un mélange de quatre gaussiennes de moyennes et variances différentes, faciles à identifier. En faisant croître le nombre d'observations total (chaque groupe a le même nombre d'observations), on obtient les temps de calculs suivants :

Le temps affiché est le temps moyen pour 15 exécutions de l'algorithme. Doubler le nombre de neurones sur les deux dernières couches semble donc doubler le temps d'exécution. Le temps d'exécution apparait également globalement linéaire du nombre d'observations pour les DGMMs. La complexité de l'algorithme, à nombre de couches fixé, a donc l'air d'être environ en $o(n \times k)$ avec n le nombre d'observations et k le nombre total de neurones (en dehors de ceux situés sur la première couche dont le nombre est fixé par le nombre de groupes à discerner dans les données). Passer en Keras pourrait permettre de paralléliser les opérations sur chaque couche et de se débarasser du facteur multiplicatif k de la complexité. A nombre de couches fixé, faire tourner un modèle avec 2 neurones sur chaque couche ne devrait alors pas coûter vraiment plus cher qu'un modèle avec 8 neurones par couche.

Implémentation du modèle en Keras

Keras est une librairie très récente (stable depuis décembre 2017) spécialement conçue pour

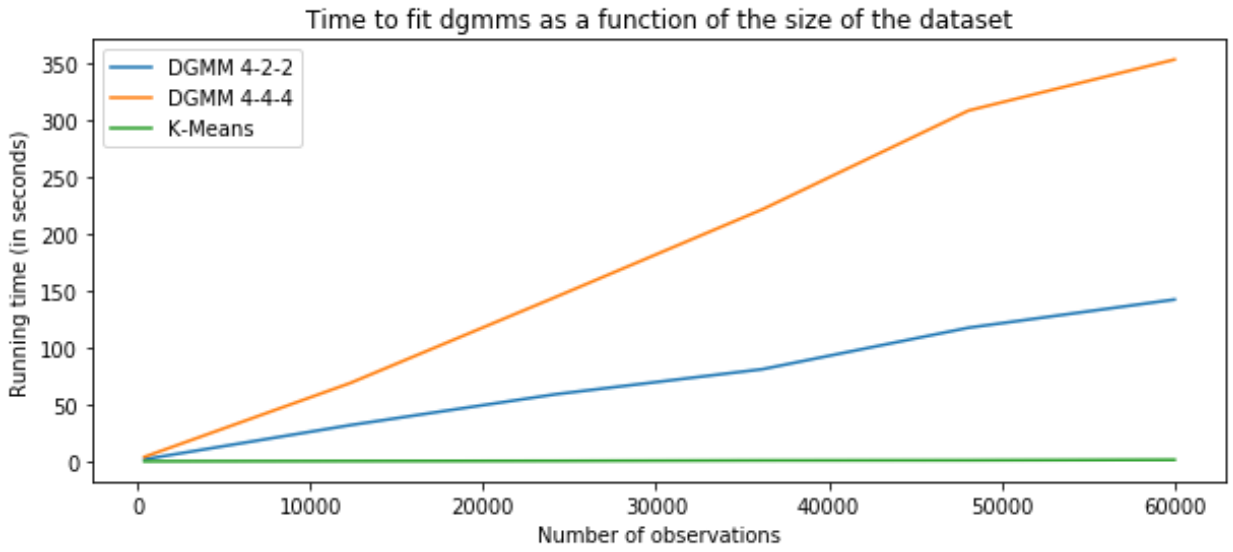

FIGURE 3 – Evolution du temps moyen d'exécution de deux spécifications DGMMs et des k-means lorsque le nombre d'observations croît

le développement de modèles de Deep Learning supervisés. Elle impose une structure au code relativement rigide bien que naturelle. Les couches aussi bien que les modèles eux-mêmes sont des objets qui peuvent être appelés et qui héritent des classes *Layer* et *Model* de Keras.

Pour les couches, il est nécessaire de définir au moins quatre méthodes de classes :

— **__init__** : comme pour l'ensemble des objets, cette méthode initialise l'objet en ses attributs et en fixant leurs valeurs.

— **build** : Cette méthode permet de définir et d'initialiser les poids à entrainer du modèle. Dans notre cas, elle initialisera les poids à l'aide d'analyse factorielle ou de d'Analyse en Composante Princale Probabiliste (PPCA). Dans notre version bayésienne du modèle, les poids seront initialisés à l'aide des priors nouvellement définis.

— **call** : C'est la méthode appelée lors de l'entrainement du modèle. Dans la version originale du modèle cela revient à exécuter les étapes SEM et à actualiser les poids. Dans notre extension cela revient à tirer des observations à partir des distributions variationnelles et à raffraichir les poids.

— **compute_output_shape** : Donne la dimension de $z^{(l)}$ renvoyée par la couche (l), *i.e.* $N \times r_l$.

—

De même, pour créer la classe correspondant au modèle, certaines méthodes et attributs doivent obligatoirement être définies au préalable. Ce faisant, Keras est capable d'entrainer efficacement les modèles de Deep Learning malgré le nombre substantiel de paramètres à

estimer. Keras permet également une grande modularité dans le sens où il est très facile de changer la méthode d'entrainement, la fonction de perte ou le nombre de couches du modèle. La plupart des modèles et couches présents sur Keras appartiennent au champs de l'apprentissage supervisé. Cependant des modèles comme les Variational AutoEncoders (VAE) ou encore les Generative Adversarial Networks (GAN) sont aussi présents sur Keras, ce qui m'a laissé penser qu'il était possible de porter les DGMMs sur Keras.

Pour profiter réllement des avantages liés à Keras, il est nécessaire de convertir les opérations vectorielles et matricielles de numpy en opérations tensorielles. J'espère finir ce travail avant la fin du mois de novembre.

3 Caractériser le lien structure phytoplanctonique/variables environementales

Les applications statistiques en océanographie constitue le deuxième volet de cette thèse. La nature des phénomènes océanographiques se prête bien à l'utilisation de méthodes de *clustering* en haute dimension comme les DGMMs. La taille des jeux de données actuels rend également obligatoire l'utilisation de modèles passant à l'échelle. L'augmentation de la taille des jeux de données résulte avant tout d'avancées technologiques comme la cytométrie en flux, qui sera la source de données principale de cette thèse.

Le principe général du cytomètre en flux est le suivant : un échantillon d'eau à analyser est pompé depuis l'environnement étudié dans le cytomètre. Le cytomètre aligne une à une les cellules contenu dans l'échantillon et les fait passer devant un faisceau laser. On récupère pour chaque cellule analysée un profil optique, qui prend la forme d'un jeu de courbes. La capacité de traitement du cytomètre peut aller jusqu'à 10.000 cellules à la seconde, ce qui génère donc des jeux données rapidement massifs.

La cytométrie en flux a permis d'accroitre de manière très importante les capacités de collecte de données relatives au phytoplancton par rapport aux anciennes techniques microscopiques qui voient leurs origines remonter aux années 1840 [20].

Si la collecte a largement été automatisée, le traitement des données récoltées est, lui, encore loin de l'être totalement. Certaines opérations de traitement continuent en grande partie d'être réalisées à la main et sont très chronophages. Une des tâches préliminaires de cette thèse est donc d'automatiser certains de ces processus pour pouvoir conduire des analyses sur les données ainsi acquises.

3.1 Classification des différents groupes fonctionnels du phytoplancton

Les tâches à automatiser sont exclusivement des tâches de classification et de *clustering*. Il s'agit plus précisemment d'isoler différents groupes de phytoplancton à partir des signaux envoyés par le cytomètre. En effet, si le phytoplancton est regroupé sous une dénomination unique, il est en réalité un regroupement polyphylétique de plusieurs dizaines de milliers d'espèces.

Nombre d'entre elles présentent cependant des comportements et des caractéristiques très similaires, ce qui a permis d'établir une nomenclature de différents groupes homogènes appelés groupes fonctionnels. C'est sur cette nomenclature comportant au maximum 12 classes différentes que se baseront les analyses qui suivent.

Cette dernière est cependant loin d'être une convention pour tous les océanographes. Une approche non-supervisée validant la nomenclature pourrait donc être envisagée dans un second temps. Pour résumé, le tableau 1 donne les groupes fonctionnels contenus parmi les données à ma disposition pendant le stage ainsi qu'un ordre de grandeur de leur taille :

Groupe fonctionnel	Ordre de grandeur de la taille
Synecchoccus	$6,0 \times 10^{-7}m$
Picoeucaryote	$2,0 \times 10^{-7}m$ à $2,0 \times 10^{-6}m$
Prochlorococcus	$6,0 \times 10^{-7}m$ à $1,7 \times 10^{-6}m$
Cryptophytes	$1,0 \times 10^{-6}m$ à $5,0 \times 10^{-5}m$
Nanoeucaryote	$2,0 \times 10^{-6}m$ à $2,0 \times 10^{-5}m$
Microphytoplancton	$2,0 \times 10^{-5}m$ à $2,0 \times 10^{-4}m$

TABLE 1 – Nomenclature des groupes fonctionnels disponibles dans les données ainsi que leurs tailles

Le signal renvoyé dans notre cas par le cytomètre comprend 5 courbes optiques pour chaque cellule qui traverse son faisceau laser : deux courbes de diffusion et trois courbes de fluorescence.
Attribuer les cellules aux différents groupes fonctionnels du phytoplancton à partir des cinq courbes optiques qui les caractérisent est une tâche relativement aisée pour un cytométriste. Cette dernière se fait en projetant successivement deux courbes parmi les cinq dans un plan et en délimitant des zones dans le nuage de points généré. L'ensemble des zones délimitées au cours des différentes projections permet ensuite au logiciel d'établir des zones discriminantes dans l'espace en 5 dimensions.

Cette tâche est cependant fastidieuse, subjective et consommatrice de temps. Les données recueillies par exemple lors des campagnes en mer ne peuvent pas être analysées en temps

réel et demandent un travail conséquent de classification pour être exploitables en tant que telles.
L'utilisation de techniques d'apprentissage supervisé pourrait donc être un recours intéressant. Les méthodes issues de l'apprentissage statistique ont commencé à être utilisées relativement tôt dans le champ de l'océanographie. Beaufort et Dolfus (2004) [2] avaient déjà recours à des réseaux de neurones peu profonds pour effectuer des opérations de dénombrements d'espèces.
Plus récemment, Malkassian (2011) [16] a proposé une méthode de reconnaissance d'espèces phytoplanctoniques basée sur les 5 courbes optiques mentionnées. L'analyse se concentre sur la forme des courbes, qui ressemblent globalement à des Gaussiennes et calcule des distances entre les courbes des différentes cellules grâce à une projection dans la base de Fourier.

Encore plus récemment, un grand nombre de projets de recherche sur ce sujet semble avoir émergé. Ils sont en grande partie fondés sur l'utilisation de réseau de neurones profonds. On peut citer par exemple Dunker (2019) [10], qui a recours au *transfer learning* avec le modèle ResNet v2 pré-entrainé sur le jeu de données ImageNet ILSVRC 2012 pour classer certaines espèces de phytoplancton. L'application d'un post-traitement additionnel basé sur les connaissances océanographiques permet à Dunker de parvenir à une précision moyenne très impressionnante (plus de 97%).

Il est cependant à noter que les bonnes performances obtenues par Malkassian et par Dunker sur la classification d'espèces ne sont pas directement transposables à la classification des groupes fonctionnels. En effet, les membres d'une même espèce possèdent des caractéristiques physiques (et donc optiques) plus proches que les membres d'un même groupe fonctionnel qui se comportent de la même façon mais ne se ressemblent pas obligatoirement.
Enfin d'autres projets de recherche comme ceux ayant cours au CEREGE portent également sur la classification automatisée appliquée au phytoplancton.

Le présent travail est donc avant tout exploratoire et vise à être intégré dans d'autres projets existants à plus long terme.
Ce stage était également l'occasion de confronter les résultats obtenus par des réseaux de neurones, aujourd'hui très populaires, avec d'autres méthodes d'apprentissage plus "classiques".
Les 5 courbes optiques renvoyées par le cytomètre sont en réalité un ensemble de points : Le premier point est le signal acquis lorsque le début de la cellule intercepte le laser et le dernier point lorsque la cellule a fini de traverser le laser. On peut donc traiter ces valeurs en tant que telles, mais également calculer des *features* comme des moyennes roulantes ou encore le nombre de modes de chaque courbe optique et ajuster un modèle de classification sur ces *features*. Il est aussi possible de générer les images correspondant à ces courbes et de

mener la classification sur les images de courbes. Enfin, il sera bientôt possible d'utiliser des photos prises par le cytomètre pour classer les groupes fonctionnels dont les cellules ont une taille supérieure à 2 microns.

J'ai réalisé des essais de classification pour avec trois classes à l'aide des premières photos renvoyées par le cytomètre. Ces analyses préliminaires sont disponibles dans le notebook "Particles imgs Demo" sur mon Github à l'adresse suivante : `https://github.com/RobeeF/planktonPipeline/blob/master/Particles%20imgs%20Demo.ipynb`.

3.2 Résultats de la reconnaissance automatisée d'espèces phytoplanctoniques

L'ensemble des approches de classification evoquées se heurtent néanmoins à un certain nombre de problèmes inhérents aux données utilisées.

Premièrement, le milieu océanonique pousse les espèces de très petite taille à proliférer bien plus que les plus grandes espèces. Dans un échantillon de quelques millilitres, le rapport d'abondance entre l'espèce la plus represéntée et la moins représentée avoisine en général 10^4 ou même 10^5. Cela pose des problèmes classiques de jeux de données déséquilibrés.
Afin de pallier ce problème, la mise en place de méthodes de sous-échantillonnage ou sur-échantillonnage est obligatoire.

De plus, afin de limiter la part de bruit dans le signal total collecté, le cytomètre est réglé avec différents seuils de fluorescence. Le laser bleu du cytomètre excite la Chlorophyle A contenu dans le phytoplancton qui se désexcite en émettant des rayonnements allant du orange au rouge. Cependant certains débris comme des cellules en décomposition, ou certains prédateurs du phytoplancton, comme le microzooplancton, contiennent un peu de chlorophylle A et fluorescent. Afin d'éliminer ce bruit parasite, la sensibilité du laser est en général fixée à deux seuils. Un seuil faible (FLR6 dans notre cas) pour acquérir les profils optiques des petites espèces du phytoplancton, au risque de capturer beaucoup de bruit s'il est trop bas, et un seuil haut (FLR25 dans notre cas) pour capturer uniquement les profils des plus grosses espèces. Il faut donc vérifier qu'à échantillon et groupe fonctionnel donnés, les profils collectés avec chaque seuil soient comparables.

Toutes ces corrections et tests sont réalisés au sein d'un *pipeline* qui récupèrent les données cytométriques dans un dossier, génèrent des fichiers csv directement analysables, redressent les échantillons déséquilibrés et prédit le groupe fonctionnel de chaque cellule. Ce pipeline est disponible à l'adresse suivante : `https://github.com/RobeeF/planktonPipeline`

3.2.1 Caractérisation par les images des courbes

La première idée explorée était la classification à partir des images de courbes. L'idée sous-jacente était que si un cytométriste parvient à distinguer les groupes fonctionnels à partir de l'image de la courbe sur le moniteur, un réseau de neurones pourrait aussi. Un exemple de courbe générée pour la fluorescence Orange d'un cryptophyte est donné par la figure 4.

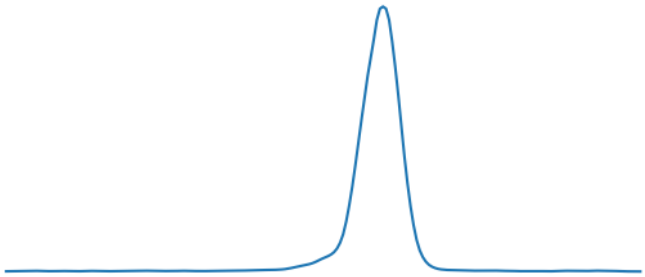

FIGURE 4 – Exemple de courbe de Fluorescence Orange pour un cryptophyte

La forme de la courbe montre que la cellule a une forme proche de celle d'une madeleine dont le bord gauche serait légèrement plus long que le bord droit. L'essentielle de la Chlorophylle A est contenue dans le "dôme" de la madeleine qui correspond au pic sur la courbe.

Les images des 5 courbes ont toutes pour dimension $(200, 300)$, ce qui les rend directement comparables (les courbes initiales ne sont pas de mêmes longueurs).
Ces dernières sont ensuite transmises "en streaming" du dossier de stockage vers le réseau de neurones par un générateur. L'intérêt d'utiliser un générateur est que l'on peut lui demander de créer de nouvelles courbes "à la volée" à partir de celles qu'il transmet, c'est-à-dire sans les stocker sur le disque dur. On peut donc par exemple appliquer à chaque courbe une rotation de 180 degrés pour rendre le réseau plus robuste à ces rotations. En effet, si les cellules sont alignées une à une avant de passer par le faisceau, le "sens" par lequel elles traversent le laser n'est pas controllable. Ainsi si la cellule est fortement dyssimétrique horizontalement et/ou verticalement, deux sens de passage différents entraineront deux jeux de courbes différents (plus précisément : égaux à une rotation près). Appliquer des rotations aux courbes avant de les passer au réseau permet donc de le rendre invariant à la rotation. Rendre plus robustes à la rotation d'autres modèles d'apprentissage qui reposent sur des distances fonctionnelles demande plus de travail, comme le montre le papier de Malkassian (2011) [16].

Le réseau utilisé ici est un Réseau de Neurones Convolutionnel (CNN), modèle aujourd'hui standard pour le traitement d'image. Ce dernier comporte une architecture relativement

simple, donnée par la Figure 5.

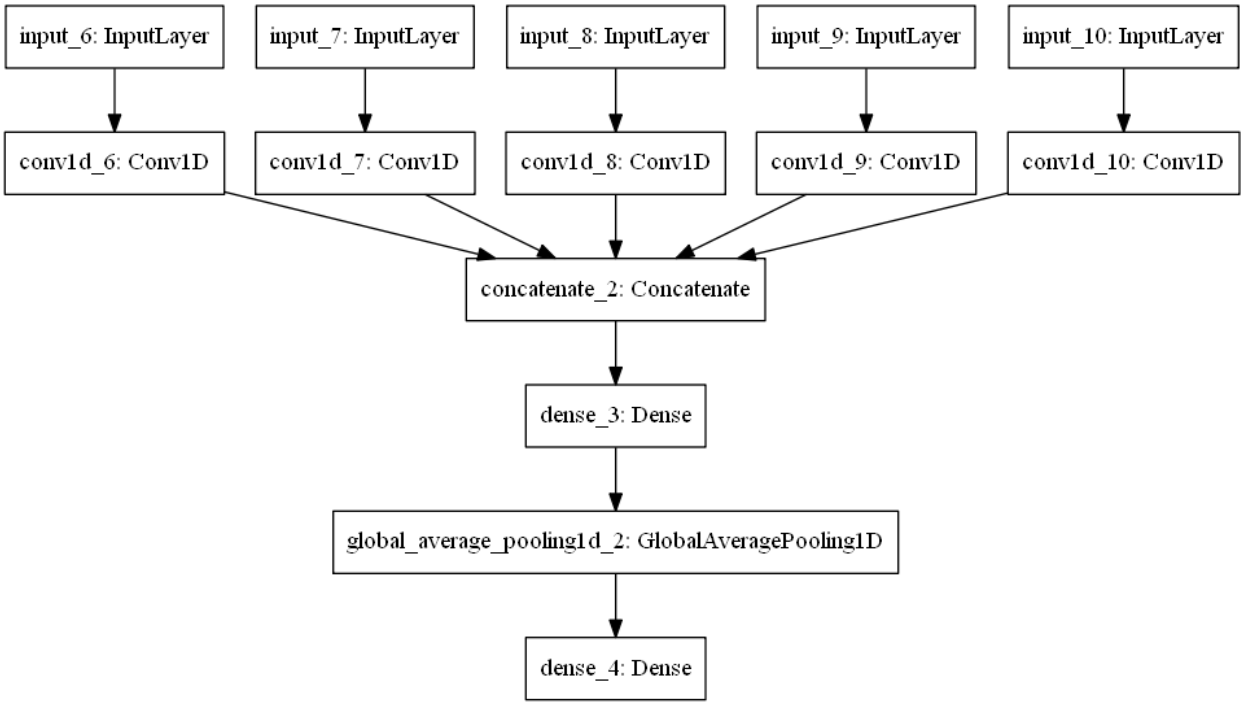

FIGURE 5 – Architecture du CNN pour le traitement des images de courbes

Ce réseau prend donc en entrée les 5 courbes (Courbure, Fluorescence Rouge, Fluorescence Orange, indice de réfraction), ce qui correspond aux Inputs 6 à 10 (les numéros n'ont pas d'importance). Chacune de ces courbes passe ensuite par une couche de convolution. Cette couche permet d'appliquer des "filtres" afin de décomposer le signal de chaque courbe en plusieurs signaux caractéristiques utiles pour différencier chaque groupe de phytoplanctons. Ces signaux caractéristiques sont ensuite rassemblés au niveau de la couche de concaténation. Le nouveau signal obtenu passe après ça par une couche dense. Le principe de la couche dense est proche de celui de la couche convolutionnelle. Cependant une couche dense comporte beaucoup plus de paramètres qu'une couche convolutionnelle. Les signaux qui en sont extraits sont donc plus informatifs, au risque de laisser passer des informations sans importance qui perturbent l'entraînement du réseau. La couche de "Global Average Pooling" qui suit compresse le signal en le résumant par un ensemble de moyennes locales. Finalement, la dernière couche dense renvoie un vecteur correspondant aux probabilités que les courbes données en entrée appartiennent à chacuns des groupes fonctionnels du phytoplancton considérés.

Les résultats de la classification des images s'est montré très décevant. En effet, il semble de le réseau ne parvienne pas à apprendre des courbes : il classe systématiquement l'ensemble des observations dans une seule classe (qui n'est pas la même d'un lancement de l'algorithme

sur l'autre). Afin de vérifier que le problème n'était pas lié au phénomène de "Vanishing Gradients", j'ai plusieurs fois changé l'architecture du réseau pour quelque chose de plus simple : prédiction à partir d'une seule courbe à la fois, une seule couche dense et pas de "Global Average Pooling", changement d'optimiseur et de taux d'apprentissage, sans que le résultat ne soit différent. Le problème de "Vanishing Gradients" renvoie au fait que certains réseaux trop profonds ne parviennent pas à faire remonter l'information jusqu'à la sortie du réseau, se traduisant par des gradients très proches de zéro et une mise à jour des poids quasiment nulle.
Enfin, j'ai essayé de fournir les données au réseau sans passer par le générateur et changé le préprocessing des images, sans succès.

Une des raisons potentielles qui pourrait expliquer ce phénomène est que le signal est trop dilué. En effet, le fait d'afficher la courbe sur un fond blanc revient à noyer l'information (les pixels bleus de la courbe) dans un très grand nombre de pixels blancs. Les images se ressemblent donc trop et les filtres successifs ne parviennent pas à en extraire de l'information. Sur les conseils de mes encadrants, j'ai appliqué à l'entrée du réseau une sorte de filtre "zoomant" pour pré-traiter les images de courbes. Ce filtre applique une moyenne mobile sur toutes les lignes de pixels de l'image avant de la passer au reste du CNN, ce qui revient dans l'idée à ajouter une couche convolutive particulière au début du modèle . Cette couche fait "déteindre" les pixels de la courbe sur les pixels voisins pour réduire la proportion de pixels blancs de l'image. Le niveau de "zoom" correspond à la fenetre de la moyenne mobile : Un trop grand "zoom" rend les courbes non identifiables tandis qu'un trop petit niveau laisse le signal trop faible. Les figures 6, 7 et 8 illustrent les images de courbes obtenues par ce biais pour différents niveaux de "zoom".

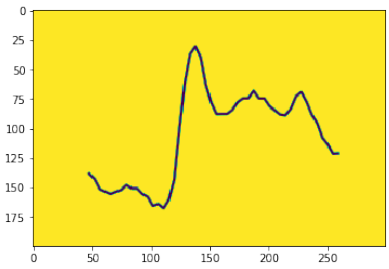

FIGURE 6 – Signal original

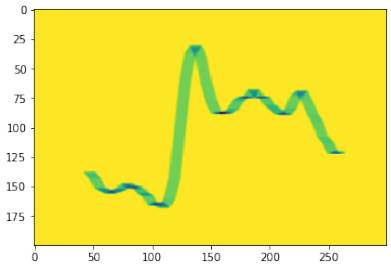

FIGURE 7 – Moyenne mobile horizontale de fenêtre 10 appliquée au signal

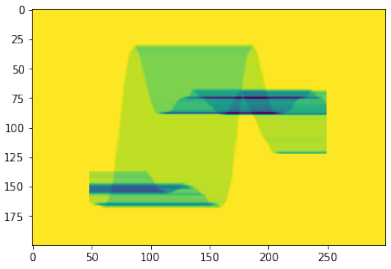

FIGURE 8 – Moyenne mobile horizontale de fenêtre 100 appliquée au signal

Ce prétraitement s'est avéré insuffisant, le réseau n'apprenant toujours pas. A la lumière de ces résultats, il semble que la classification à l'aide d'images de courbes ne soit pas une voie prometteuse.

D'autre part, cette méthode présentait également d'autres inconvénients. Générer et stocker plusieurs centaines de milliers d'images était en effet très coûteux en termes de temps et d'espace de stockage. Travailler avec les valeurs numériques qui servent à construire ces courbes semble à ce regard plus avantageux : Le signal est non dilué et l'entrainement du modèle plus rapide.

Un notebook de présentation permettant de faire varier le niveau de zoom et présentant les résultats de cette sous-partie est disponible sur le Github (sous le nom de Pulse curves demo).

3.2.2 Caractérisation par les valeurs des courbes

Les valeurs des courbes se présentaient donc sous la forme de tableau tri-dimensionels : (Nombre d'observations, 5 courbes, longueur du signal de chaque courbe).

Compte-tenu de la nature de ces données, trois modèles d'apprentissage profonds semblaient particulièrement adaptés : Un Feed-Forward Neural Network (FFNN), un Long Short Term Memory (LSTM) et un Réseau de Neurones Convolutionnel (CNN). Le FFNN, aussi appelé "Multi-layer Perceptron" (MLP) est le réseau de neurones le plus ancien, constitué uniquement de couches denses. En conséquence, le nombre de paramètres est bien plus élevé que dans les CNNs, ce qui les rend par exemple peu adaptés au traitement d'images (trop longs à entrainer).

Le LSTM est un réseau de neurones récursif, c'est-à-dire qu'il capture la dépendance qui existe entre les valeurs ordonnées d'une même séquence. Les LSTMs sont par exemple utilisés pour traiter des données textuelles dans la mesure où il existe une dépendance entre les mots d'une même phrase (ce qui confère son sens à la phrase), ou pour des séries temporelles. Dans notre cas, la particule passe devant le laser dans le sens de la longueur (ou de la largeur) et les signaux qui en résultent ressemblent donc à des séries temporelles : La valeur renvoyée par le laser à la seconde s est corrélée avec celle renvoyée à la seconde $s + 1$. A ce titre, utiliser un LSTM faisait sens.

Concernant le fonctionnement du CNN, je renvoie à la sous-partie précédente.

La longueur des séquences de valeurs pour chaque courbe est variable entre les cellules. La grande majorité des séquences comporte moins de 100 valeurs mais un petit nombre de séquences peuvent atteindre 500 valeurs. Cependant, les réseaux de neurones prennent des séquences de tailles fixes en entrée : j'ai donc tronqué les séquences plus longues que 104 valeurs et complété les séquences plus courtes avec des zéros, ce qui est une pratique courante notamment en *Natural Langage Processing*. Le choix d'une longueur de 104 a été effectué en prenant la valeur de l'avant-dernier quantile de la distribution des longueurs des séquences.

Il pourrait cependant être fixé par validation croisée.

Pour avoir un jeu de données équilibré, j'ai mis en place un sous-échantillonnage très strict : j'ai gardé un nombre d'observations par groupe égal au nombre d'observations du groupe le plus sous-représenté (à un epsilon près). Les cryptophytes et le microphytoplancton sont très peu représentés dans l'ensemble des fichiers (quelques dizaines d'observations), ce qui contraint beaucoup la taille du jeu de données.

On aboutit donc à un jeu de données bien équilibré d'environ 1500 observations (une taille très faible au regard des centaines de milliers d'observations contenues dans les fichiers). Pour pouvoir entrainer les réseaux avec plus de données, plusieurs solutions existent :

— Assez logiquement : récolter de nouvelles données pour les classes qui en manquent.
— Exclure les classes minoritaires de la prédiction (on peut donc prendre beaucoup plus de données des classes majoritaires et toujours avoir un jeu de données équilibré)
— Créer des données synthétiques avec des méthodes d'oversampling/undersampling telles que SMOTE, ENN ou SMOTEEN. Je ne sais juste pas à quel point ces méthodes arrivent à reproduire la structure de courbes des données.

Parmi les 1500 observations retenues, deux tiers de ces observations sont utilisées pour entrainer chaque réseau (FFNN, LSTM, CNN) et l'autre tiers pour tester leurs qualités prédictive (à ce stade il n'est pas encore nécessaire de créer un jeu de validation).

Les f1-scores, moyenne pondérée entre la précision et le rappel, obtenus par les 3 modèles sur le jeu de test sont donnés dans le tableau 2. On rappelle que la précision est la proportion de particules appartenant effectivement à la classe i parmi toutes celles identifiées comme faisant partie de la classe i par l'algorithme. Le rappel (ou recall) est la proportion de particules appartenant effectivement à la classe i parmi toutes les particules de la classe i existant dans le jeu de données.
Il existe un arbitrage précision/rappel et obtenir une précision et un rappel proches de 1 constitue l'horizon de tout algorithme supervisé. Par exemple, un algorithme qui prédirait "bruit" pour toutes les particules aurait un rappel de 1 et une précision relativement faible pour la catégorie "bruit".

Les paramètres des modèles et le nombre d'époques d'entraînement ont été sélectionnés par validation croisée. La taille du batch est de 32 et l'optimiseur est ADAM avec un taux d'apprentissage de $1E^{-3}$ pour l'ensemble des modèles.
Le FFNN a été entrainé sur 80 époques, le LSTM sur 7 et le CNN sur 32 (les deux derniers modèles sur-apprennent gravement lorsque le nombre d'époques est supérieur). Enfin, un "drop-out" de 0.2 a été appliqué au FFNN pour réduire le sur-apprentissage. Le principe du drop-out consiste à inhiber certaines connexions entre neurones de couches successives **à l'entrainement** : cela agit comme un régulariseur. Un drop-out de 0.2 signifie que 20% des

connexions ont été inhibées à l'entrainement.

Groupe fonctionnel	f1-score FFNN	f1-score LSTM	f1-score CNN	Nombre d'observations
airbubbles	**0.97**	0.85	0.00	14
cryptophyte	**0.84**	0.54	0.59	67
microphytoplancton	**0.86**	0.82	0.78	62
nanoeucaryote	**0.87**	0.59	0.24	61
bruit	**0.78**	0.12	0.40	74
picoeucaryote	**0.96**	0.80	0.86	77
prochlorococcus	**1.00**	0.70	0.70	69
synechococcus	**0.99**	0.92	0.93	80

TABLE 2 – f1-scores des 3 modèles sur le jeu d'entraînement

Les airbubbles sont comme leur nom l'indique des bulles d'air dont la reconnaissance à part entière possède un intérêt pour les océanographes.

Les meilleurs f1-scores sont obtenus pour l'ensemble des classes par le FFNN. Ce dernier comporte certes plus d'époques d'entrainement que les autres mais est le plus simple en termes d'architecture. Chaque époque est donc bien plus courte que celles des deux autres modèles, ce qui relativise voire annule le surcoût d'entraînement auquel on aurait pû s'attendre.

Si les résultats me semblent solides en termes relatif (le FFNN surpasse réellement les deux autres modèles dans cette tâche), la performance absolue doit être légèrement relativisée. Premièrement, le nombre de jeux de paramètres testés par validation croisée étaient relativement faible (environ 5 jeux différents par modèle). Deuxièmement, le nombre d'observations d'entraînement est vraiment faible pour ce type de modèle, plus de données sont nécessaires pour réellement conclure.

Des résultats plus détaillés sont disponibles dans le notebook "Pulse values Demo" sur le Github.

3.2.3 Caractérisation par les features calculées sur les courbes

Le signal original (les 5 courbes) peut-être décomposé en plusieurs *features*. En effet, l'avantage des réseaux de neurones est qu'ils apprennent les features qui sont réellement importantes. Cependant, ces derniers nécessitent un très grand nombre de données d'entraînement. Pour la taille de notre jeu de données, calculer des *features* et entrainer des modèles d'apprentissage statistique classique peut aboutir à de meilleurs résultats. J'ai donc cherché à construire un contrefactuel solide afin de comparer les résultats avec ceux issus des réseaux de neurones.

Les *features* utilisées sont calculées directement par le logiciel d'analyse cytométrique. Cette quarantaine de variables sont parmi les plus utilisées en cytométrie (aire sous la courbe, des indices de courbure, valeur maximale/minimale etc...).

Le classifieur utilisé pour le contrefactuel est une Forêt Aléatoire, très fréquemment utilisée depuis l'article fondateur de Breiman (2001) [6]. Concernant le déséquilibre du jeu de données, j'ai comparé les performances du modèle avec et sans sur-échantillonnage. J'ai donc évalué le modèle sur un jeu de données avec un schéma de sous-échantillonnage identique à celui précédemment décrit et sur un jeu de données augmenté à l'aide de la méthode SMOTEEN.

Le principe général de SMOTEENN tient en deux étapes : générer des données synthétiques pour chaque classe à partir des données existantes (partie SMOTE de l'algorithme), puis sous-échantilloner parmi les observations (d'origine et synthétiques) pour ne garder que les plus informatives et discriminantes (partie ENN). Les algorithmes SMOTE et ENN sont deux algorithmes dissociés, mais combiner les deux donne souvent empiriquement de meilleurs résultats.

La sélection des "meilleurs" paramètres du modèle a une fois encore été conduite par validation croisée. Les paramètres testés étaient le nombre d'arbres, leur profondeur maximale, le critère de séparation des branches ainsi que le nombre maximum de *features* à considérer pour effectuer ces séparations.

Les f1-scores présentés dans le tableau 3 sont ceux obtenus par les deux modèles présentant chacun la meilleure performance cross-validée sur les jeux d'entrainement respectivement avec et sans SMOTEENN . Le jeu de test est bien entendu, lui, le même pour les deux modèles entrainés.

Groupe fonctionnel	f1-score jeu sous-échantilloné	f1-score jeu SMOTEENN
cryptophyte	0.34	**0.60**
microphytoplancton	0.79	**0.84**
nanoeucaryote	0.82	**0.87**
bruit	0.88	**0.95**
picoeucaryote	0.69	**0.87**
prochlorococcus	0.52	**0.88**
synechococcus	**0.99**	**0.99**

TABLE 3 – Performances du meilleur modèle validé croisé sur données sous-échantillonné et SMOTEENN

Le schéma d'échantillonnage SMOTEENN semble apporter un gain de performance par rapport au sous-échantillonage strict. Les performances de ce classifieur sont en revanche en deça des performances du FFNN opérant directement sur les valeurs des cinq courbes.

Des résultats plus détaillés sont disponibles sur le notebook "Listmode Demo" sur Github.

3.3 Caractériser le lien existant entre distribution des groupes fonctionnels et variables environnementales

Les résultats obtenus par le FFNN sur les valeurs des courbes sont très encourageants. "Mettre en production" un tel modèle entrainé pourrait donc permettre de faire de la reconnaissance en temps réel de groupes fonctionnels et de leurs abondances. En particulier, cela permettrait de connaitre l'évolution de la distribution jointe des différents groupes fonctionnels au cours du temps. Nous appelerons cette distribution jointe "structure des groupes fonctionnels du phytoplancton" ou par abus de langage "structure phytoplanctonique".

Caractériser la structure des communautés phytoplanctoniques se trouvant à un moment et à un endroit donnés en fonction d'un certain nombre de facteurs est une des problématiques de recherche en océanographie que cette thèse se propose d'explorer. Cette dernière sous-partie se veut donc essentiellement prospective et pose les bases des développements statistiques en océanographie à venir.

La dynamique et la structuration des groupes fonctionnels du phytoplancton sont gouvernées par plusieurs types de déterminants. On compte tout d'abord, les rapports divers qu'entretiennent entre eux les différents groupes fonctionnels et qui relèvent de relations proies/prédateurs ainsi que d'une logique de compétition pour les ressources. Ces interactions sont aujourd'hui relativement bien connues et modélisées. Cette thèse n'aura donc pas pour objet d'apporter un éclairage nouveau sur ces phénomènes. A ce titre, des modèles existants appartenant au champ de l'océanographie ou au champ des statistiques seront utilisés pour intégrer ces dynamiques internes aux populations phytoplanctoniques.

Au-delà de ces dynamiques internes, les structures des groupes fonctionnels du phytoplancton réagissent à des facteurs exogènes. Au premier rang de ceux-ci, peuvent être retrouvées des variables environnementales comme la température de l'eau de mer, la salinité, la concentration en sels nutritifs et l'irradiance. Ces structures sont également déterminées par la prédation par d'autres espèces (cilliés, flagellés, micro et méso-zooplancton) et les lyses virales (phénomènes moins connus dans le cas des lyses virales, mais non étudiés dans le cadre de cette thèse). La conjonction de ces deux types de facteurs crée des niches écologiques, c'est-à-dire une spécialisation temporaire et spatialement cloisonée des différents groupes fonctionnels au sein de l'écosystème, qui restent cependant extrêmement dynamiques [21].

Les équilibres créés par ces dynamiques sont particulièrement fragiles et peuvent être pertubés par des événements impulsionnels, relativement localisés et courts dans le temps. Ces

derniers sont souvent créés par des variations météorologiques intenses, comme les coups de vent, les orages ou encore des pluies importantes et entrainent des déplacements de masse d'eau relativement importants. Des travaux océanographiques récents ont permis de mettre en lumière que les communautés phytoplanctoniques, les picoeucaryotes dans le cas de Thyssen & al. (2008) [22], ou d'autres groupes fonctionnels comme dans Dugenne & al. (2014) [9] ont des délais de réponse relativement courts (de l'ordre de la demi-journée) à ce genre d'événements intenses. La réponse du phytoplancton à ces événements ne peut donc être observée qu'à l'aide de séries temporelles à haute fréquence (possédant par exemple plusieurs observations par heure) et acquises de manière automatisée.

De plus, de manière intuitive la collecte de données océaniques est plus souvent réalisée lorsque les conditions météorologiques sont clémentes. De ce fait, moins d'observations sur les structures phytoplanctoniques sont disponibles lors d'événements impulsionnels que lorsque les conditions météorologiques sont plus calmes. Les jeux de données résultant sont donc déséquilibrés , ce qui est dû à la collecte des données et non pas à la rareté de ces événements. Ces jeux de données déséquilibrés posent des problèmes statistiques relativement bien documentés qu'il faudra prendre en compte.

Heureusement, certaines séries temporelles acquises par l'Institut Méditerranéen d'Océanographie (M.I.O) présentent à la fois une fréquence relativement élevée (infra-horaire) et comportent un nombre acceptable d'observations obtenues lors d'événements météorologiques intenses tel que le Mistral qui sévit environ un tiers de l'année dans le Sud-Est de la France. La série EOL ou encore la série Berre entrent notamment dans ce cadre. D'autres séries temporelles obtenues dans le cadre du projet d'initiative d'excellence A*MIDEX CHROME, ou lors d'autres missions telles que PEACETIME ou encore OSCAHR pourront être adaptées pour notre cas d'étude. Ce sera également le cas des données issues de l'analyse en continu de l'eau de mer de l'Anse des Cuivres à Endoume depuis la mise en place très récente du laboratoire SSL@MM (Sea Water Sensing Laboratory @ MIO Marseille) qui réalise un prélèvement automatisé d'eau côtière toutes les deux heures. Enfin, des données plus basse fréquence (bimensuelle) mais sur longue durée venant du réseau national des stations marines SOMLIT (depuis 2009) ou du programme national MOOSE (depuis 2011) pourront servir de point de référence aux analyses.

Le but de l'analyse statistique serait donc de caractériser les relations existant dans le temps et l'espace entre les structures phytoplanctoniques et les variables environnementales locales que cela soit lors d'événements impulsionnels ou non.

Plus précisemment, le modèle devra caractérisé la dépendance existant entre la série des distributions multivariées des différents groupes de phytoplancton (qui serait notre Y multivarié de tailles $N \times k$ avec N le nombre d'observations et $k \in [7, 10]$, le nombre de groupes)

et les variables environnementales (elles aussi des séries temporelles incluant la température, la salinité, les sels nutritifs etc...).

Pour ce faire trois grands types d'approches peuvent être conduites : :

— Avoir recours à un modèle "boite noire" essentiellement tourné vers la prédiction de la structure phytoplanctonique
— Inclure de l'aléatoire dans des modèles océanographiques existants
— Un modèle purement statistique

Une approche uniquement centrée sur la prédiction reposerait par exemple sur l'utilisation de Réseau de Neurones Récurrents déjà présentés plus haut. Ces modèles permettraient de prendre en compte la dépendance temporelle et spatiale existant entre les différents prélèvements. Cependant, il semble que la partie interprétabilité du modèle soit une demande forte des océanographes, ce qui rend cette approche peu adaptée. Rien n'empêche néanmoins d'ajuster ce modèle aux données afin d'avoir un contrefactuel en termes de pouvoir prédictif.

La deuxième approche consisterait à inclure des outils statistiques au sein de modèles océanographiques existants. Un des modèles sur lequel l'approche pourrait se baser est celui de Kenitz et Williams (2013) [14]. Ces derniers tentent de rationaliser le "paradoxe du plancton". Introduit en 1961 par Hutchinson [13], il énonce être contre-intuitif l'existence d'autant d'espèces de planctons alors que les ressources sont limitées et l'environnement relativement homogène. On s'attendrait donc en effet à ce qu'une des espèces devienne dominante et créée une compétition excluante [12] qui verrait un faible nombre d'espèces survivre.
Leur modèle montre que ce paradoxe peut-être résolu en considérant que l'effet du vent, des courants ou par exemple d'évènements impulsionnels créés des dynamiques chaotiques qui empêchent le système d'atteindre l'équilibre de long-terme (*i.e.* la compétition excluante). Ces phénomènes physiques ont en retour une grande influence sur l'offre de nutriments disponibles pour les espèces de phytoplancton. Les auteurs montrent alors que plus les espèces phytoplanctoniques diffèrent dans leurs besoins nutritifs plus la diversité en termes d'espèces est un équilibre soutenable.

Ce modèle permet donc de prendre en compte aussi bien les variables environementales physiques (courants, évènements impulsionnels), que les variables environnementales "nutritives" (températures, sels nutritifs etc)... pour expliquer la structure phytoplanctonique. C'est un modèle mécaniste (modélisant des mécanismes physiques existants ou probables) et déductif (partant de théories océanographiques pour en déduire des implications). Les travaux entrepris au cours de la thèse pourraient permettre d'ajouter une composante inductive à cette classe de modèle. Une des pistes intéressante serait de remplacer les équations différentielles d'offres de nutriments par des processus stochastiques comme les modèles ARMA ou

VARMA. L'estimation des paramètres directement à partir des données permettrait d'éviter de simuler plusieurs millions de jeux de paramètres. A l'inverse, l'estimation des paramètres des équations différentielles pourrait se faire à l'aide de méthodes d'Approximate Bayesian Computation (ABC) [3]. Les lois *a priori* permettraient d'inclure les connaissances océanographiques et les lois *a posteriori* de mieux caractériser le lien observé dans les données entre la structure phytoplanctonique et les variables environnementales.

Enfin, la dernière classe de modèles considérée est celle des modèles purement statistiques. L'approche la plus naturelle serait celle de la régression multivariée. On pourrait par exemple considérer les données comme étant des données de panels dont le couple lieu/date ferait office d'identifiant. La composante spatiale pourrait alors être négligée dans un premier temps, ce qui permettrait d'ajuster un modèle VARIMA (Vectorial Auto-Regressive Integrated Moving Average). Un tel modèle renseignerait aussi bien sur le lien structure phytoplanctonique/variables environnementales que sur les liens existants entre les covariables.

Une autre approche consisterait à discrétiser la distribution jointe. Cette discrétisation se ferait grâce à un *clustering* sur les Y_i à l'aide des DGMMs. On aboutirait donc à G groupes homogènes auxquels on donnerait un label g. Ceci permettrait de faire de la classification univariée et simplifierait l'analyse et l'interprétation : On caractériserait le lien entre des structures phytoplanctoniques similaires et les variables environnementales les générant.

4 Conclusion

Le but de ce stage était de pouvoir amorcer les travaux de la thèse et de pouvoir explorer un grand nombre de pistes différentes que cela soit en statistiques ou en océanographie.

Concernant la partie statistique, cela m'a donné l'occasion de prendre du temps pour bien comprendre les raffinements et les possibilités offertes par les Deep Gaussian Mixtures Models. Le développement d'un Variational DGMM permettrait de rendre le modèle plus identifiable et efficace computationnellement.

En ce qui concerne l'océanographie, la mise en place des premiers modèles d' apprentissage statistique ont donné un aperçu de leur capacité à automatiser le traitement des données cytométriques. Certaines pistes se sont montrées fructueuses à l'instar du FFNN pour classer les groupes fonctionnels à partir des valeurs des courbes. A l'inverse, d'autres approches se sont révélées être des impasses comme le fait de recourir à des images de courbes.
Enfin, ces quelques mois m'ont également permis de réfléchir à la modélisation de l'impact temporel et spatial des variables environnementales sur la distribution des groupes fonctionnels de phytoplancton. Une approche purement statistique sera dans un premier temps

menée, avant d'explorer la possibilité d'inclure de l'aléatoire dans les modèles océanographiques existants.

Si le stage a été l'occasion d'explorer beaucoup de possibilités, il est évident que les prochains mois viseront à préciser le projet de thèse et à se concentrer sur quelques directions cohérentes.

Références

[1] Simon Barthelmé, Nicolas Chopin et Vincent Cottet. « Divide and conquer in ABC : Expectation-Propagation algorithms for likelihood-free inference ». In : *Handbook of Approximate Bayesian Computation* (2018), p. 415–34.

[2] L. Beaufort et D. Dollfus. « Automatic recognition of coccoliths by dynamical neural networks ». In : *Marine Micropaleontology* 51.1-2 (2004), p. 57–73.

[3] Mark A Beaumont, Wenyang Zhang et David J Balding. « Approximate Bayesian computation in population genetics ». In : *Genetics* 162.4 (2002), p. 2025–2035.

[4] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006. Chap. 10.

[5] David M Blei, Alp Kucukelbir et Jon D McAuliffe. « Variational inference : A review for statisticians ». In : *Journal of the American Statistical Association* 112.518 (2017), p. 859–877.

[6] Leo Breiman. « Random forests ». In : *Machine learning* 45.1 (2001), p. 5–32.

[7] G. Celeux et J. Diebolt. « The SEM Algorithm : A probabilistic teacher algorithm derived from the EM algorithm for the mixture problem ». In : *Computational Statistics Quarterly* 2 (1985), p. 73–82.

[8] Botond Cseke et Tom Heskes. « Approximate marginals in latent Gaussian models ». In : *Journal of Machine Learning Research* 12.Feb (2011), p. 417–454.

[9] M. Dugenne et al. « Consequence of a sudden wind event on the dynamics of a coastal phytoplankton community : an insight into specific population growth rates using a single cell high frequency approach ». In : *Frontiers in Microbiology* 5 (2014), p. 485. issn : 1664-302X. doi : 10.3389/fmicb.2014.00485. url : https://www.frontiersin.org/article/10.3389/fmicb.2014.00485.

[10] Susanne Dunker. « Hidden Secrets Behind Dots : Improved Phytoplankton Taxonomic Resolution Using High-Throughput Imaging Flow Cytometry ». In : *Cytometry Part A* (2019).

[11] C. Fraley et A.E. Raftery. « Model-based clustering, discriminant analysis, and density estimation ». In : *Journal of the American statistical Association* 97.458 (2002), p. 611–631.

[12] Garrett Hardin. « The competitive exclusion principle ». In : *science* 131.3409 (1960), p. 1292–1297.

[13] G Evelyn Hutchinson. « The paradox of the plankton ». In : *The American Naturalist* 95.882 (1961), p. 137–145.

[14] Kasia Kenitz et al. « The paradox of the plankton : species competition and nutrient feedback sustain phytoplankton diversity ». In : *Marine Ecology Progress Series* 490 (2013), p. 107–119.

[15] Y. Lecun. « Une procédure d'apprentissage pour réseau à seuil asymétrique ». In : *Proceedings of Cognitiva 85, Paris* (1985), p. 599–604.

[16] Anthony Malkassian et al. « Functional analysis and classification of phytoplankton based on data from an automated flow cytometer ». In : *Cytometry part A* 79.4 (2011), p. 263–275.

[17] K.V. Mardia, J.T. Kent et J.M. Bibby. *Multivariate analysis / K.V. Mardia, J.T. Kent, J.M. Bibby*. English. Academic Press London ; New York, 1979, xv, 521 p. : isbn : 0124712509 0124712525. url : `http://www.loc.gov/catdir/toc/els031/79040922.html`.

[18] G.J. McLachlan et D. Peel. *Finite mixture models*. English. Includes bibliographical references (p. 349-393) and indexes. New York : Wiley, 2000. isbn : 0471006262 (cloth : alk. paper). url : `http://public.eblib.com/choice/publicfullrecord.aspx?p=219004`.

[19] F. Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[20] Victor Smetacek, Marina Montresor et Peter Verity. *Marine Productivity : Footprints of the past and steps into the future*. Blackwell Science, Ltd., Oxford, 2002.

[21] T.F. Thingstad, E. Strand et A. Larsen. « Stepwise building of plankton functional type (PFT) models : A feasible route to complex models ? » In : *Progress in Oceanography* 84.1-2 (2010), p. 6–15.

[22] M. Thyssen et al. « Short-term variation of phytoplankton assemblages in Mediterranean coastal waters recorded with an automated submerged flow cytometer ». In : *Journal of Plankton Research* 30.9 (2008), p. 1027–1040.

[23] C. Viroli et G.J. McLachlan. « Deep Gaussian mixture models ». In : *Statistics and Computing* (2017), p. 1–9.

5 Remerciements

Je tiens à remercier tout d'abord Nicolas Chopin qui a accepté de m'encadrer toute cette année en Formation Par la Recherche puis en stage cet été. A votre contact, j'ai découvert de nombreuses notions comme par exemple les champs Gaussiens ou encore les méthodes variationnelles et l'Expectation-Propagation que j'ai pu mettre en application durant ce stage. Vous avez toujours pris le temps de m'aider, jusqu'à refaire les calculs erronés de certains papiers, lorsque j'étais bloqué tout en me laissant libre de travailler de la façon que je jugeais la meilleure. Cela constitue une réelle fierté de vous avoir eu comme professeur et encadrant.

J'aimerais également remercier Denys Pommeret, Melilotus Thyssen, Gérald Grégory et Samuel Soubeyrand qui m'ont tout de suite donné envie de choisir ce sujet pluridisciplinaire. Lors de nos premiers appels, vous avez immédiatement été avenants, compréhensifs et transparents envers moi en mettant à disposition les ressources nécessaires pour que je puisse bien cerner le sujet. Ces premiers mois de travail m'ont réellement enthousiasmé et je suis très content de travailler avec vous pendant les trois années à venir. Plus particulièrement, je voudrais remercier Denys de se rendre disponible malgré ses nombreuses contraintes géographiques et professionnelles. Melilotus, merci beaucoup d'avoir pris tout ce temps pour me présenter le laboratoire et les chercheurs et pour ta patience envers mon absence de culture océanographique... Enfin, merci Samuel et Gérald de prendre part au projet sur votre temps et à distance dans le cas de Samuel.

A Notebooks à rajouter en annexe