
Université Paris 8

BCI : Detection et classification des activités cérébrales

# Rapport I projet tuteuré

*Auteurs :*
Nizar ABAK-KALI
11290569
Alexis
ZURAWSKA
12310497

*Tuteur :*
M. Larbi BOUBCHIR

18 janvier 2016

# Table des matières

# Introduction

Durant notre première année de master en informatique à l'université Paris 8 Vincennes - Saint-Denis, nous devions développer des applications au cours d'un module intitulé "Projets tuteurés". Ce travail serait effectué sur toute l'année et doit faire l'objet d'une présentation en plus de ce rapport-ci. Nous devions choisir une thématique parmi celles proposées. Ensuite, notre tuteur attitré nous attribuait un projet en nous expliquant ce qu'il attendait de nous. En ce qui nous concerne, mon binôme et moi, nous devons développer un réseaux de neurones capables de reconnaître différents états de stimulation d'une personne atteinte de la maladie d'Alzheimer. C'est après vous avoir présenté l'ensemble de nos lectures sur l'utilité et l'implémentation des réseaux de neurones et des interfaces cerveaux-machines que nous vous présenterons l'architecture de notre application, ce après quoi nous évoquerons les résultats que nous attendons de ce projets et les perspectives futures de développement.

# Première partie

# Introduction Théorique

# Chapitre 1

# BCI

BCI : Brain Computer Interface (ou Interface Cerveau-Machine en français).

Aussi appelée Interface Neuronale Directe (abrégée IND), il s'agit d'une interface de communication directe entre un cerveau et un composant externe (généralement un ordinateur). Ce procédé consiste en fait à restaurer, au moins partiellement des facultés perdues, voire non connues (dans le cas d'un handicap de naissance). C'est le cas notamment d'une personne ayant perdu totalement la vue à qui on a implanté un tel procédé au niveau de son cortex visuel, ce qui lui a permis de percevoir de nouveau la lumière. Si ces procédés ne sont pas miraculeux pour le moment, ils peuvent également permettre d'effectuer des actions par l'intermédiaire de la pensée. Ainsi, des personnes ont pu écrire sur un écran d'ordinateur ou déplacer le curseur d'une souris en imaginant simplement l'action de le faire. D'autres ont pu déplacer un bras robotisé pour qu'il leur ramène quelque chose par exemple. Ce procédé peut être unidirectionnel (la machine envoie des données au cerveau ou le contraire mais pas les deux en même temps) ou bidirectionnel. Ces interfaces neuronales directes présentent toutefois plusieurs limites :

- chacune d'elles ne fait qu'une tâche précise et non plusieurs ;
- elles étaient initialement développées dans un but médical rendant l'accès difficile au grand public ;
- à cela s'ajoute le fait que les populations défavorisées n'auront pas les moyens de se procurer de tels équipements, le prix étant très élevé.

# Chapitre 2

# Généralité

Ici, deux concepts clés doivent être définis pour comprendre en quoi consiste notre projet : les réseaux de neurones ainsi que l'Electroencéphalographie (EEG).

D'abord, il est bon de voir un réseau de neurones artificiels comme un modèle de calcul représentant schématiquement les réseaux de neurones biologiques. C'est à celui-ci que l'on va imposer l'apprentissage d'un échantillon de données. Pour cela, nous alllons utiliser l'algorithme du gradient, le gradient étant, en mathématiques, un vecteur qui représente la variation d'une fonction en fonction de la variation de ses paramètres. Tout comme nous, le réseau de neurones n'est pas parfait et fera des erreurs lors de son apprentissage. Il faudra donc utiliser la rétropropagation du gradient qui est un algorithme servant à corriger les erreurs du réseau de neurones pour qu'il ne lest reproduise pas.

Il est également important de définir l'EEG : méthode d'exploration cérébrale mesurant l'activité du cerveau au moyen d'électrodes posés sur le cuir chevelu. On l'appelle aussi électroencéphalogramme et il est comparable à l'électrocardiogramme.

# Chapitre 3

# Contexte

Dans le cadre de ce projet, nous devions développer un réseau de neurones afin de reconnaître les différents états d'une personne atteinte de la maladie d'Alzheimer. Cinq états devaient être détectés :

- lecture ;
- attention sur un film ;
- écriture ;
- discussion ;
- repos.

# Deuxième partie

# Architecture de l'application

# Chapitre 4

# Organigramme

cf Annexes .

# Chapitre 5

# Composant de l'application

cf. Annexes pour les diagrammes de classes.

L'application est composée de cinq packages : eeg, math, neuron, charts et donnees. Nous allons donc décrire le contenu de chacun de ces packages ainsi que le rôle des classes qui les composent :

- eeg : contient la classe EEG_time_signal qui contiendra les fonctions concernant le traitement du signal en fonction des données reçues et traitées ;
- math : contient toutes les fonctions mathématiques avec la classe ComplexNumber permettant de créer et gérer les nombres complexes, la classe ComplexArray qui permet de stocker les valeurs réelles et imaginaires des nombres complexes, la classe Hilbert qui contient les fonctions concernant les espaces d'Hilbert (extension des espaces euclidiens à des dimensions finies quelconques ou infinies) et enfin la classe Fft qui contient les fonctions en rapport avec la Transformée Rapide de Fourrier(ou Fast Fourrier Transform) ;
- neuron : contient la classe CaracteristiqueTemporel qui contient toutes les fonctions ayant attrait aux caractéristiques temporelles d'un neuronne pour le réseau qui sera par la suite créé. Ces caractéristiques temporelles regroupent notamment la moyenne des données, la variance, etc... ;
- charts : contient la classe Graphique afin de générer les graphiques une fois les résultats obtenus à partir des données grâce à la librairie JFreeCharts ;
- donnees : contient la classe RecupFichier permettant de récupérer la base de données pour la stocker en mémoire afin de permettre au programme de travailler avec.

# Chapitre 6

# Explication Technique

## 6.1 Idéés

Nous avons eu l'idée, pour ce projet, de créer un "classifieur" représenté par un réseau de neurones afin d'obtenir des résultats à partir d'un flux de données que nous stockerons en mémoire dans un tableau. Chaque donnée sera représentée par un vecteur. Pour chaque vecteur, chacune de ses dimensions représentera une caractéristique du signal EEG de la personne atteinte que nous chercherons à reconnaître (le tout afin de reconnaître les états tel : lecture, repos, regarder un film, etc...). Nous sommes partis de ces idées là car, lors de nos lectures sur les interfaces cerveau-machine, il s'est avéré que ce mode de traitement du signal est très utilisé .

## 6.2 Outils

Nous avons développé notre application au moyen du langage Java et nous avons utilisé l'IDE Intellij qui est très pratique car très ergonomique et très intuitif. Deplus nous avons utilisé Git comme gestionnaire de version, le code étant hébergé sur les serveurs de Github. De surcroît nous avons utilisé JavaDoc pour générer un documentation du code. Par ailleurs, nous avons inclus la librairie JFreeCharts pour réaliser des graphiques sans les contraintes de la bibliothèque Swing pour créer des interfaces ainsi que l'extension LaTex afin de rédiger plus facilement nos rapports directement au sein de l'IDE. Par ailleurs, nous avons utilisé le logiciel argoUML pour réaliser le diagramme de classes de l'application et OpenOffice Draw pour réaliser l'organigramme.

## 6.3 Algorithmes

Nous allons utiliser l'algorithme des k-means ou k-moyennes afin de réaliser notre application d'analyse des états d'une personne atteinte de la maladie d'Alzheimer. Celui-ci consiste, à partir d'un ensemble de données, un ensemble de paramètres statistiques tels que la moyenne, la variance, le mode, la médiane, etc... Par ailleurs, il sera important de calculer la distance euclidienne entre ces données qui seront représentées par des vecteurs. Ainsi, plus la distance entre ces vecteurs sera petite, plus ces données seront rapprochées afin de répartir en "petits tas" les données qui sont proches. Cela permettra ensuite au réseau de neurones créé d'apprendre de manière supervisée (on donne la réponse en cas d'erreur jusqu'à ce que le réseau de neurones trouve tout seul) l'ensemble des données et de pouvoir les classifier ensuite afin de donner des résultats cohérents sur ces données.

# Troisième partie

# Resultat

# Chapitre 7

# tableau de resultat

Bien que nous ayons commencé les développements au commencement des projets, nous n'avons pas encore de résultat, et cela pour plusieurs raisons :

- d'abord, nous n'avons pas encore la base de données sur laquelle nous devons travailler ;
- ensuite, nous n'avons pas fini le développement de l'application, les fonctions de la classe CaracteristiqueTemporel n'ayant pas été développées notamment. Par ailleurs, certaines classes se verront ajoutées et/ou modifiées (comme la classe recupFichier par exemple, ou encore la classe représentant le réseau de neurones). Pour l'instant, toutes les fonctions mathématiques ont été développées. La classe permettant la récupération du fichier est quasiment terminée et le développement des autres classes restent à venir. Si nous avons rapidement la base de données, nous pensons finir notre application entre mars et avril en travaillant de manière hebdomadaire.

# Conclusion

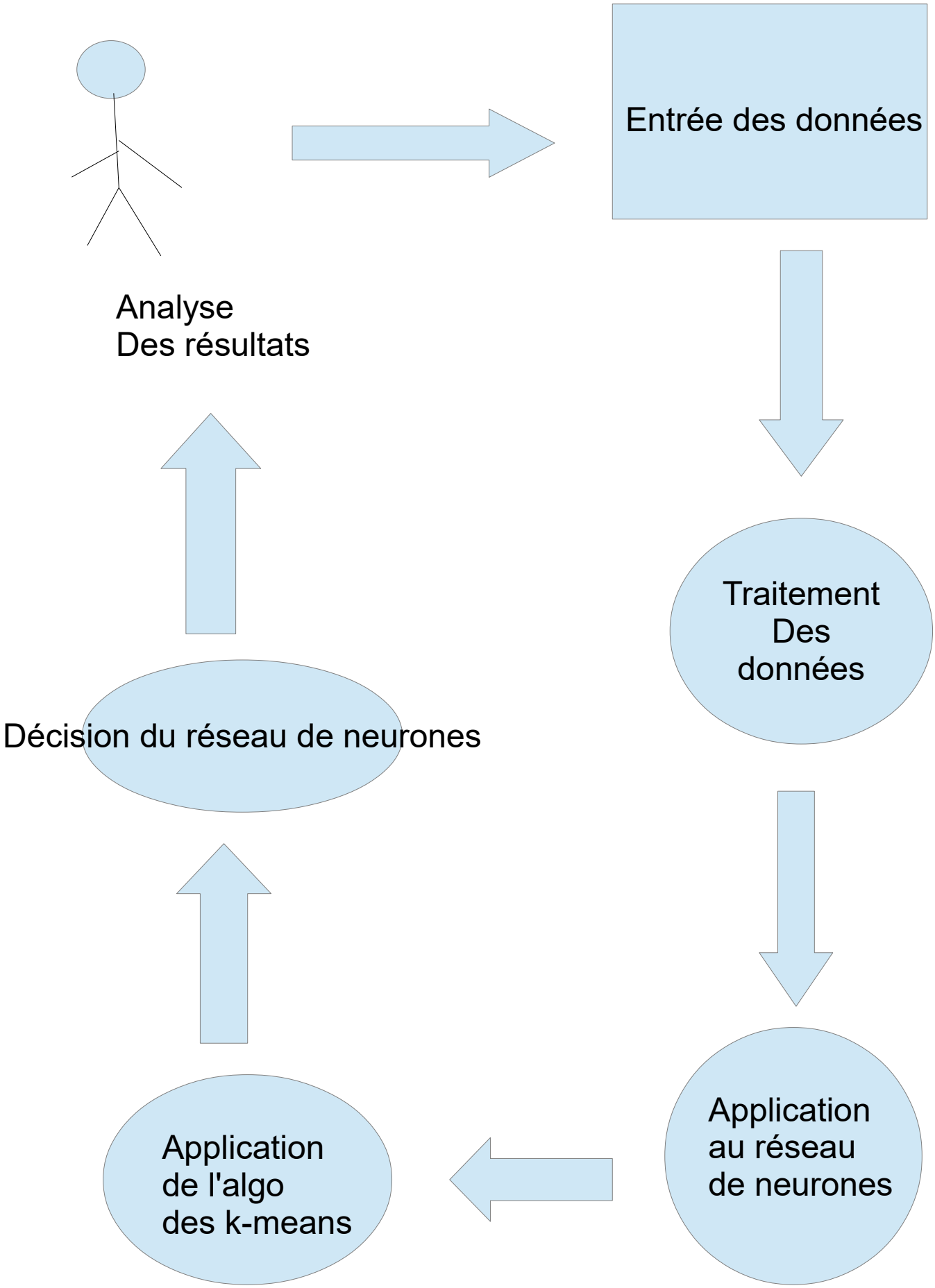
# Quatrième partie

# Annexes

# Chapitre 8

# Organigramme

# Appendices


Entrée des données
Analyse
Des résultats
Traitement
Des
données
Décision du réseau de neurones
Application
au réseau
de neurones
Application
de l'algo
des k-means

# Chapitre 9

# Diagrammes de classes

# Appendices

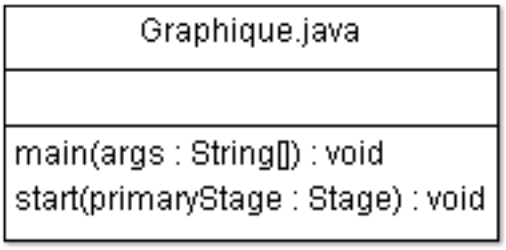



FIGURE 1 – Classe Graphique

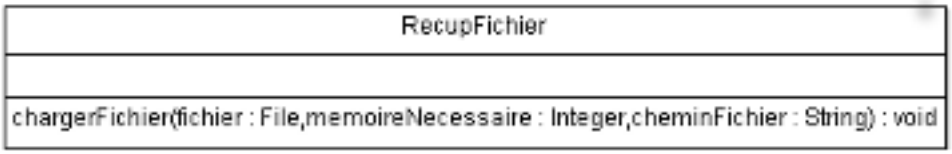



FIGURE 2 – Classe RecupFichier

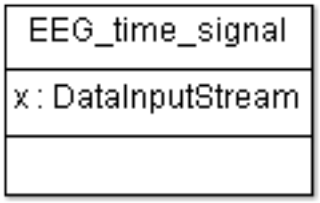



FIGURE 3 – Classe EEG_time_signal

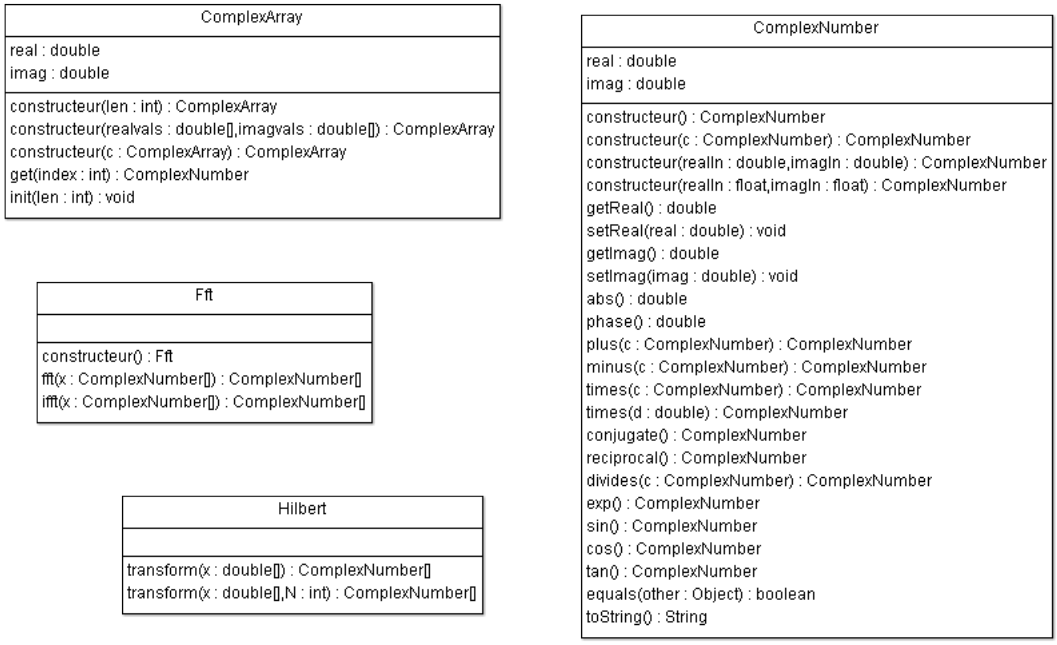



FIGURE 4 – Classes pour les calculs mathématiques

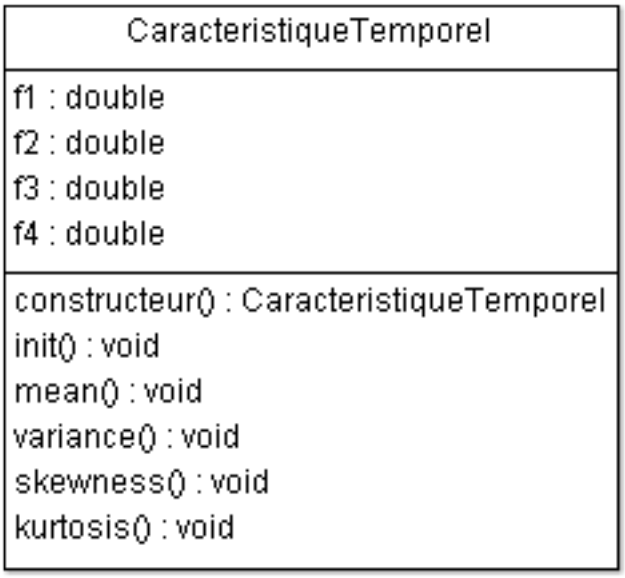



FIGURE 5 – Classe Caractéristique

FIGURE 6 – Packages composant l'application