Université de Bordeaux

Collège Sciences et Technologie

Développement d'un outil de classification automatique de signaux neuronaux biologiques

Cahier des charges

Auteurs:

BLAIS Benjamin

COTTAIS Déborah

DE OLIVEIRA Lila

JOUAN Clément

THOUVENIN Arthur

Bordeaux
10-03-2018
Compte rendu rédigé en LaTeX

Sommaire

1 Introduction

Il existe à ce jour quelques centaines de types de neurones découverts présentant des fonctions bien spécifiques, par exemple les motoneurones, les neurones sensitifs ou encore les neurones bipolaires de la rétine. Cependant, il n'est toujours pas possible d'identifier un type de neurone sans avoir à recourir à des méthodes immunologiques, ce qui implique la plupart du temps de travailler sur des cellules mortes. Les neurones sont classés de différentes façons :

- selon le nombre de neurites sur le neurone, Figure 1 en Annexe (unipolaire, pseudo-unipolaire, bipolaire, multipolaire)
- selon leurs fonctions (neurones sensoriels, motoneurones, interneurones)
- selon leur anatomie (neurones pyramidaux, neurones en étoiles, cellules de Purkinje, etc.)

Toutefois il est possible de prendre de nombreuses mesures sur ces neurones sans avoir recours à des méthodes mortelles, par exemple grâce aux courants électriques et potentiels d'actions. Ce projet a pour but de créer et donc de proposer à un utilisateur un logiciel permettant de classer des types de neurones en fonction de données enregistrées expérimentalement sur des tissus vivants.

Des données ont ainsi été prélevées sur des neurones dans des coupes de tissus nerveux. Elles serviront ainsi à l'élaboration d'un logiciel capable de séparer deux types de neurones (neurones de type I et neurones de type II) en fonction des mesures électrophysiologiques réalisées sur ces derniers. Le type I ainsi que le type II correspondent respectivement à des neurones épineux moyen (Medium spiny neuron) du striatum et des neurones du noyau accumbens.

Le logiciel permettra aux utilisateurs de savoir quels types de neurones ont été analysés, évitant ainsi une charge supplémentaire aux niveaux financier et temporel. De plus, cela limitera l'utilisation de souris transgéniques qui sont toujours difficiles à se procurer et/ou à utiliser dans le domaine de la recherche.

2 Contexte-Analyse des données

Avant la réalisation du logiciel, il est nécessaire d'analyser les données fournies, afin d'établir des stratégies quant aux choix à prendre lors du développement du logiciel.

Afin de programmer le logiciel attendu, il est important de s'approprier les données et d'en comprendre la construction ainsi que le contenu. Cela permet de mettre en place les fonctions nécessaires et de réaliser les calculs utiles au programme. De plus cela permet aussi à l'utilisateur de manipuler ses données via le logiciel.

Une analyse des données, afin de répondre à la question posée, est également nécessaire, afin de mettre en place la méthode de classification qui est utilisée dans le logiciel, il est nécessaire de devoir définir les paramètres ayant un impact significatif sur le type de neurone. Pour cela, diverses analyses doivent être réalisées.

2.1 Analyse des données entrantes ou fichiers

Trois échantillons serviront à l'analyse de données. Les chercheurs en neurosciences ne maîtrisent pas forcément les outils informatiques, c'est pourquoi les données fournies sont souvent sous forme de fichiers *txt*, *csv* ou *xlsx*. Pour les analyses, ce sont des données sous format *txt* qui ont été transmises, elles ont été transformées au format *csv* ou *xlsx* afin d'avoir les données sous la forme tabulaire, afin d'être exploitées et traitées plus simplement.

L'organisation des données se fait comme suit : chaque ligne contient un neurone ainsi que ses caractéristiques définies par 8 paramètres (expliqués dans le Lexique), ceux-ci correpondent à des mesures électrophysiologiques. Les unités sont des variables quantitatives mais ont des unités différentes. Voici un aperçu des données :

Table 1: Aperçu des données (sélectionnées aléatoirement)

nClass	IR	RMP	RH	ST	DTFS	SA	SD	fAHP
1	180.3	-76.32	95	-36.75	357	68.51	3	-0.73
2	67.89	-72.25	205	-40.67	432.7	82.92	2.3	-1.07

Une valeur du paramètre RMP dans l'échantillon 3 a dû être modifiée. En effet, celle-ci était positive alors que les valeurs de ce paramètre doivent être négatives (valeur 69.53). Une erreur a été réalisée lors de la prise des données. L'objectif, durant cette analyse de données, est de déterminer les paramètres permettant la classification des neurones.

2.2 Analyses statistiques

Afin de mieux visualiser les données, il a été décidé de réaliser des boxplots (Figure 3 en Annexe) et d'effectuer des tests statistiques dans le but de savoir si un ou plusieurs paramètres influençent de manière significative le type de neurone.

Ces boxplots ont été réalisés avec Excel pour un meilleur aspect graphique. Ils ont cependant été vérifiés précédement avec des boxplots via le logiciel *Rstudio*, qui est un logiciel plus à même d'analyser des données avec précision.

Pour les tests statistiques, le logiciel *Rstudio* sera utilisé. Une étude est d'abord menée échantillon par échantillon, suivie d'une étude globale des trois échantillons regroupés en un seul et même jeu de données. Celle-ci a été conduite afin d'observer s'il existe une différence significative entre les deux types de neurones selon les paramètres. Deux tests ont été réalisés : un test paramétrique à savoir le test t de Student et le test non paramétrique de Wilcoxon. Il est important de tester préalablement la distribution des données en effectuant un test de Shapiro. En effet, si les données suivent une loi normale alors il faut préférentiellement utiliser le test t de Student qui est un test plus robuste que celui de Wilcoxon. Si la p-value du test est inférieure à 0,05 alors on va refuter l'hypothèse H0 et accepter l'hypothèse alternative H1. Les hypothèses seront :

- Test de Shapiro (H0 : Les données suivent une loi normale)
- Test t de Student et test de Wilcoxon (H0 : Il n'y a pas d'influence du paramètre sur la classe du neurone)

Les résultats concernant les tests de Shapiro, de Student et de Wilcoxon pour les échantillons testés individuellement et testés en étant regroupés sont réprésentés dans les tableaux (Table 2 en Annexe).

2.2.1 Conclusion de l'analyse des paramètres un à un

Au regard des différents boxplots, le paramètre IR (Fig 3b en Annexe) semble être intéressant. En effet, pour chaque échantillon (1, 2, 3) ou pour les trois réunis, les neurones de type I présentent des valeurs légèrement plus élevées que ceux de type II. Cependant cette observation n'est aucunement significative. La même observation peut être faite pour le paramètre RMP, même si la différence est moins marquée. Inversement, pour RH et DTFS les valeurs des neurones de type I sont inférieures à celles des neurones de type II.

Ainsi, à partir des boxplots des différents paramètres, aucun critère ayant un effet significatif sur le type de neurone ne peut être mis en avant. Les paramètres qui seront à surveiller lors de l'analyse combinée seront DTFS et IR.

Les tests statistiques montrent que les paramètres permettant de mettre en évidence une différence selon le type de neurone diffèrent selon l'échantillon. En effet, il est possible d'observer pour l'échantillon 1 (Table 2 en Annexe) une différence significative pour le paramètre ST et DTFS alors que pour l'échantillon 2 (Table 3 en Annexe)la différence significative entre les deux types de neurones s'observent seulement sur ST. Concernant l'échantillon 3 (Table 4 en Annexe), les paramètres qui ressortent sont IR, RMP, RH, SD, fAHP alors que pour l'échantillon total (Table 5 en Annexe), seul le paramètre IR montre une différence significative entre les deux classes de neurones.

Il n'est donc pas possible, à cette étape, de retenir des paramètres spécifiques et convaincant permettant de différencier les neurones. Il serait cependant possible d'exclure SA puisque pour tous les échantillons, ce paramètre ne montre pas de différence significative entre les deux types de neurones. Cependant, il a été décidé de le conserver afin de réaliser une analyse de paramètres combinés. Effectivement, il pourrait se montrer important s'il se voyait combiné à d'autres paramètres.

Cette première analyse ne permet pas d'aboutir à une hypothèse sur le choix de paramètres significatifs ni de conclure sur une classification possible des neurones. C'est pourquoi une analyse plus poussée à partir de paramètres combinés en utilisant le Machine Learning sera réalisée.

3 Etat de l'art

Le sujet étant un sujet de recherche, il est important de s'intéresser à l'avancée des travaux concernant le domaine neurophysiologique. Pour rappel, le but de ce projet est de créer un logiciel qui permettra de distinguer les différents types de neurones grâce à des enregistrements effectués *in vitro*. A ce jour, il est possible d'affirmer qu'il n'existe pas de méthode permettant la classification neuronale grâce à ces données.

3.1 Le Machine Learning et les méthodes de classification supervisée

Il est intéressant de se pencher sur les méthodes de classification utilisées dans d'autres domaines que dans les neurosciences. Un bref aperçu de l'organisation du Machine Learning est offert grâce à la figure 2 en Annexe. En effet, de nombreux systèmes et entreprises utilisent des algorithmes de classification, et, l'un des leaders dans ce domaine est la librairie *scikit-learn*. Cette librairie est actuellement utilisée par bon nombre d'entreprises tel que AirBnB, Booking.com, Spotify ou encore d'autres entreprises qui les financent comme Google. Elle intervient dans de nombreuses problèmatiques impliquant les méthodes de Machine Learning.

Le Machine Learning est une méthode d'apprentissage permettant d'enseigner une méthode à une machine. Cette dernière sera alors capable de reconnaître certains schémas, par l'utilisation de multiples tests statistiques, dans un jeu de données. La machine sera ainsi capable de trier les données grâce à un apprentissage sur ces mêmes données. Il intervient dans de nombreux domaines comme :

- la reconnaissance faciale
- l'autocomplétion
- la reconnaissance d'images
- la recommendation de produits
- la conduite autonome
- la reconnaissance vocale
- et bien d'autres...

Le principe de Machine Learning repose en général sur deux autres notions:

- l'apprentissage supervisé
- l'apprentissage non supervisé

Il existe différents systèmes permettant une classification supervisée comme les arbres de décisions, la régression multiple, la régression logistique ou non supervisée comme le clustering.

L'apprentissage supervisé nécessite un jeu de données dont l'affiliation à chaque échantillon est connue, c'est-à-dire appartenant à tel groupe ou à tel autre. Les données sont pré-étiquetées (comme c'est le cas dans ce projet : un neurone appartient à un des deux types), ce qui permet de contrôler les prévisions données par l'algorithme. Il sera alors en théorie possible de prédire au mieux le type de neurones grâce aux données électrophysiologiques issues de cet apprentissage.

L'apprentissage non supervisé permet de réaliser des classifications ou des séparations "à l'aveugle" en utilisant la plupart du temps les propriétés topologiques des jeux de données.

Il existe actuellement des logiciels permettant une classification supervisée ou non supervisée comme le logiciel libre *MIXMOD*. C'est un logiciel complet qui fournit des réponses correctes concernant la classification en s'appuyant sur différents modèles statistiques (modèles gaussiens, modèles de mélange multinominaux...) dont les calculs sont effectués dans la bibliothèque de calcul *libMixmod* écrite en *C++*. La bibliothèque est accessible sur le logiciel *Rstudio* et sur *Python*. Cependant ce logiciel n'est pas suivi de très près par la commuanauté. Effectivement, celui-ci ne comptabilise que quelques centaines de téléchargements par an, ainsi que des mises à jours trop peu fréquentes.

Il existe également le logiciel *Weka*, ce logiciel propose avant tout une interface graphique, ainsi que le passage en ligne de commande ou non pour l'utilisateur. C'est un logiciel basé sur le langage *Java* et utilisé par de nombreux personnels universitaires. Cependant il n'est pas très bien structuré et documenté, et comme pour MIXMOD celui-ci n'est pas très suivi par la communauté du Machine Learning.

Il a précédement été abordé le cas de la librairie *scikit-learn*, celle-ci se détache nettement des deux logiciels précédement cités. En effet, *scikit-learn* accumule plusieurs point positifs tels que :

- Un bon pré-processing des données
- Une bonne structure et documentation
- Un très bon suivi par la communauté
- Un ensemble de bibliothèques statistiques complètes

En effet cette librairie est Open Source sur *GitHub*, elle fait partie des projets les plus suivis sur cette plateforme, près de 30 000 personnes la suivent et 1000 y ont déjà contribué. Le code est très régulièrement corrigé et de nombreux modules sont ajoutés (quasi-quotidiennement). C'est pourquoi *scikit-learn* sera utilisée afin de développer le logiciel.

En ce qui concerne les méthodes de classification, il est nécessaire de se concentrer sur une méthode supervisée. En effet, les données fournies sont étiquettées,le type de chaque neurone est déjà connu. Cependant il existe de nombreuses méthodes de classification supervisées comme :

- Méthode des moindres carrés
- SVM (Machine à vecteurs de support)
- Régression logistique
- Réseau de neurones
- Méthode de k plus proches voisins
- Arbre de décision
- Classification naïve bayesienne
- Inférence grammaticale
- Espace de versions
- Et bien d'autres...

Les arbres de décision permettent de traiter des variables de différentes natures, de plus cette méthode est insensible aux valeurs extrêmes. Cependant celle-ci est peu robuste, difficile à utiliser dû à sa haute sensibilité. Elle est également très sensible à de légères perturbations dans les données. Or, les travaux se font ici avec des données scientifiques et il peut y avoir des perturbations concernant les données expérimentales. Il serait donc très difficile d'automatiser cette méthode sur les données fournies.

Pour ce qui est du *réseau de neurones*, c'est une méthode très précise. De plus, il est difficile de causer un dysfonctionnement de cette dernière. Cependant ce réseau est très complexe et les paramètres sont très difficiles à interpréter, ce qui n'est pas avantageux pour ce logiciel. En effet le logiciel doit servir à des utilisateurs qui ont peu de connaissances en informatique ou en statistique.

Il ne sera pas décrit ici toutes les méthodes de classification supervisée existantes. Cependant la méthode des *SVM* semble être la plus à même de répondre aux attentes du logiciel. En effet la méthode utilise un système de classification binaire qui peut-être vu comme un inconvénient. Cependant, il n'est pas souhaité que le logiciel puisse classer les neurones en seulement deux types. Il est alors possible d'y voir un avantage étant donné que les algorithmes sont très optimisés, il y aura une meilleure classification des neurones, conduisant alors à des résultats plus robustes.

3.2 L'interface graphique

Il existe de nombreuses interfaces graphiques utilisées en *Python* telles que *Tkinter* (qui est la plus connue), *wxPython*, *PyGTK*, *PyQt*, *PyKDE* ou encore *Kivy*. *Tkinter* possède un avantage, celui d'être inclu directement dans la librairie standard de *Python* et d'être relativement facile d'utilisation, mais cela est aussi un des défauts de cette librairie qui n'est pas très développée. Cependant, la librairie *Kivy* correspond le plus aux attentes du projet. Contrairement aux autres et en particulier à *Tkinter*, elle possède plusieurs widgets et donc des éléments plus visuels pour l'utilisateur, elle est dynamique et versatile et elle n'a pas besoin d'avoir recours à l'utilisation de ligne de commande sur le terminal. Elle est utilisée pour des applications interactives (ce qu'on souhaite pour notre logiciel) et elle peut également être utilisée dans le développement d'applications mobiles. De plus, elle fonctionne sur toutes les plateformes (*Linux*, *Windows*...). Tous ces éléments font que c'est sur *Kivy* que notre choix c'est porté.

4 Analyse des besoins fonctionnels

Il est important de proposer des fonctionnalités à l'utilisateur qui pourra charger son fichier de données concernant les paramètres électrophysiologiques pour chaque neurone afin de classer les neurones en deux types (I et II). Le fichier d'entrée sera donc demandé à l'utilisateur au format *csv*, *xlsx* ou fichier *txt*, afin que ce dernier n'ait pas à changer ultérieurement son fichier de données. Ainsi le fichier de sortie proposé à l'utilisateur sera du type choisi par l'utilisateur. Il pourra également charger un jeu de données d'entraînement pour classer ses données.

Une sauvegarde des résultats en sortie du logiciel sera également mise en place.

De plus, le choix sera laissé à l'utilisateur de sélectionner les différents paramètres de la méthode *SVM*. Il aura donc le choix avec "*RBF*" (Radial Basis Function), "*Polynome*" "*Sigmoid*" et "*Linear*". De plus, les hyper-paramètres pourront être choisis par ce dernier (*gamma*, c, k). Si l'utilisateur de ce programme ne souhaite pas se prononcer sur ces paramètres, ces derniers seront mis par défaut afin de faciliter l'utilisation du programme.

5 Analyse des besoins non fonctionnels

Le programme sera conçu de sorte à ce qu'il puisse être éxecutable sur plusieurs systèmes d'exploitation telles que *Windows*, *Linux* ou encore *Mac OS*. Ceci permettra d'éviter tout conflit à l'utilisateur quant à la plateforme qu'il utilisera. Il faudra donc mettre en place un script capable d'installer le logiciel sur n'importe quelle plateforme.

Une analyse des données sera préalablement réalisée. En effet, cette phase du projet est nécessaire et obligatoire afin d'avoir un rendu exploitable pour la création du programme. Cette analyse passera par l'étude des différents échantillons à disposition.

Afin d'aboutir à un programme fonctionnel répondant aux besoins du client, il sera nécessaire de passer par plusieurs algorithmes constituants le script de notre programme en langage *Python*. En effet ce langage dispose de structures de données de haut niveau et d'une approche de la programmation simple mais efficace. De plus il est adapté à la manipulation de données quantitatives, telles que les données fournies.

Il faudra recourir à des librairies existantes telles que *scikit-learn*, *numpy*, *scipy* ou *matplotlib*. Des libraires supplémentaires sont suceptibles d'être utilisées selon les exigences ou les besoins du client.

Le jeu de données fourni est composé d'éléments déjà classés. Il est ici souhaité de pouvoir classer de nouveaux élements en fonction des paramètres vus précédemment. Il sera alors nécessaire de séparer l'ensemble des données en deux parties différentes : une servira de données d'entrainement tandis que l'autre servira de données test. Il sera alors réalisée une classification s'appuyant sur une méthode probabiliste permettant de séparer les élements selon un nombre de classe défini.

Il existe de nombreuses méthodes de classification supervisée. Les *SVM* (machine à vecteurs de support) semblent être les plus à même de répondre aux attentes fixées, il ne sera cependant pas exclu d'avoir recours à d'autres méthodes de classification supervisées.

5.1 Présentation des *SVM*

Comme il a été vu précédemment, l'étude des paramètres pris séparément ne permet pas d'établir une différenciation des neurones, les combinaisons de paramètres devront donc être étudiés. Pour cela, les *SVM* seront utilisées. Pour cette méthode, un échantillon "test" sera nécessaire ainsi qu'un deuxième permettant l'analyse qui sera l'extrapolation des résultats donnés par le "test". Il a été décidé d'utiliser l'échantillon 1 comme celui d'entraînement et le 2 comme celui d'analyse. Dans un second temps, une autre approche sera mise en place en prenant cette fois-ci le fichier regroupant les trois échantillons dans lequel 70% des résultats seront consacrés à l'entraînement et les 30% restant pour le test T de Student.

Il est important d'obtenir des résultats semblables à l'issu de ces deux tests afin d'obtenir une combinaison possible des paramètres influençant sur le type du neurone.

Les *SVM*, machines à vecteurs de support ou séparateurs à vaste marge (en anglais support vector machine) sont des algorithmes permettant une classification binaire. Cette méthode sera utilisée en premier lieu. Elle se base sur le principe de répartition des données en n dimensions où n correspond au nombre de paramètres en entrée. Ainsi ici, nous aurions un graphique à 8 dimensions, ce qui est impossible à visualiser pour un humain. Afin de mieux comprendre, la figure est observable 5. Sur ces figures, il est possible d'observer que la vue en 2D peut limiter la classification des données en revanche quand il est ajouté une dimension pour arriver en 3D, deux groupes bien distincts se retrouvent visible. C'est ainsi qu'une machine sera capable de séparer des données en fonction de n dimensions.

En second lieu, si le temps restant imparti pour ce projet le permet, une autre méthode sera proposée telle que le perceptron (algorithme reposant sur des réseaux neurones formels) ou une autre méthode de classification supervisée. L'utilisateur aura donc le choix de la méthode concernant le traitement de ses données. De plus, il disposera d'informations concernant les différentes méthodes qui lui seront proposées.

L'optimisation de la méthode conduira à un jeu d'aller-retours entre modélisation et évaluation qui s'effectuent pour obtenir les performances les plus satisfaisantes possibles. Il est même possible dans certains cas de remettre en question certaines hypothèses de départ et de repartir dans une phase d'exploration afin de mieux appréhender les données.

6 Conclusion

Le client cherche à mettre au point une méthode qui permettrait de distinguer deux populations de neurones à partir des paramètres énumérés précédemment. Ainsi, la version finale du logiciel devra présenter à minima une méthode de classification la plus précise possible. Cela permettant, à partir de données électrophysiologiques, une identification des neurones en fonction de leur classe (type I ou II).

Il pourra également, au regard des délais, être fourni une interface graphique afin que l'utilisateur n'ait pas à recourir à des lignes de commande via le terminal. En outre, d'autres méthodes étudiées pourront possiblement être suggérées à l'utilisateur afin de lui laisser le choix quant à l'étude de ses données.

7 Annexes

7.1 Figures et Tableaux

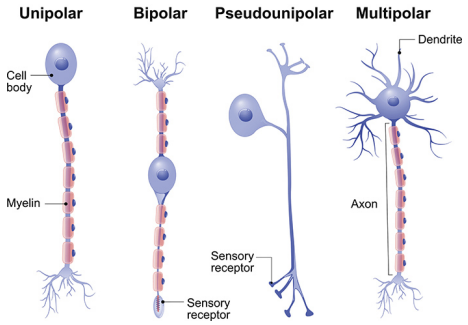


Figure 1: Différents types de neurones

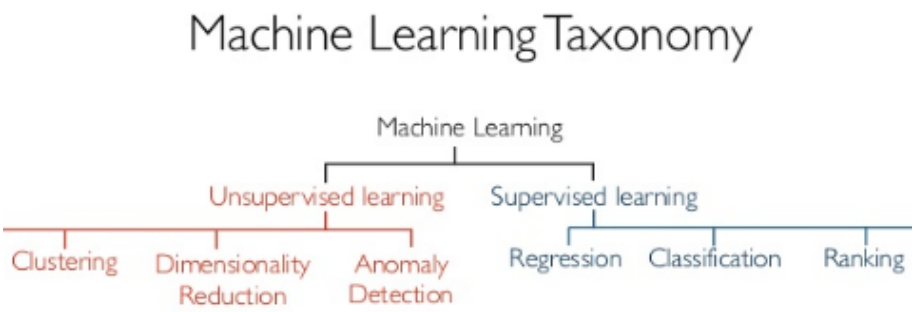
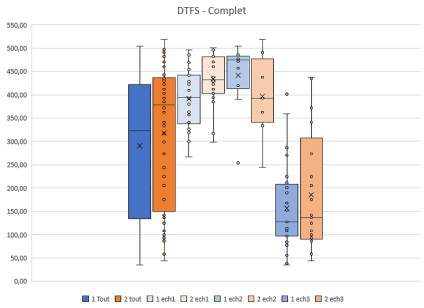


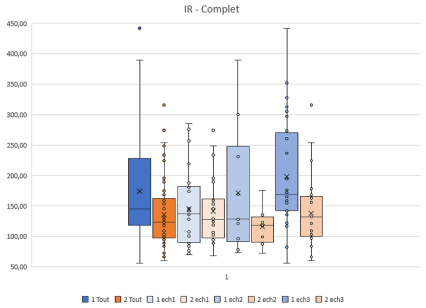
Figure 2: Organisation du Machine Learning

Organisation des boxplots de la figure 3 :

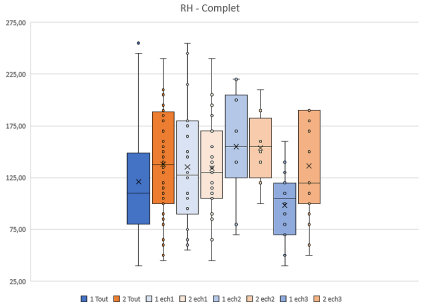
- Première boîte (bleue foncée) = "1 Tout",
  l'ensemble des mesures réunies (3 échantillons) pour le type I
- Seconde boîte (Orange foncée) ="2 Tout",
  l'ensemble des mesures réunies (3 échantillons) pour le type II
- Troisième boîte (bleue très claire) = "1 ech1",
  les données de type I de l'échantillon 1
- Quatrième boîte (orange très claire) = "2 ech1",
  les données de type II de l'échantillon 1
- Cinquième boîte (bleue claire) = "1 ech2",
  les données de type I de l'échantillon 2
- Sixième boîte (orange claire) = "2 ech2",
  les données de type II de l'échantillon 2
- Septième boîte (bleue) = "1 ech3",
  les données de type I de l'échantillon 3
- Huitième boîte (orange) = "2 ech3",
  les données de type II de l'échantillon 3



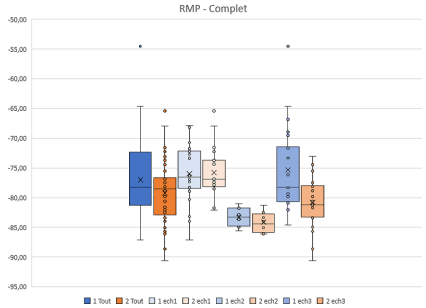
(a) DTFS



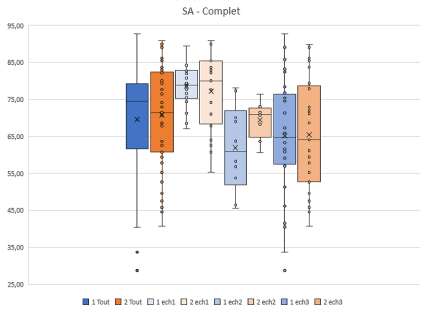
(b) IR



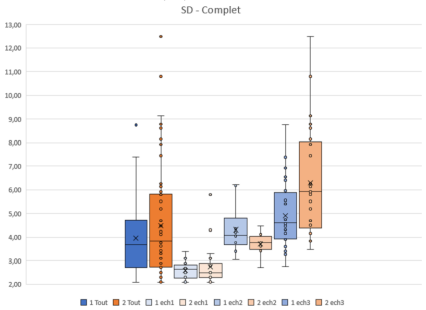
(c) RH



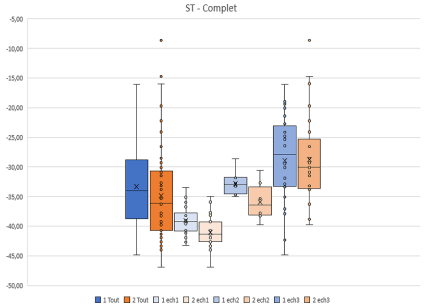
(d) RMP



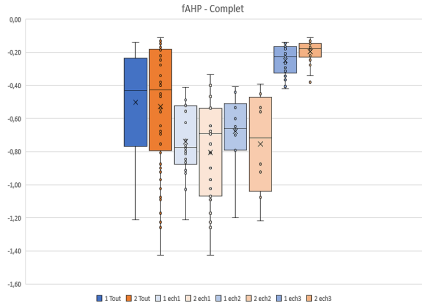
(e) SA



(f) SD



(g) ST



(h) fAHP

Figure 3: Effet du type neuronal et de l'échantillon sur les paramètres du modèle

Paramètres	P-value Shapiro	Test différenciation	P-value
IR	0.0011	Wilcoxon	0.8661
RMP	0.3419	Test t de Student	0.9046
RH	0.2808	Test t de Student	0.9463
ST	0.7046	Test t de Student	0.0128
DTFS	0.3806	Test t de Student	0.0211
SA	0.2160	Test t de Student	0.5709
SD	1.3240e-07	Wilcoxon	0.9005
fAHP	0.5062	Test t de Student	0.3884

Table 2: Echantillon 1

Paramètres	P-value Shapiro	Test différenciation	P-value
IR	0.0339	Wilcoxon	0.3154
RMP	0.3374	Test t de Student	0.3185
RH	0.5262	Test t de student	0.9526
ST	0.2600	Test t de Student	0.0297
DTFS	0.1122	Test t de Student	0.2645
SA	0.8125	Test t de Student	0.0912
SD	0.0393	Wilcoxon	0.2468
fAHP	0.3658	Test t de Student	0.5743

Table 3: Echantillon 2

Paramètres	P-value Shapiro	Test différenciation	P-value
IR	0.0013	Wilcoxon	0.0045
RMP	0.0110	Wilcoxon	0.0038
RH	0.0297	Wilcoxon	0.0104
ST	0.8582	Test t de Student	0.9393
DTFS	0.0006	Wilcoxon	0.6395
SA	0.5074	Test t de Student	0.9602
SD	0.0038	Wilcoxon	0.0304
fAHP	0.0136	Wilcoxon	0.0304

Table 4: Echantillon 3

Paramètres	P-value Shapiro	Test différenciation	P-value
IR	1.3870e-06	Wilcoxon	0.0149
RMP	0.0083	Wilcoxon	0.0650
RH	0.0536	Test t de Student	0.0690
ST	3.5420e-05	Wilcoxon	0.1336
DTFS	8.0090e-07	Wilcoxon	0.2951
SA	0.0007	Wilcoxon	0.7951
SD	6.8520e-08	Wilcoxon	0.5768
fAHP	2.6250e-06	Wilcoxon	0.7972

Table 5: Tous les échantillons réunis

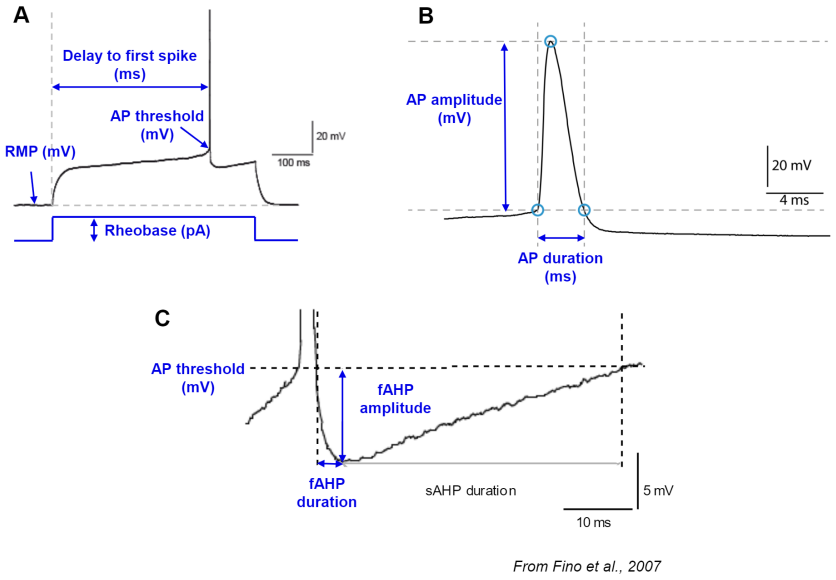


Figure 4: Exemple de relevés des différents paramètres reportés sur des graphiques

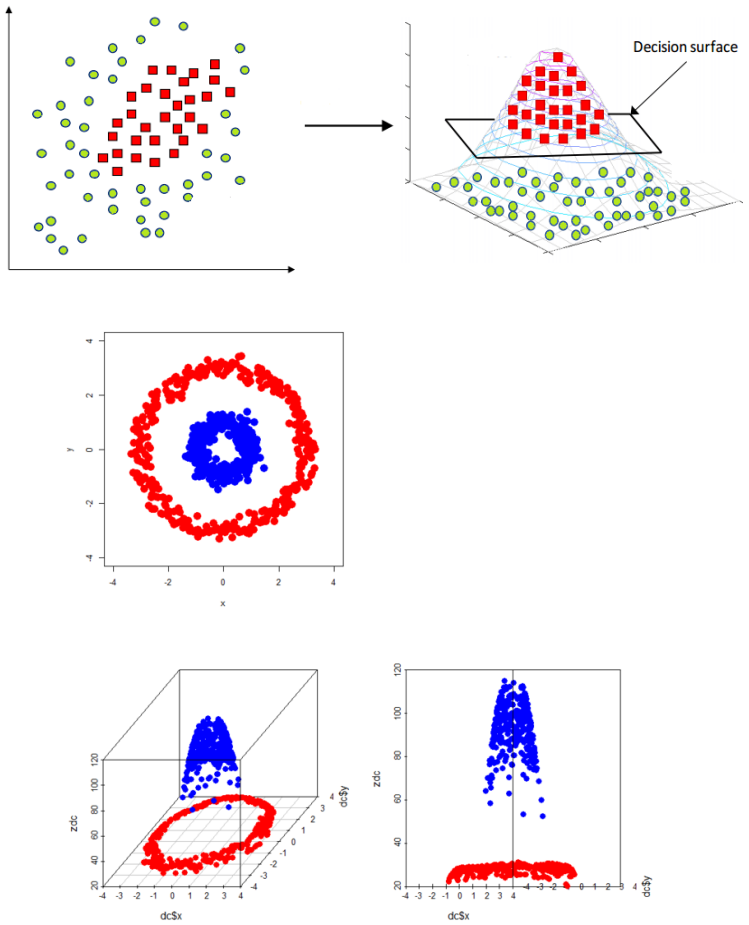


Figure 5: Visualisation de classifications de données via la méthode *SVM*

7.2 Lexique

Description des différents paramètres donnés, leurs unités et acronymes, voir Figure 4 :

- *nClass* : Neuron Class, il existe des centaines de neurones mais ici on utilisera seulement deux types identifiés comme 1 et 2.
- *IR* : Input Resistance (mOhm), cela correpond à une des lois d'Ohm : V = IR où V,I et R correspondent respectivement à une tension, un courant et une résistance.
- *RMP* : Resting Membrane Potential (mV), c'est le potentiel de la membrane étudiée au repos.
- RH : Rheobase (pA), l'intensité minimale de courant excitant qui permet de déclencher un potentiel d'action.
- *ST* : Spike threshold (mV), tension à laquelle le potentiel d'action est déclenché.
- *DTFS* : Delay to first spike (ms), il correspond au délai entre lequel on émet un courant dans un neurone et le moment où l'on observe le premier pic de réaction.
- *SA* : Spike ampli (mV), amplitude du pic de réaction.
- *SD* : Spike duration (ms), durée du pic
- *fAHP* : dV/dt (mV/ms), cela correpond à une variation de tension dans le neurone en fonction du temps.

8 Sources

- https://blogs.scientificamerican.com/brainwaves/know-your-neurons-classifying-the-many-types-of-cells-in-the-neuron-forest/
- https://www.futura-sciences.com/sante/dossiers/medecine-voyage-cerveau-525/page/3/
- https://www.mixmod.org/?lang=fr
- https://www.cs.waikato.ac.nz/ml/weka/
- https://qbi.uq.edu.au/brain/brain-anatomy/types-neurons
- https://www.rstudio.com/
- https://scikit-learn.org/stable/
- https://www.slideshare.net/Phelion/apprentissage-statistique-et-analyse-prdictive-en-python-avec-scikitlearn-par-alexandre-gramfort
- https://github.com/scikit-learn/scikit-learn
- https://www.slant.co/topics/6620/versus/~pyqt_vs_kivy_vs_tkinter
- http://dspace.univ-tlemcen.dz/bitstream/112/322/11/ChapitreII.pdf
- http://www.math.univ-angers.fr/~labatte/enseignement%20UFR/master%20MIM/classificationsupervisee.pdf
- http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
- http://blog.csdn.net/sinat_35257860/article/details/58226823
- http://tjo-en.hatenablog.com/entry/2015/04/20/190000

References

[1] Raúl Garreta, Guillermo Moncecchi. *Machine learning in Python.*. Packt publishing, Novembre 2013, 118.

[2] Gavin Hackeling. *Mastering Machine Learning with scikit-learn*. Packt publishing, Octobre 2014, 238.