# Data mining project

Marvin FOURASTIÉ
Patrick SARDINHA

Juin 2019

## 1 Introduction

Dans le milieu médical, il est nécessaire de détecter les phases de la marche d'une personne (« foot off », « foot strike ») suivant sa pathologie. Les centres médicaux utilisent donc des plaques de pression permettant de détecter les « foot off »et « foot strike », mais cependant pour certains patients, il est impossible d'utiliser cette méthode. Par conséquent les données fournies par les plaques de pression sont incomplètes. Pour palier à ce problème, une détection manuelle peut être effectuée, mais elle demande beaucoup de temps.

Le défi proposé par *SofameHack* est ainsi de développer un algorithme qui se base sur différents capteurs présents sur l'ensemble du corps du patient afin de détecter de manière automatique les évènements liés à la marche. Pour réaliser ce projet, nous avons à disposition des fichiers .c3d annotés et classés par maladie : « cerebral palsy »(CP), « idiopathic toe walker »(ITW) et « feet deformities »(FD). Ces fichiers sont annotés mais ceux-ci sont incomplets, ils ne contiennent que quelques « foot strike »et/ou « foot off ».

Dans une première partie, nous présenterons comment nous avons traité les données et quels sont les outils ainsi que les algorithmes que nous avons testé. Nous verrons par la suite celui que nous avons retenu afin d'automatiser les détections et comment nous avons calculé le score. Puis, dans une deuxième partie nous parlerons des résultats obtenus. Enfin dans une troisième partie, nous analyserons ces résultats et nous discuterons des améliorations à mettre en place.

## 2 Méthode

Au départ, nous nous sommes penchés sur l'extraction des données et comment nous allions les exploiter. Nous avons utilisé le logiciel *Mokka* afin de visualiser les fichiers .c3d dans le but de comprendre les différentes démarches selon les maladies. Ce logiciel permet aussi de voir la trajectoire des capteurs pour en retenir les plus pertinents. Nous avons de plus utilisé la bibliothèque *BTK* afin d'importer les fichiers dans notre environnement *Python*.

Une fois l'environnement mis en place, nous avons testé plusieurs algorithmes tels que « decision tree classifier », « KNN classifier »et « nearest centroid classifier ». Tout d'abord nous avons entraîné ces classifieurs en prenant les 3 capteurs de chaque pied qui sont la pointe du pied (« TOE »), le talon (« HEE ») et la cheville (« ANK »). Afin de définir les frames ne contenant pas d'évènement, nous avons choisi d'annoter « no-event »un intervalle de frames juste avant et juste après un « foot off »ou un « foot strike »déjà marqué dans le fichier. Ceci nous a donc permis d'étudier les premiers résultats renvoyés et ainsi de les analyser manuellement en visualisant les évènements avec *Mokka*. Avec les résultats obtenus, nous avons remarqué que « nearest centroid classifier »nous renvoyait des intervalles assez grands de frames avec comme label « foot off », « foot strike »ou « no-event », ce qui n'était pas le cas des autres algorithmes. Chaque intervalle annoté d'un label contient ainsi un évènement réellement réalisé. C'est ainsi que nous nous sommes basés sur cet algorithme.

« Nearest centroid classifier »est un algorithme de classification qui se base sur le calcul des moyennes des classes. Dans notre cas nous avons trois classes qui sont « foot off », « foot strike »et « no-event ». Lors de la phase de « training », l'algorithme calcule la moyenne de chaque classe. Les instances avec lesquelles nous calculons cette moyenne sont les coordonnées Z de différents capteurs. Ces moyennes sont appelées « centroids ». Ensuite lors de la phase de test, pour classifier chaque instance de test, un calcul de la distance de ce point avec chaque « centroid »est effectué. Cette instance sera classifiée suivant le minimum de cette distance. La distance utilisée est une distance euclidienne.

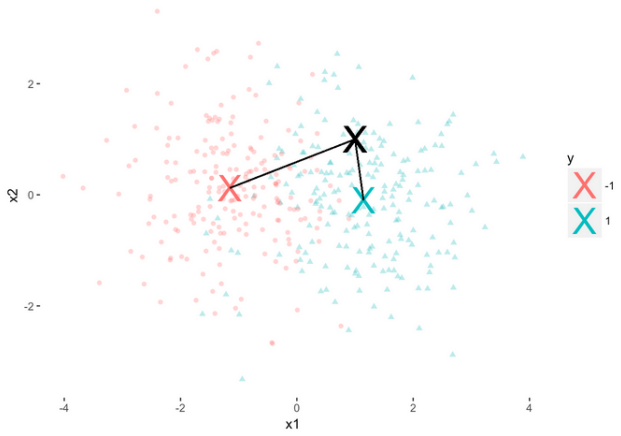


FIGURE 1 – Nearest centroid classifier : exemple pour 2 classes
*https ://idc9.github.io/stor390/notes/classification/classification.htmlnearest_centroid*

Sur la figure 1, les croix bleues et rouges représentent les « centroids »de deux classes, la croix noire indique la position de l'instance de test. Les droites noires représentent les distances calculées entre les « centroids »et le point de test, ainsi seul le minimum de cette distance est retenu et la croix noire appartiendra donc à la classe bleue.

« Nearest centroid classifier »permet de nous renvoyer, comme expliqué dans l'introduction, des intervalles de frames assez larges pour chaque label. L'idée est donc pour chaque intervalle, de déduire exactement à quelle frame l'évènement se produit. Nous avions déduit par nos observations que les « foot off »étaient très souvent placés au début des intervalles associés mais qu'au contraire les « foot strike »étaient eux plus placés sur la fin de nos intervalles. Cependant, il est possible avec cette technique d'obtenir des mêmes labels consécutivement. L'idée a donc été d'insérer dans ce cas le label manquant, en calculant pour le patient en question, des moyennes concernant sa marche.

Pour tester notre modèle, nous avons utilisé la « cross-validation ». Dans un premier temps, nous avons divisé les fichiers en deux parties, la première représente les fichiers de « testing »(environ $\frac{1}{3}$ des fichiers), et la seconde représente les fichiers de « training »(environ $\frac{2}{3}$ des fichiers). Nous avons ensuite appliqué la « cross-validation »en intervertissant les données de « testing »et de « training »(3 permutations). Et nous obtenons ainsi le score en moyennant les résultats.

Pour calculer le score, nous avons utilisé la formule suivante : pour chaque fichier proposé, on calcule la distance en frames de nos labels prédits avec les labels réellement annotés. Puis, nous mettons à l'exponentiel toutes les différences des distances obtenues et nous en faisons la somme : $score = \sum e^{dist}$.

# 3 Résultats

Il est tout d'abord nécessaire de préciser que tout au long du projet, nous avons entraîné et testé notre modèle en séparant les différentes maladies présentes chez les patients, c'est-à-dire que nous avons séparé les « CP », des « FD », des « ITW »(nous avons donc effectué une « cross-validation », comme expliqué dans la partie 1, pour chacune des maladies).

Pour les patients étant atteints de « cerbral palsis »nous avons décidé de sélectionner les capteurs suivants :

- La pointe du pied (« TOE »).
- Le talon (« HEE »).
- La cheville (« ANK »).
- La différence sur la coordonnée Z entre les pointes des pieds.

Nous avons choisi ces capteurs, puisqu'en étudiant les courbes sur *Mokka*, nous les avons trouvés intéressants. Nous obtenons les scores suivants :

| | permutation 1 | permutation 2 | permutation 3 |
|---|---|---|---|
| score foot off | 2.35e+17 | 1.21e+06 | 4.76e+05 |
| score foot strike | 5.40e+12 | 1.34e+09 | 7.89e+04 |
| score global | 2.35e+17 | 1.34e+09 | 5.55e+05 |

Le score global est donc de 7.85e+16.

En ce qui concerne la pathologie de « feet deformities »nous avons choisi les capteurs suivants :

- La pointe du pied (« TOE »).
- Le talon (« HEE »).
- La cheville (« ANK »).

Pour cette maladie nous avons pris uniquement les capteurs des pieds, puisque tous les patients marchent approximativement de la même façon. Nous obtenons ainsi les scores suivants :

| | permutation 1 | permutation 2 | permutation 3 |
|---|---|---|---|
| score foot off | 8.99e+06 | 2.07e+02 | 6.01e+04 |
| score foot strike | 3.44e+03 | 2.17e+02 | 2.38e+03 |
| score global | 9.00e+06 | 4.24e+02 | 6.25e+04 |

Le score global est donc de 3.02e+06.

Pour les malades atteints de « idiopathic toe walker »nous avons pris les capteurs suivants :

- La pointe du pied (« TOE »).
- Le talon (« HEE »).
- La cuisse (« THI »).
- Le tibia (« TIB »).
- La différence sur la coordonnée Z entre les pointes des pieds.

Nous avons ici sélectionné plus de capteurs afin d'essayer d'améliorer le score. Nous avons les résultats suivants :

| | permutation 1 | permutation 2 | permutation 3 |
|---|---|---|---|
| score foot off | 8.63e+15 | 3.67e+19 | 7.97e+20 |
| score foot strike | 4.20e+25 | 2.01e+18 | 1.30e+20 |
| score global | 4.20e+25 | 3.87e+19 | 9.27e+20 |

Le score global est donc de 1.40e+25.

# 4 Discussion

En ce qui concerne les scores obtenus, nous pouvons constater que pour la maladie « CP », ils sont assez élevés (en moyenne 7.85e+16). Cependant, en étudiant les résultats donnés par la validation croisée, nous pouvons voir qu'un seul des cas donne un résultat élevé (2.35e+17), ce qui biaise la moyenne car les deux autres résultats sont de l'ordre de $10^6$. Notre méthode d'analyse détecte ainsi de manière correct les évènements pour la plupart des patients, cependant il existe certains cas trop différents que ce qui a été appris par l'algorithme.

Pour la pathologie « FD », nous pouvons observer que les scores obtenus sont corrects. En effet, en moyenne nous avons 3.02e+06 comme score. Nous pouvons remarquer de plus que pour certaines permutations de la validation croisée, les scores sont de l'ordre de $10^2$. Nous pouvons ainsi en déduire que notre algorithme est plutôt efficace dans l'analyse des démarches pour cette pathologie.

Pour le dernier cas, la pathologie « ITW », nous constatons des scores assez élevés, ils sont en moyenne de 1.40e+25. L'ensemble de « training »prend en compte les capteurs sur les deux pieds, or nous avons pu remarquer à l'aide de *Mokka*, que les patients atteints par cette maladie présentent une façon de marcher différente entre les deux pieds. Cette hypothèse peut expliquer les résultats élevés obtenus.

Pour chacune des maladies étudiées, nous avons décidé de sélectionner différents capteurs. En effet, en faisant des tests, nous avons pu observer plusieurs résultats et nous avons donc pris les meilleurs. Cependant nous aurions voulu faire plus de tests et diversifier les capteurs ainsi qu'appliquer des opérations sur les coordonnées des capteurs. Il est ainsi possible que les résultats présentés ici ne soient pas optimaux.

De plus, notre algorithme est sensible à la régularité de la marche des patients. En effet, nous sommes parti du principe qu'une personne étant atteinte d'une maladie ou non présente une démarche régulière et périodique. Or il existe des cas où ce principe ne s'applique pas, ce qui, de ce fait, induit en erreur l'algorithme lors des estimations des évènements non prédits.

Enfin pour obtenir des résultats plus satisfaisants, nous pourrions augmenter le nombre de fichiers pour avoir un « training »plus efficace. Il serait aussi intéressant de classer les fichiers de façon plus spécialisée.

# 5 Conclusion

Dans cette étude, après avoir testé différents algorithmes de « machine learning », nous avons retenu « nearest centroid classifier »afin d'étudier automatiquement la démarche de patients présentant différentes maladies. Nous avons donc entraîné notre algorithme suivant différents capteurs, ce qui nous a permis de calculer des scores qui représentent les erreurs de prédictions. Les scores que nous avons obtenus sont plus ou moins bons. Afin de les améliorer nous nous sommes penchés sur certains points qui sont d'augmenter le nombre de patients étudiés, de sélectionner des capteurs plus pertinents ou encore d'améliorer nos prédictions par exemple.