SofameHACK

Data Mining – Printemps 2019

Adrien Chabert & Tommaso Peletta

Introduction

SofameHack est un concours organisé par Sofamea, qui est la Société Francophone d'Analyse du Mouvement chez l'Enfant et l'Adulte, en collaboration avec l'HUG. Ce concours consiste à déterminer des évènements spécifiques pendant la marche de personne qui sont sujettes à des pathologies qui impactent leur déplacement. Les évènements à déterminer sont le moment où ils posent le pied et le moment où ils le lèvent. Tout ceci est fait dans le but d'aider le médecin à comprendre les pathologies qui affectent le patient et d'améliorer sa situation.

Pour déterminer ces événements, nous allons utiliser des algorithmes d'apprentissage sur les données qui nous ont été fourni. Notre système devra fonctionner pour différents types de pathologies.

Présentation des données

Le déroulement des événements de la marche d'un patient suit toujours le même modèle. Si on commence par un lever de pied gauche, ensuite on a forcément un posé du pied gauche, suivi par un levé de pied droite et termine par un posé du pied droite (figure 1).

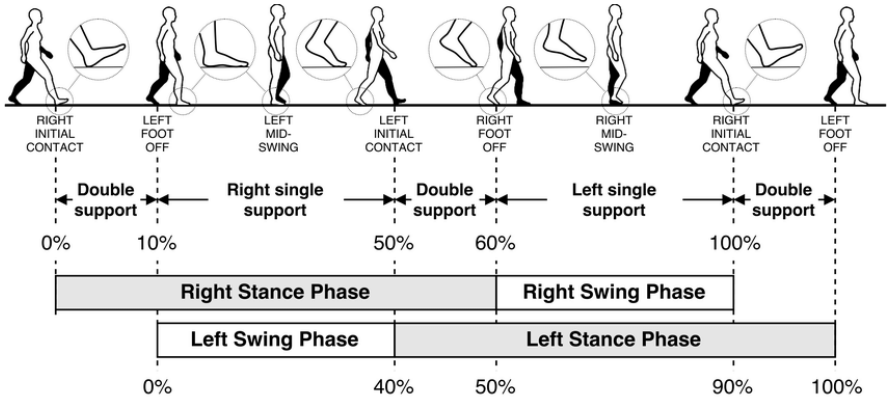


Figure 1 : Déroulement d'une marche chez une personne.

Description des types de pathologies

Trois types de pathologies intéressent notre travail. La première concerne les personnes souffrant d'une infirmité motrice cérébrale, noté par l'acronyme « CP » (Cerebral Palsy). Ceci regroupe diverses atteintes neurologiques qui causent plusieurs types de troubles moteurs. La deuxième concerne les personnes avec des déformations au pied (Cavus Foot, Equinus Foot, Flat Foot, Club Foot et Varus Foot), abrévié « FD » (Foot deformity). La troisième et dernière concerne les personnes qui ont tendance à marcher sur la pointe des pieds, noté « ITW » (Idiopathic Toe-Walker).

Format des fichiers et les capteurs

Un dossier de données a été fourni pour chaque pathologie. 45 fichiers ont été fourni pour CP, 23 pour FD et 20 pour ITW. Chaque fichier est nommé selon le format suivant : *Pathologie_idDuPatient_DateDeVisite_identifiantDuPassage.c3d*. Les fichiers sont écrits avec

le format c3d. C'est un type de fichier largement utilisé dans le domaine du biomécanique pour des données synchronisées en 3 dimensions.

Jusqu'à 35 capteurs ont été placé sur le corps de patients pendant qu'ils marchaient. Chaque fichier contient la position 3 dimensionnel de chaque capteur à tout moment de la marche. 20 capteurs sont toujours présents sur le patient et 15 sont optionnels.

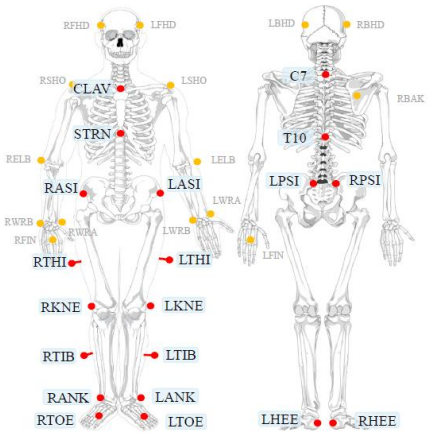


Figure 2 : Position des 35 capteurs sur le corps. En rouge les capteurs permanents et en jaune les 15 capteurs optionnels.

Certains évènements sont déjà annotés sur les fichiers. Ces évènements ont été détecté à l'aide d'une plaque de force disposé dans le sol. La taille de cette plaque est limitée. Malheureusement l'évènement est détecté que si le pied se trouve complètement à l'intérieur de cette plaque de force. C'est pourquoi les fichiers sont incomplètement annotés. Ainsi le but de notre projet est d'annoter les évènements avec plus de précision que la plaque de force.

Méthodologie

Afin de résoudre notre problème, nous allons utiliser des algorithmes de Machine Learning qui nécessite des données d'apprentissage. Durant ce chapitre, on va décrire les sujets suivants :

- Les capteurs et données utilisés pour l'apprentissage
- Les algorithmes d'apprentissage testé
- Le traitement et la vérification de nos résultats

Sélection des données

Comme décrit précédemment, le but de ce projet est d'indiquer des évènements tels que le levé de pied et le posé de pied. Les évènements peuvent être classifiés selon 3 catégories : le levé de pied (FO), le posé de pied (FS) et aucun évènement (NoEvent). La catégorie NoEvent correspond au moment où il n'y a ni un FO, ni un FS. Ainsi ceci correspond à la majeure partie de la marche.

Afin de pouvoir entrainer nos algorithmes de classification, on a besoin d'avoir des données sur des évènements FO, FS et NoEvent. Les données sur les FO et les FS sont récupérées quand les fichiers sont annotés. A chaque évènement annoté FO ou FS, deux évènements NoEvent sont créés à +/- k. La valeur k correspond au nombre de frame séparant un NoEvent d'un FO ou FS.

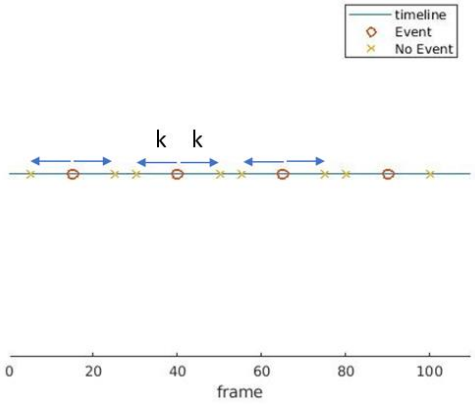


Figure 3 : Représentation de la récupération des évènements

Nous n'avons pas utilisé tous les capteurs pour classifier les données car certains capteurs sont indépendants du mouvement influé par la marche (par exemple la tête). De plus, nous avons exclu l'utilisation des capteurs optionnels. Ils ont été choisis selon une démarche logique et de façon à avoir la plus petite erreur de classification. Les capteurs utilisés sont les suivants :

- Le talon (HEE)
- La pointes des pieds (TOE)
- Le genoux (KNE)

Etant donnée que l'humain possède deux jambes, cela nous fait un total de 6 capteurs. Nous avons utilisé 4 attributs pour l'apprentissage, il s'agit de :

1. La différence entre la position sur l'axe z de TOE et la moyenne de TOE sur l'axe z
2. La différence entre TOE et HEE du même pied sur l'axe x
3. La différence entre TOE et KNE du même pied sur l'axe x

Les évènements du pied gauche peuvent servir à l'apprentissage du pied droit et vice versa.

Afin de séparer les données de test et les données d'apprentissage, on utilise un système de validation croisée où à chaque fois on prend un tier des données pour le test et deux tiers pour l'apprentissage sur une pathologie. C'est important de noter qu'on ne met jamais un même patient dans les données de test et dans les données d'apprentissage.

Algorithme d'apprentissage

Pour la construction de notre algorithme nous avons testé les algorithmes d'apprentissage suivants : Decision Tree Classifier, K-Nearest Neighbors et Logistic Classifier. Les algorithmes de classification sont des algorithmes supervisés avec un résultat discret.

Decision Tree Classifier ou arbre de décission crée un arbre de classification. Chaque nœud de l'arbre correspond à un attribut qui est choisit en fonction de sa probabilité d'erreur. C'est un algorithme classique d'apprentissage automatique.

K-Nearest Neighbors regroupe les k plus proches voisins (distance euclidienne) des données d'apprentissage et pour chaque voisinage défini un label. Ce label est choisi par rapport au label qui est le plus représenté dans le voisinage. Aussi pour chaque voisinage est défini un centroïde. Lors de la classification d'une donnée, on cherche le plus proche centroïde de cette dernière et ainsi on détermine son label.

Une régression logistique est un modèle de régression linéaire qui utilise une fonction de couple complexe. En particulier, à la place d'utiliser une fonction linéaire, on utilise une fonction sigmoïde. L'idée est de la fonction logistique est de limiter la fonction de coût entre 0 et 1, ce qui nous permet de classifier toutes les valeurs.

Traitement des résultats

Après avoir entraîné notre modèle avec les données d'apprentissage, on peut prédire nos résultats sur les données de test chaque pied indépendamment. On attribut un évènement (soit FO, soit FS, soit NoEvent) à chaque frame des fichiers tests. On obtient ainsi une liste des labels correspondant à chaque frame et on voit qu'il y a deux problèmes à résoudre :

1. Autour du vrai évènement (On considère plus les NoEvent), l'algorithme renvoi plusieurs annotation correspond à l'évènement réel. Il y a des regroupements des mêmes évènements dans notre liste.
2. Certains évènements ne sont pas prédits ce qui crée des regroupements sur une large plage de frame.

Pour régler le premier problème, on prend la moyenne tronquée du regroupement d'évènement prédit. Ceci nous permet d'approximer l'évènement réel.

Pour régler le deuxième problème, on calcule la moyenne des deux évènements qui entoure l'évènement manquant en utilisant k-means. Ensuite on ajoute l'évènement manquant au milieu des deux évènements qui l'entoure.

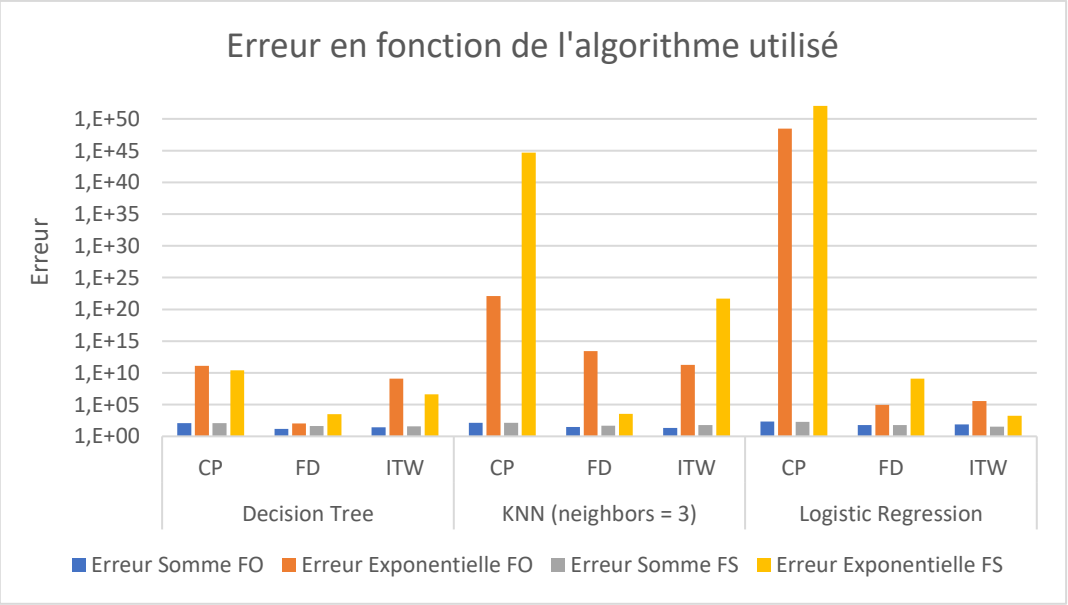
Pour vérifier la qualité de prédiction, nous utilisons deux erreurs différentes. La première est la somme des différences entre le frame prédit d'un certain évènement (FO et FS) et le vrai évènement annoté sur les fichiers testés. La deuxième erreur utilise la somme des

exponentielles des différences. L'erreur final est la moyenne des erreurs de la validation croisée.

Résultats

En exécutant l'algorithme pour différents classifiers et les différentes pathologies, on obtient les résultats suivants :

		k	Erreur Somme FO	Erreur Exponentielle FO	Erreur Somme FS	Erreur Exponentielle FS
Decision Tree	CP	[0,9]	1,25E+02	1,34E+11	1,18E+02	2,42E+10
	FD	[0,9]	1,46E+01	1,13E+02	4,00E+01	3,20E+03
	ITW	[0,9]	2,70E+01	1,20E+09	3,83E+01	4,10E+06
KNN (neighbors = 3)	CP	[0,9]	1,42E+02	1,28E+22	1,35E+02	4,89E+44
	FD	[0,9]	2,93E+01	2,63E+13	4,50E+01	3,57E+03
	ITW	[0,6]	2,23E+01	1,77E+11	5,73E+01	4,70E+21
Logistic Regression	CP	[0,9]	2,05E+02	2,96E+48	1,96E+02	1,18E+52
	FD	[0,9]	5,77E+01	8,56E+04	5,67E+01	1,19E+09
	ITW	[0,6]	7,30E+01	3,71E+05	3,30E+01	1,72E+03



On peut voir que Decision Tree Classifier obtient en générale les meilleurs résultats. En particulier, les pathologies CP et FD sont mieux classifier par Decision Tree Classifier et ITW est mieux classifier de peu par la régression logistique.

La somme des erreurs est en générale assez petite, alors que l'erreur exponentielle est parfois très élevée. Ceci signifie que parfois un évènement n'était pas bien détecté ou complètement ignoré. Plus on ajoute de NoEvent dans l'apprentissage, plus l'algorithme à tendance à ignorer des FO et FS.

Selon les résultats, on voit que la pathologie CP est la plus difficile à prédire, probablement dû à une hétérogénéité des patients. Alors que ITW et FD sont plus homogènes, ce qui nous permet d'avoir une meilleure précision des évènements.

Conclusion

Notre but était de déterminer des évènements lors de la marche de patient. Après avoir tester plusieurs algorithmes d'apprentissage, on peut conclure que le meilleure que le problème est très complexe dû à la grande diversité des patients. En général, Decision Tree est le meilleur classifier affecté par FD. Les meilleurs résultats pour ITW sont obtenus avec la régression logistique. On n'arrive pas à obtenir d'aussi précis résultats pour CP, le meilleur classifier étant Decision Tree.

Enfin, afin d'obtenir de meilleurs résultats, il aurait fallu avoir plus de donner préalablement annoter.