RAPPORT FINAL

INF14 : Prédiction de la glycémie du diabétique à partir de ses activités

30 avril 2018

Antonin Boniteau, Noranne Gabouge, Ayman Idrissi Kaïtouni, Arthur Loison, Théo Mondou, Alexis Prodel, Nicolas Zucchet

ÉCOLE POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

TABLE DES MATIÈRES

INTRODUCTION

Le **diabète** est un problème de santé publique épineux qui n'a eu de cesse de prendre de l'ampleur durant ces dernières années. Si l'Organisation Mondiale de la Santé (OMS) estimait qu'en 1980 quelque 108 millions de personnes en étaient atteintes, elle a malheureusement dû constater une **croissance démesurée**, avec plus de 422 millions de diabétiques en 2016. Plusieurs millions de décès trouvent leur origine dans cette maladie, et ce chiffre ne cessera d'augmenter dans les futures années.

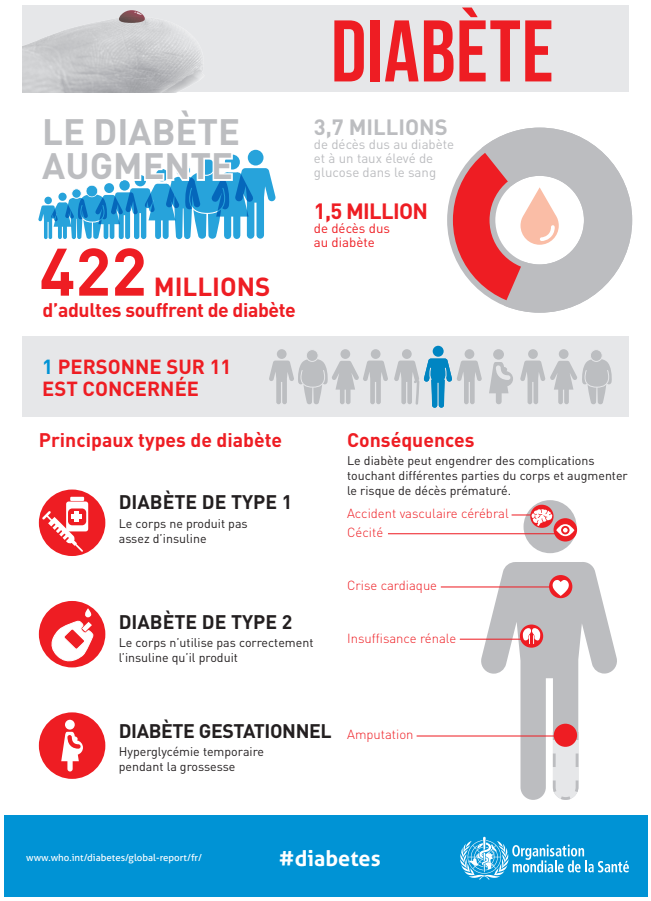

Figure 1 – Journée mondiale de la santé, 7 avril 2016, OMS

Thérapeutiquement, cette maladie est difficile à appréhender. On en recense plusieurs formes, le diabète de type 1 et le diabète de type 2, dont on ne sait actuellement pas traiter les causes mais seulement les symptômes. Dans le premier cas, le diabète est causé par l'absence de cellules produisant l'insuline permettant l'assimilation du glucose et baissant le taux de glucose dans le sang. Dans le deuxième cas, une résistance progressive des cellules à l'insuline rend cette dernière inefficace. Cela pose un véritable problème médical, car une dérégulation de la glycémie peut entraîner une hypoglycémie (évanouissements, troubles moteurs, convulsions, etc.) ou encore une hyperglycémie (dégradation lente des vaisseaux sanguins et nerfs, cécité, infarctus etc.). Ces mécanismes complexes rendent la plupart des traitements intrinsèquement imparfaits : chez les diabètes de type 2, il faut relever la glycémie et changer son régime pour d'une part réguler la glycémie et d'autre part combattre le surpoids qui est souvent l'un des facteurs aggravants ; chez les diabètes de type 1, il faut recourir à de nombreuses injections d'**insuline** pour maintenir le taux de glucose à des niveaux raisonnables. Chez les diabétiques de type 2, seul le régime peut donc influer.

Les diabétiques de type 1 doivent ainsi régulièrement s'injecter, mais surtout **doser** leur insuline en fonction de leur glycémie. Or, la prédiction de la glycémie est complexe à de nombreux égards. Plurifactorielle, cette prédiction doit prendre en compte de nombreux paramètres, comme la nourriture consommée, les activités réalisées ou prévues. Le constat

actuel est que de nombreux diabétiques, par souci de simplicité dans cette prise en charge lourde, choisissent souvent de se mettre en hyperglycémie plutôt qu'en hypoglycémie, car cette dernière peut être fatale à court terme, alors que l'hyperglycémie l'est à long terme.

De nombreux acteurs médicaux sont très présents dans ce domaine, comme le laboratoire Novo Nordisk ou encore Arkray. La start-up Healsy avec laquelle nous avons travaillé au long de ce projet est quant à elle la première entreprise à fournir un système prédictif de la glycémie, grâce à une application conjuguée à un capteur afin de surveiller en continu et d'ajuster la glycémie du patient.

S'il existe actuellement des modèles prédictifs de glycémie, ces algorithmes quantitativement complexes ne prennent pas en compte un paramètre primordial cependant, à savoir le métabolisme. Non seulement ce métabolisme est la clef de la difficulté de prise en charge, nécessitant ainsi du « sur mesure », mais il engendre une complexification stupéfiante : une séance de sport sur un individu donné aura des conséquences évidentes et réelles sur la glycémie et donc sur le bien-être du patient. Afin d'avoir une prise en charge thérapeutique optimale, il s'agit donc de mettre au point un processus simple et rapide permettant de prendre en compte le facteur métabolique individuel dans le traitement du diabète. Ce facteur métabolique passe notamment par les activités qu'effectue le patient.

L'une des pistes qu'Healsy souhaite explorer est donc la prise en compte de l'activité dans leur modèle, afin d'améliorer la précision de leurs outils. L'enjeu pour notre PSC est par conséquent de développer un algorithme d'apprentissage qui permette de déterminer l'activité actuellement suivie par le patient. Les objectifs de ce projet s'inscrivent donc dans la *Human Activity Recognition* (HAR).

I
OBJECTIFS DU PROJET

I.1 Moyens mis en œuvre par Healsy

Afin de pouvoir mener à bien ce projet, Healsy a mis à notre disposition, par le biais de notre tuteur Nicolas Caleca, plusieurs outils :

— des montres connectées permettant de faire des relevés de données physiologiques (pouls, gyrométrie, accélération linéaire) ;
— une application utilisée durant leurs campagnes de levée de fonds et de test ;
— des smartphones pour la collecte et l'envoi des données des montres connectées.

I.2 Objectifs initiaux et leur évolution

Les objectifs initiaux de notre projet, tels que nous avons pu les exposer dans la proposition détaillée, étaient les suivants :

— mettre au point plusieurs algorithmes d'apprentissage permettant de détecter les activités effectuées ;
— mettre en place un protocole de récupération des données auprès du patient ;
— optimiser les algorithmes ainsi que le protocole de recueil des données ;
— développer les algorithmes afin qu'ils soient en temps réel.

Depuis l'élaboration de nos objectifs et de notre PSC en début d'année académique, de nombreux facteurs nous ont poussé à redéfinir et reformuler les objectifs et les limites de notre projet. Ces problèmes seront décrits en détail dans la partie suivante, concernant nos démarches. Les principales conséquences ont été l'impossibilité d'exploiter nos propres données, ainsi que la perte de l'accès aux données de gyrométrie et d'accélération linéaire. Ces changements importants nous ont donc forcé à revoir le cadre de notre étude :

— déterminer la faisabilité la de prédiction efficace l'activité avec une montre et un téléphone ;
— mener une étude comparative permettant de jauger l'importance des paramètres (pouls, accélération linéaire, gyrométrie).

II
DÉMARCHE

II.1 Recueil des données

Notre démarche originale était la suivante :

— trois membres du PSC ont emprunté des montres à Healsy ;
— chacun devait porter la montre et récolter des données pendant plusieurs jours, voire plusieurs semaines ;
— chacun des trois membres notait scrupuleusement les activités effectuées au cours du processus, en notant date et heure de début et de fin, ainsi que le type d'activité ;
— en parallèle, plusieurs membres du PSC mettaient au point des programmes permettant d'adapter les données dans un format directement utilisable pour l'apprentissage.

Le recueil des données se faisait ainsi en mettant en ligne ces dernières sur les serveurs d'Healsy, puis en les récupérant via une interface en ligne dédiée.

II.1.1 • Récupération et traitement des données

Nous disposons en l'état d'un algorithme efficace effectuant ces différentes tâches :

— il prend en entrée des fichiers du type `heart-rate.csv` correspondant aux données brutes fournies par la montre connectée pour une seule donnée physiologique (e.g. le pouls) et sort un tableau épuré avec date et mesure sur deux colonnes ;
— il effectue cela pour le pouls, l'accélération linéaire et le gyromètre, il corrèle ensuite les trois tableaux de données pour en fournir un seul à quatre colonnes : temps (en timestamp), pouls, accélération linéaire, gyromètre. Il dédouble les données si besoin en fonction des fréquences d'acquisition différentes ;
— il lit un fichier texte rempli par l'utilisateur indiquant ses activités avec heure de début et heure de fin, associe un identifiant à chaque activité et ajoute le numéro de cette activité à chaque ligne correspondant à la bonne activité ;
— il effectue un découpage en fenêtres, avec des blocs de données d'une même activité pour augmenter artificiellement leur nombre afin de faciliter l'apprentissage.

Ainsi, en l'état, l'algorithme prend le nom du dossier dans lequel se trouvent les `.csv` correspondant aux données brutes, ainsi que le fichier contenant les activités renseignées par la personne, et sort plusieurs fichiers `.txt` chacun associé à une grandeur mesurée :

— `bpm.txt` qui contient la liste chronologique des pouls ;

— `gyrx.txt`, `gyry.txt`, `gyrz.txt` qui contiennent les données du gyromètre respectivement selon x, y et z ;

— `accx.txt`, `accy.txt`, `accz.txt` qui contiennent l'accélération linéaire respectivement selon x, y et z.

Des données de même rang correspondent à une même mesure. Par exemple, si l'on a les fichiers suivants :

Fichier	Première valeur	Deuxième valeur	...
`bpm.txt`	`79`	`81`	`...`
`gyrx.txt`	`3e-2`	`4e-2`	`...`
`gyry.txt`	`6e-2`	`4e-2`	`...`
`gyrz.txt`	`2e-2`	`1e-2`	`...`
`accx.txt`	`2e-2`	`1e-2`	`...`
`accy.txt`	`1e-2`	`7e-2`	`...`
`accz.txt`	`5e-2`	`5e-2`	`...`

Alors `[79, 3e-2, 6e-2, 2e-2, 2e-2, 1e-2, 5e-2]` (i.e. la première colonne) correspond à la première mesure, et `[81, 4e-2, 4e-2, 1e-2, 1e-2, 7e-2, 5e-2]` (i.e. la deuxième colonne) correspond à la deuxième mesure.

Suite à des changements au cours du PSC, les données que nous fournit actuellement Healsy, nous avons retravaillé notre programme afin qu'il n'ait plus à récupérer que des données de pouls, car l'accélérométrie n'était plus recueillie par les applications d'Healsy.

II.1.2 • Difficultés face aux données récupérées

L'un des problèmes les plus importants qui nous a contraint à repenser entièrement notre démarche a été le recueil de données significatives et correctes. Comme évoqué précédemment, Healsy a mis en place une plateforme et une application permettant de recueillir des données au format .csv grâce aux montres qui nous ont été prêtées. La première phase de recueil des données était censé se dérouler en début de projet, aux alentours des mois de septembre et d'octobre. Or, nous avions à l'époque relevé de nombreuses incohérences :

— chaque mesure était affectée à un temps sous forme de timestamp (par exemple, 1 531 519 200 correspond au 14 juillet 2018 à minuit pile), mais nos mesures avaient deux temps qui leur étaient attribués. Ces temps étaient parfois espacés de plusieurs jours, parfois de quelques heures, ou encore d'une poignée de secondes. Nous avions théorisé que l'un deux était le temps de mesure et l'autre le temps d'envoi au serveur, ce qui nous a été confirmé par notre tuteur ;

— quand bien même nous avons su déterminer quel temps correspondait à la vraie mesure, une analyse plus fine des données nous a montré que rien n'était cohérent : parfois les dates correspondaient à des moments où aucune mesure n'a été faite, d'autres

fois, le pouls donné ne correspondait pas à l'activité réellement effectuée (pouls de 80 pour une séance de sport intensive) ;

— enfin, des pouls inférieurs à 30 étaient parfois donnés, voire des pouls négatifs ou nuls. Ces problèmes viennent du capteur lui-même, qui parfois a des difficultés à mesurer le pouls.

Concernant ce deuxième point, le plus problématique, nous avons mis en place une solution pour corriger artificiellement ces décalages en temps, qui a consisté à calibrer la mesure grâce à un enregistrement court et précis et de noter sa date de début et de fin afin de repérer le décalage en temps induit sur la mesure. Cependant, après réalisation de ce calibrage, on a remarqué que ce décalage variait d'un enregistrement à l'autre, rendant ainsi le calibrage obligatoire avant toute mesure. En plus d'ajouter une étape laborieuse à tout enregistrement, un enregistrement long peut s'interrompre et se relancer indépendamment de toute manipulation de la montre connectée. Ainsi, ce simple problème nous a beaucoup handicapé et rendu inexploitables toutes les mesures dont le temps a été décalé au cours d'un même enregistrement.

Toutefois, afin de ne pas perdre totalement ces données, nous identifions visuellement les données correspondant aux périodes d'efforts grâce au pouls plus élevés. Cela nous a permis d'avoir quelques enregistrements de longueurs de l'ordre de la minute où l'on a pu associer l'activité qui lui correspondait. Ces données étaient toutefois bien trop peu nombreuses pour les exploiter de quelque manière que ce soit.

Afin de remédier à la plupart de ces anomalies, nous les avons évidemment fait remonter à Healsy qui a alors développé une mise à jour de l'application que nous n'avons reçue que début mars. Du fait de l'éloignement géographique avec Healsy qui ne pouvait intervenir rapidement pour corriger nos problèmes logistiques, nous n'avons pu faire cette mise à jour que sur une seule des montres qu'ils nous ont fournies.

Nous avons alors recommencé les enregistrements entre mars et avril. La récupération des données a été plus lente que sur la première session d'enregistrement d'octobre car, sur cette nouvelle version, la plateforme en ligne ne fonctionnait plus et nous récupérions les données par mail via l'équipe d'Healsy.

Nous avons par la suite mené plusieurs tests et adapté en conséquence le code permettant de rendre ces données utilisables par nos algorithmes de *machine learning*. Malheureusement, un recueil de données récent que nous avons mené entre fin mars et mi avril montre que ces problèmes ont persisté. Donnons à cet effet des exemples bien concrets. Par exemple, dans le fichier .csv figurent des données de la montre d'Alexis datant du 12 avril à 23h40 alors qu'il ne la portait pas. Pire encore, il ne s'agit pas d'un simple décalage temporel de +5 heures comme on le supposait initialement, car nous avons des données qui passent directement du 12/04 à 23h40 au 13/04 à 1h05, et ce alors que le relevé d'Alexis ne mentionne pas de saut de 1h25 dans ses activités.

Nous insistons sur le fait que ces problèmes sont loin d'être anodins et sont fondamentalement handicapants. Si le temps donné n'est pas le bon, alors il est impossible d'attribuer une activité à une mesure ; en conséquence, il est impossible de former un ensemble d'en-

traînement pour nos algorithmes de machine learning, et ainsi, il est impossible de faire du machine learning.

En dernier lieu, l'utilisation de cette nouvelle application de récupération des données a engendré une autre complication. Dans un souci pratique de mémoire vis-à-vis des montres intelligentes qu'Healsy nous avait prêtées, seules les données de pouls sont dorénavant mesurées, alors que nous étions censés disposer avant cela des données d'accéléromètre et de gyrométrie. En effet, les données d'accélération étaient mesurées 3 fois par seconde et saturaient la mémoire de la montre au bout de quelques heures. Or, ces dernières données étaient précieuses et permettaient à nos algorithmes de faire un crible beaucoup plus fin : par exemple, il était possible de détecter la différence entre une montée d'escaliers et une descente d'escaliers (signe opposée de l'accélération). Au vu de cette perte d'information, nous avons décidé de ne pas mettre en œuvre la prédiction en temps réel, car en effet, la précision que nous obtenons uniquement avec le pouls est bien moindre. Nous avons également décidé de reconduire l'étude en prenant en compte ce nouveau facteur, et en comparant pouls seul et pouls avec accélérométrie.

Face à ces difficultés, nous avons décidé d'abandonner ce pan là du projet, de redéfinir nos objectifs et les limites du PSC, et d'appliquer notre partie *machine learning* sur des bases de données disponibles en ligne. Ces bases de données sont décrites en détail dans le paragraphe suivant.

II.1.3 • Spécificité des bases de données

Compte tenu des difficultés évoquées, nous avons orienté notre étude sur l'analyse des résultats obtenus avec les bases de données HARUS [2] et PAMAP2 [10]. Ce sont des bases de données disponibles en ligne, qui ont des données similaires à ce que les montres étaient censées nous fournir.

Il est sans aucun doute judicieux de s'intéresser aux méthodologies et protocoles mis en place pour établir ces bases de données afin d'avoir plus de recul sur les résultats que nous avons obtenus.

Concernant PAMAP2, il y avait 9 testeurs, pour 19 activités à reconnaître au total. Nous pouvons déjà émettre quelques réserves sur les activités : elles sont nombreuses, parfois trop spécifiques, ce qui rend la distinction parfois difficile (on y trouve par exemple : marche nordique, repassage, corde à sauter) et parfois se recoupent (par exemple, être assis et regarder la télévision). De plus, le protocole expérimental demandait aux sujets d'effectuer les 19 activités pendant quelques minutes sur une durée totale d'une heure, cela diffère grandement de notre objectif et de nos méthodes. PAMAP2 recense environ 260 000 données.

Le nombre d'activités que les 30 sujets d'HARUS ont effectuées est plus restreint : elles sont au nombre de 6 et sont moins spécialisées – marcher, monter les escaliers, descendre les escaliers, être assis, être debout, être allongé. Les données obtenues ont également subi des transformations, à savoir un filtre passe-bas du troisième ordre à fréquence de coupure 20 Hz pour enlever le bruit, ainsi qu'une transformation de Fourier rapide sur certaines des

données. Il est également à noter que le protocole d'HARUS ne mesurait que des données d'accéléromètre. HARUS recense environ 1 300 000 mesures.

Par ailleurs, la fréquence de mesure pour PAMAP2 est de 100 Hz, et la fréquence de mesure pour HARUS est de 50 Hz. C'est bien plus élevé que les fréquences des montres connectées que nous utilisons, qui ont une fréquence de 3 Hz pour les données d'accélération, et de 0.2 Hz pour le pouls.

On notera également que les protocoles expérimentaux mis en place diffèrent du nôtre : notre but est d'effectuer un apprentissage sur le quotidien de la personne et sur plusieurs semaines donc, alors que les études évoquées ci-dessus ont mis en place un protocole de quelques heures dans lequel le sujet fait un maximum d'activités différentes.

II.2 Démarche informatique

II.2.1 • Double approche

Pour sa relative facilité d'utilisation et la richesse de ses bibliothèques, nous avons décidé d'utiliser Python. Les deux bibliothèques que nous avons principalement utilisées sont d'une part `scikit-learn` et d'autre part `TensorFlow`.

Nous avons mis en place deux approches radicalement différentes répondant à deux problématiques séparées :

— dans un premier temps, nous avons mis en place un modèle relativement simple à base de SVC (*Support Vector Classification*), une implémentation de SVM, i.e. une machine à vecteurs de support. Nous avons pour cela utilisé `scikit-learn` pour établir ce modèle. D'une part, ce fut une première implémentation, relativement simple et légère, dans une première approche. L'enjeu a par la suite été de tester l'influence du pouls et de l'accéléromètre sur la capacité de prédiction, suite à l'annonce d'Healsy que l'on ne disposerait plus que du pouls. Enfin, nous avons également testé l'influence de fenêtres, que nous expliquerons dans la partie consacrée. L'approche est donc fondamentalement comparative, la plupart des paramètres étaient fixés ;

— nous avons mis en place un modèle beaucoup plus lourd et complexe, un réseau de neurones convolutif. L'enjeu de cette approche était d'optimiser les capacités de notre modèle en influant sur divers paramètres. Cette fois-ci, c'est `TensorFlow` que nous avons principalement utilisé. Cette fois-ci, les deux données de pouls et d'accéléromètre étaient utilisées.

Ces deux approches correspondent à deux objectifs fondamentaux de notre projet : d'une part, optimiser un modèle en influant sur les paramètres (classique de machine learning) ; et d'autre part, faire face à la perte des données d'accélérométrie en jaugeant leur influence sur les résultats obtenus, estimer l'influence des fenêtres.

II.2.2 • Machine à vecteurs de support (SVM)

Comme rappelé précédemment, nous avons utilisé Python avec `scikit-learn`.

La machine à vecteurs de support (SVM) est un algorithme de machine learning très utilisé pour des problèmes de classification, non seulement pour son efficacité mais aussi pour sa relative simplicité.

Le fonctionnement du processus est le suivant. Admettons que l'on possède deux catégories. On possède alors des données d'entraînement et des données de test, en particulier, on sait à quelle catégorie appartiennent les données d'entraînement, mais pas les données de test. Il s'agit ensuite de les représenter dans un espace vectoriel de dimension égale au degré de liberté des données considérées. L'enjeu est d'exhiber un hyperplan affine qui sépare rigoureusement les deux catégories, afin d'établir un processus de décision très simple (la donnée est alors de la catégorie qui se trouve du même côté de l'hyperplan). Or, deux problèmes se posent :

— d'une part, ce n'est pas toujours possible, les données peuvent ne pas pouvoir se séparer linéairement ;

— d'autre part, de nombreux hyperplans conviennent.

Afin de choisir l'hyperplan, on exige de lui qu'il maximise la distance aux deux frontières. Les vecteurs qui réalisent cette distance sont appelés vecteurs de support, d'où le nom.

Le premier problème est beaucoup plus délicat. L'enjeu est donc d'appliquer des transformation successives à l'espace afin de rendre la séparation possible. En particulier, on essaie d'agrandir l'espace, en terme de dimension, dans un espace qui est appelé *l'espace de redescription*. L'opération qui permet d'obtenir cette transformation se fait à travers une opération φ que l'on peut choisir parmi un ensemble de telles opérations.

Enfin, la véritable astuce se trouve dans le *kernel trick* (astuce du noyau). Il s'agit non plus de considérer la fonction de plongement φ, mais la fonction dite de noyau F qui associe à (x, y) la produit scalaire $(\varphi(x), \varphi(y))$. Certains théorèmes d'algèbre linéaire assure que la fonction F convient si, et seulement si, elle vérifie des propriétés spectrales et de continuité. De là, il s'agit donc de piocher des opérateurs de noyau qui vérifient ces propriétés parmi une bibliothèque de fonctions. La fonction F est donc le premier paramètre sur lequel nous pouvons influer.

Cependant, les vecteurs ne sont pas assurés d'être séparables par un hyperplan, même après un choix judicieux de l'opérateur de noyau. C'est pour cela que l'on introduit un deuxième paramètre que nous pouvons modifier, appelé paramètre de pénalité C. Il quantifie la quantité d'erreurs autorisées par la séparation. S'il est bien choisi, les données seront linéairement séparables, et ainsi un processus de décision est bien établi.

Nous avons effectué quelques tests en faisant varier les paramètres de pénalité et l'opérateur de noyau, et les avons fixés ($C = 1$ et `kernel = 'rbf'`) en vue de notre approche d'opposition accélérométrie et pouls.

II.2.3 • Réseau de neurones convolutif (CNN)

Un réseau de neurones est un système composé de plusieurs couches de neurones interconnectées. Dans un problème de classification comme le nôtre, chaque couche de neurones récupère les données de la couche précédente, les transforme et leur attribue une probabilité d'appartenance à chaque activité avant de les transmettre à la couche suivante. Les données sont donc finalement attribuées à l'activité dont le score est le plus élevé à la dernière couche. Enfin, une fonction de perte associée à la dernière couche du réseau calcule l'erreur sur la classification calculée par rapport à au résultat que l'on attendait.

Afin d'optimiser ce processus, on « entraîne » le réseau de neurones sur des données déjà classifiées par activité. Cet entraînement consiste à faire varier les paramètres de la transformation de chaque couche de neurones en utilisant l'algorithme du gradient sur la fonction de perte. On se déplace ensuite dans la direction opposée à ce gradient, afin de se rapprocher ainsi se rapprocher du minimum local.

Après avoir suffisamment entraîné le réseau de neurones, ce dernier peut alors procéder à la classification sur des données non-labellisées.

Les réseaux de neurones convolutifs ont la spécificité d'être plus efficaces sur des données à plusieurs dimensions comme les images. Les données temporelles de pouls et d'accélération que l'on utilise correspondent au format des données adapté à ce type de réseaux.

Afin d'interpréter des données à plusieurs dimensions, une ou plusieurs couches de convolution sont ajoutées au début du réseau de neurones. Celles-ci vont appliquer un filtre aux différentes régions des données afin d'en détecter les particularités, les *features*. Ce filtre est en réalité un produit de convolution entre un *feature* donné et une portion des données afin de pondérer cette portion et ainsi quantifier ça correspondance avec sa *feature*. De plus, entre chaque couche de convolution, une couche de *pooling* est appliquée et va réduire la résolution des données en groupant des données successives afin de pouvoir détecter les *features* à différentes échelles.

C'est après cette étape de convolution où l'on a reconnu tous les *features* que l'on envoie la données de tous ces *features* dans un réseau de neurones classique. Le réseau de neurones convolutif ne s'entraîne donc pas sur toutes les dimensions de données mais sur la donnée des *features* qui est bien plus restreinte, ce qui donne toute l'efficacité à cet algorithme.

Dans notre cas, l'algorithme va prendre une succession de données et, après l'étape de convolution sur une fenêtre, va reconnaître par exemple un balancement du bras. Ce balancement de bras n'est pas caractéristique d'une activité particulière mais s'il a lieu de manière successive sur une durée de quelques minutes, l'algorithme pourra reconnaître une marche.

L'étape de *pooling* consiste elle à regarder une période temporelle plus large et de voir s'il s'agit d'une phase de repos où il y a peu de mouvement même si ponctuellement les capteurs observent un déplacement comme se relever quelques instants dans la nuit par exemple.

Pour la mise en place en pratique, nous utilisons TensorFlow. Pour optimiser au mieux l'apprentissage, nous pouvons faire varier plusieurs paramètres :

— les fonctions de **coût** et **d'optimisation** : la première va permettre au modèle de savoir dans quelle direction aller lors de l'apprentissage. La fonction de cout étant assez particulière aux données, nous ne chercherons pas à utiliser cette fonction comme paramètre (on utilise *softmax with cross entropy*). Le choix de la fonction d'optimisation permet de réduire le temps de calcul au profit d'un résultat plus approximatif. Comme la majorité des algorithmes utilisés dans le domaine qui nous intéresse ici, nous utilisons l'algorithme d'Adam, une extension de l'algorithme de descente de gradient stochastique.

— **epochs** : c'est le nombre de fois que l'on réinsère les données dans le modèle. Plus ce nombre est grand, meilleurs seront les résultats. Toutefois, le cout temporel est assez important. Nous pourrons donc facilement jouer sur ce paramètre.

— **learning rate** : ce paramètre impacte l'optimisation (il agit donc sur la fonction d'optimisation présentée ci-dessus). Plus il est faible, plus l'optimisation est fiable mais plus le temps de calcul est élevé. S'il est trop élevé, il y a un risque élevé d'y avoir un apprentissage qui diverge. C'est un paramètre important à choisir, mais il n'est pas intéressant de le faire varier. Nous le prendrons égal à 0.0001.

— **keep probability** : La valeur `keep_probability` est utilisée pour contrôler le taux d'abandon utilisé lors de la formation du réseau de neurones. Cela signifie que chaque connexion entre les couches (dans ce cas entre la dernière couche et la couche de lecture) ne sera utilisée qu'avec une probabilité de 0.5 lors de la formation. Cela réduit l'*overfitting*.

II.2.4 • Structure du code

L'approche concernant l'apprentissage a été la suivante :

— création d'une classe `DataReader`, qui permet l'extraction et la mise en forme des données d'une base de données, ses arguments sont :

  • `dataset`, le nom de la base de données ;
  • `n` (facultatif), nombre de données maximal.

  et ses fonctions sont :

  • `data_extraction` qui extrait les données sous forme brute ;
  • `split` qui sépare les données en un échantillon d'entraînement et un échantillon test ;
  • `standardize` qui normalise les données ;
  • `windowing` qui sépare les données en différentes fenêtres.

  Cette classe avait pour but de prendre en entrée les fichiers `.txt` épurés fournis par la partie de traitement des données.

— création d'une classe abstraite `Model` pour implémenter plus facilement les différents algorithmes de machine learning, ses attributs sont :

- `data` qui est une instance de la classe `DataReader`

et ses fonctions sont :

- `train`, pour entraîner le modèle sur les données d'entraînement du `DataReader` ;
- `predict` qui, une fois l'entraînement fait, prédit le résultat de ce qu'on lui donne ;
- `efficiency` qui permet de donner la précision du modèle et le tableau des erreurs ;
- `confusion_matrix` qui permet d'obtenir la matrice de confusion (permet de visualiser les erreurs commises par l'algorithme) ;
- `report` qui est une fonction de classification.

— implémentation d'un réseau de neurones convolutif (CNN), avec `tensorflow` ;

— implémentation d'une machine à support de vecteurs (type SVC), avec `scikit-learn`.

III
RÉSULTATS ET ANALYSE : RÉSEAU DE NEURONES (CNN)

III.1 Optimisation des facteurs

III.1.1 • Approche naïve

Dans un premier temps, nous cherchons à avoir la meilleure probabilité possible de trouver la bonne réponse. Sur les deux bases de données que sont PAMAP2 et HARUS, on obtient les résultats suivants, en fraction de bonnes réponses (le paramètre e est pour epochs). On rappelle que epochs est le nombre de fois où l'on réinsère les données dans le modèle, et N le nombre de données que l'on donne en entraînement.

PAMAP2

N \ e	100	200	500	1000
500	0.407	0.513	0.420	0.427
1000	0.236	0.220	0.196	0.230
2000	0.356	0.563	0.717	0.837
5000	0.431	0.504	0.673	0.845
10000	0.222	0.490	0.723	0.889

Les temps d'exécution quant à eux sont les suivants (extraction des données comprises), en secondes :

N \ e	100	200	500	1 000
500	3.76	4.25	4.28	4.51
1000	10.3	7.73	7.92	7.22
2000	76.7	141.90	341	672
5000	75.2	141	341	680
10000	75	142	343	705

Pour HARUS, les résultats sont les suivants :

N \ e	100	200	500	1000
500	0.101	0.227	0.233	0.120
1000	0.130	0.113	0.123	0.213
2000	0.365	0.698	0.918	0.955
5000	0.860	0.945	0.967	0.973
10000	0.941	0.951	0.964	0.978

Les temps d'exécution quant à eux sont les suivants (extraction des données comprises), en secondes :

N \ e	100	200	500	1 000
500	16	16	16.25	16.15
1000	33.5	32	30.7	31.6
2000	160	213	421	807
5000	718	1027	1832	3223
10000	1113	1769	3782	6969

On observe une augmentation globale de la qualité des réponses quand l'on augmente la taille de données en entrée et la répétition du processus, sans avoir assez de données pour pouvoir voir apparaître un effet négatif à cette augmentation.

Nous pouvons émettre d'ores et déjà quelques remarques :

— la partie haut gauche du tableau est assez peu significative étant donnée la forte fluctuation des résultats liée au différents processus probabilistes qui se succèdent (dans la séparation des données entraînement/test, dans l'apprentissage), on ne considèrera pas les 4 cases en haut à gauche dans la suite ;

— pour une augmentation légère du pourcentage de réussite, il faut souvent faire des calculs beaucoup plus coûteux.

— même avec des calculs assez lourds on peine à arriver au dessus de 90% de bonnes réponses pour PAMAP2. Cela vient du fait qu'il y a une activité de plus dans la base de donnée PAMAP2 que dans HARUS. De plus, elles sont plus difficilement discernables ;

— concernant le temps d'exécution, on observe des disparités suffisamment importantes pour ne pas avoir à considérer uniquement la différence d'états de l'ordinateur. Le temps d'exécution prend en compte un temps d'extraction des données, constant sur une ligne du tableau. La structure des données est plus lourde pour la base de données

HARUS, ce qui nous amène à faire des calculs plus longs et créer un décalage affine différent sur les deux bases de données. Ensuite, HARUS met à disposition plus de canaux et donc le temps de traitement augmente ;

— on peut modéliser le temps d'exécution par une fonction affine (modélisation légitime pour N assez grand) du type $T = \alpha(N) + \beta(N) \times \text{epochs}$; alors les coefficients sont plus importants pour HARUS que pour PAMAP2. Enfin, entre $N = 2000$, 5000, et 10000 pour PAMAP2 le temps d'exécution est le même. Cela vient du fait que l'on est obligé d'extraire toutes les données et donc que $\alpha(N)$ ne dépend plus de N à partir d'une certaine valeur.

L'optimisation naturelle qui consisterait donc à augmenter le plus possible N et epochs n'est donc pas forcément très efficace. On choisira donc une autre quantité à maximiser que le simple taux de réussite.

III.1.2 • Fonction de coût

Si l'on revient à notre objectif initial, on cherche à ce que l'utilisateur confirme une activité qu'il est en train de faire. Il y aurait donc une opération à faire. Accepter ou non une activité est aussi couteux en terme de temps pour l'utilisateur que de choisir entre plusieurs activités. Optimiser le taux de prédiction de l'activité n'a alors plus vraiment d'importance, on s'intéresse plutôt à la probabilité que l'activité faite soit dans les 2 ou 3 activités les plus probables. Le choix du nombre de ces activités dépendrait également du nombre d'activités totales parmi lesquels l'algorithme doit choisir.

Aussi, on ne traite toujours pas le problème du temps que prend le calcul. On cherche à améliorer le rapport qualité/temps du mieux possible. On définit ainsi le cout suivant $\frac{p^{1/2}}{temps}$. Ainsi, lorsque l'on cherche à optimiser cette fonction, on pénalise fortement un résultat avec un temps de calcul important mais on ne pénalise pas beaucoup une probabilité de 0.85 par rapport à une probabilité de 0.9. Le faible temps d'exécution des petits N, les favorisant très fortement dans ce calcul, on rajoute une contrainte, par exemple que le taux de réussite soit supérieur à un certain seuil.

On cherche donc à maximiser

$$\frac{p^{1/2}}{temps} \text{ sous la contrainte } p \geq 0.88$$

En reprenant les tableaux précédents en ajoutant ce marqueur, on obtient :

Taux de réussite des deux premières propositions PAMAP2

N \ e	100	200	500	1000
500	0.357	0.523	0.362	0.487
1000	0.279	0.345	0.347	0.240
2000	0.486	0.646	0.887	0.923
5000	0.559	0.604	0.886	0.942
10000	0.650	0.740	0.870	0.930

Enfin, le calcul du coût pour donne respectivement pour PAMAP2 et HARUS. On rappelle qu'on cherche à maximiser le coût sous la contrainte $p \geq 0.88$:

Fonction coût PAMAP2

N \ e	100	200	500	1000
500	0.1588	0.1702	0.1406	0.1547
1000	0.0513	0.0759	0.0745	0.0678
2000	0.0091	0.0056	0.0028	0.0014
5000	0.0099	0.0055	0.0028	0.0014
10000	0.0107	0.0061	0.0027	0.0014

Fonction coût HARUS

N \ e	100	200	500	1000
500	0.0204	0.0298	0.0297	0.0214
1000	0.0107	0.0105	0.0114	0.0146
2000	0.0038	0.0039	0.0023	0.0012
5000	0.0013	0.0009	0.0005	0.0003
10000	0.0009	0.0006	0.0002	0.0001

On peut donc choisir les paramètres optimaux. Pour les deux protocoles, ces paramètres sont $N = 2000$ et epochs = 500. Étant donné qu'il y a 6 ou 7 activités et qu'elles sont globalement équiréparties, cela revient à 300 données par activités. Si une donnée est une fenêtre de captation d'une quinzaine de secondes et que l'on veut séparer les prises de données d'une petite minute, il faut récupérer les données d'une activité sur cinq heures. En pratique, cela peut être assez problématique si l'on ne dispose pas d'une solution complémentaire pour les tailles de données faibles : pour une activité hebdomadaire d'une heure, l'algorithme va mettre 5 heures à être pleinement détectée !

À noter que ces paramètres peuvent être différents pour les deux bases de données : chaque base de donnée à son propre protocole de prise des données et il faut refaire ce processus pour tout autre protocole, en particulier si le nombre d'activités augmente.

III.1.3 • Critique du modèle de choix des paramètres

Ce modèle nous permet de prendre en compte un aspect important pour passer à un échelon plus industriel, le temps.

Toutefois, les fonctions utilisées restant primaires, on est surement pas assez fins dans certaines zones du plan (*p*, *temps*)

La contrainte imposée est elle aussi arbitraire. Suivant les obligations de résultats que l'on s'impose on peut la modifier, en particulier en ajoutant une borne supérieure au temps, pour limiter les temps d'utilisation des serveurs. Cela est d'autant plus vrai que lorsque lon rajoute des données d'apprentissage, il faut relancer une partie de l'apprentissage.

III.2 Analyse à l'optimal

Maintenant que l'on possède des paramètres qui sont intéressants, nous allons analyser plus en détail le résultat. On utilise pour cela des matrices de confusion, qui placent en abscisses l'activité prédite et en ordonnée l'activité réelle, et qui permettent de visualiser les erreurs faites par la prédiction.

Matrice de confusion de HARUS, $N = 2000$, $e = 500$

REAL \ PREDICTED	walking	ascending stairs	descending stairs	sitting	standing	lying
walking	1	0	0	0	0	0
ascending stairs	0	1	0	0	0	0
descending stairs	0	0	1	0	0	0
sitting	0	0	0	0,89	0,11	0
standing	0	0	0	0,13	0,87	0
lying	0	0	0	0	0	1

Matrice de confusion de PAMAP2

REAL \ PREDICTED	lying	sitting	standing	ascending stairs	descending stairs	vacuum cleaning	ironing
lying	0,72	0,24	0	0	0	0,02	0,02
sitting	0	0,63	0,14	0	0	0,02	0,21
standing	0,08	0,08	0,78	0	0	0,04	0,02
ascending stairs	0	0,03	0,06	0,58	0,11	0,22	0
descending stairs	0	0	0,08	0,45	0,27	0,19	0,03
vacuum cleaning	0	0	0,07	0,07	0,07	0,71	0,07
ironing	0,04	0,08	0	0	0	0,02	0,86

On observe tout d'abord la forte disparité en terme de dispersion, comme pouvaient le suggérer les faibles taux de réussite de l'algorithme sur la base de données PAMAP2 par rapport à HARUS. Si l'on augmente N ou epochs, les éléments de la matrice se concentrent de plus en plus autour de la diagonale.

Concernant les activités qui ne sont pas pleinement distinguées, le comportement est assez différent sur les deux bases de données :

— **HARUS** : Toutes les activités sont clairement distinguées à part les positions assise et allongée qui sont confondues. Si l'on se place en termes d'accéléromètres et de cardiomètres, ces deux positions sont assez similaires. Ce qui leur permet d'être différenciées ici (et pas dans PAMAP2), c'est la présence d'un magnétomètre, qui facilite la différenciation. La plupart des montres grand public ne possèdent pas un tel capteur et donc les résultats obtenus avec cette base de données sont largement supérieurs à ce que pourrait obtenir avec des capteurs de plus mauvaise facture.

— **PAMAP2** : La distinction est beaucoup plus compliquée que pour HARUS. Cela vient en particulier de l'absence de magnétomètre dans ce jeu de données. Si l'on regarde plus en détail la matrice de confusion, on peut distinguer trois groupes d'activités : les activités statiques, les activités physiques « complètes » (escaliers, aspirateur) et les activités physiques « incomplètes » (fer à repasser). Il est difficile de distinguer des activités au sein d'un même groupe d'activités mais assez facilement entre les groupes. La finalité de notre projet étant d'avoir le type d'activité physique faite par l'utilisateur, les activités au sein d'un même groupe se ressemblant physiologiquement, cette

dispersion n'est pas forcément un problème.

III.3 Conclusion

Le CNN permet d'avoir des résultats très satisfaisants pour une prédiction sur un petit échantillon d'activités. Avec des données extrêmement complètes (HARUS), on peut espérer atteindre 97% de bonnes prédictions alors qu'avec des données moins complètes, ce chiffre est de l'ordre de 85% (et cela monte à 94% si l'on s'autorise à donner deux choix à l'utilisateur). Toutefois, cela nécessite une durée d'apprentissage très importante (environ 5 heures par activité).

Dans une application à la vie d'un patient, il semble assez peu raisonnable d'utiliser cet algorithme avec une montre classique. En effet, les quantités d'informations nécessaires pour entraîner suffisamment d'algorithme requièrent des temps de captations importants qui, pour des activités assez rares, nécessitent plusieurs semaines voire mois. Une version plus vraisemblable serait de se limiter à quelques groupes d'activités, ayant des propriétés similaires sur la glycémie. Ainsi, on peut espérer avoir des taux de réussite à 2 ou 3 choix proches des 90%.

IV
RÉSULTATS ET ANALYSES : MACHINE À VECTEURS DE SUPPORT (SVC)

IV.1 Avant-propos : guide d'interprétation des résutats

Avant de livrer les résultats et leur analyse, il convient sans doute d'expliquer quelques grandeurs, données et méthodes implémentées.

Premièrement, la base de données utilisée dans toute l'analyse qui suit est la base PAMAP2, avec une machine à support de vecteurs SVC. Nous avons choisi de prendre 75% des données pour l'entraînement et 25% des données pour le test. Enfin, les paramètres majeurs du SVC ont été choisis parmi ceux qui sont optimaux : nous avons utilisé un noyau RBF type gaussienne, et un paramètre de pénalité d'erreur de 1. On rappelle que les problèmes multiclasse sont traités selon le schéma *one-vs-one*.

Parmi les activités figurant dans le protocole PAMAP2, nous nous sommes intéressés aux activités suivantes, dont on donne les identifiants :

Identifiant	Activité
1	Allongé
2	Assis
3	Debout
4	Marche
5	Course à pied
6	Vélo

Identifiant	Activité
7	Marche nordique
12	Ascension escaliers
13	Descente escaliers
16	Ménage (aspirateur)
17	Repassage

Deuxièmement, trois grandeurs sont données dans l'analyse. La première est donnée par la précision :

$$\text{précision}_i = \frac{\text{nb d'activités correctement attribuées à l'activité } i}{\text{nb de d'activités attribuées à l'activité } i}$$

La deuxième est le rappel (recall en anglais) :

$$\text{rappel}_i = \frac{\text{nb d'activités correctement attribuées à l'activité } i}{\text{nb de d'activités étant effectivement } i}$$

Enfin, la troisième, le f1-score est la moyenne harmonique de la précision et du rappel :

$$\text{f1-score}_i = \frac{2}{\frac{1}{\text{rappel}_i} + \frac{1}{\text{precision}_i}}$$

Le f1-score est un moyen d'obtenir une appréciation de la qualité de l'algorithme : pour qu'il soit élevé, il faut que les deux grandeurs soient élevés. C'est le f1-score que l'on aura tendance à regarder. Par souci de rigueur intellectuelle, les résultats détaillés par activité seront également systématiquement données.

Enfin, nous discuterons en détail de l'influence de ce que nous avons appelé « fenêtres » sur les résultats. Afin d'augmenter la pertinence des données et d'augmenter les corrélations, nous avons choisi de mettre un place un découpage des données. Par exemple, admettons que le signal original corresponde au tableau à gauche suivant :

Sans fenêtre

Temps	Pouls
1524922693	103
1524922698	98
1524922703	101
1524922708	102
1524922713	99
...	...

Avec fenêtre taille 2

Temps	Pouls
1524922693	103
1524922698	98
1524922703	101
1524922708	102
1524922713	99
...	...

Plutôt que de fournir directement ces données là, nous les découpons en blocs. Par exemple, une fenêtre de taille 2 consistera à regrouper les données comme dans le tableau précédent à droite. Cela permet de traiter chaque bloc de 2 données comme une unique donnée, nous verrons l'influence de cette technique sur les résultats que nous obtenons. Un gros désavantage de cette méthode est la le temps d'exécution de l'algorithme qui s'en trouve grandement dégradée.

IV.2 Analyse du graphique des données pouls – accéléromètre

IV.2.1 • Analyse fréquentielle des signaux

Une première approche qui nous permettra de diriger l'étude peut consister à étudier les signaux bruts, corrélés avec les activités. Comme nous pouvons le voir sur la figure page suivante, la simple visualisation des 4 données (pouls et accéléromètres) nous donne intuitivement énormément d'informations.

C'est sur les accéléromètres que l'on distingue le plus d'informations : en observant le graphique de l'accélérométrie en fonction du temps, on a intuitivement envie de découper l'espace temporel en différentes zones correspondant à différentes activités. Lors de chaque activité, il y a manifestement des grandeurs caractéristiques, que ce soit la moyenne, l'amplitude des oscillations ou même leur fréquence ; toutefois, le bruit semble prendre une pace très importante. Or, un apprentissage automatique va naturellement répertorier ces différentes pulsations aux différentes activités assez facilement. La force du *machine learning* est que l'on n'aura pas besoin d'éliminer de bruit. Une fenêtre de données de très faible grandeur (2-3 oscillations) devraient permettre à un algorithme de *machine learning* de dé-

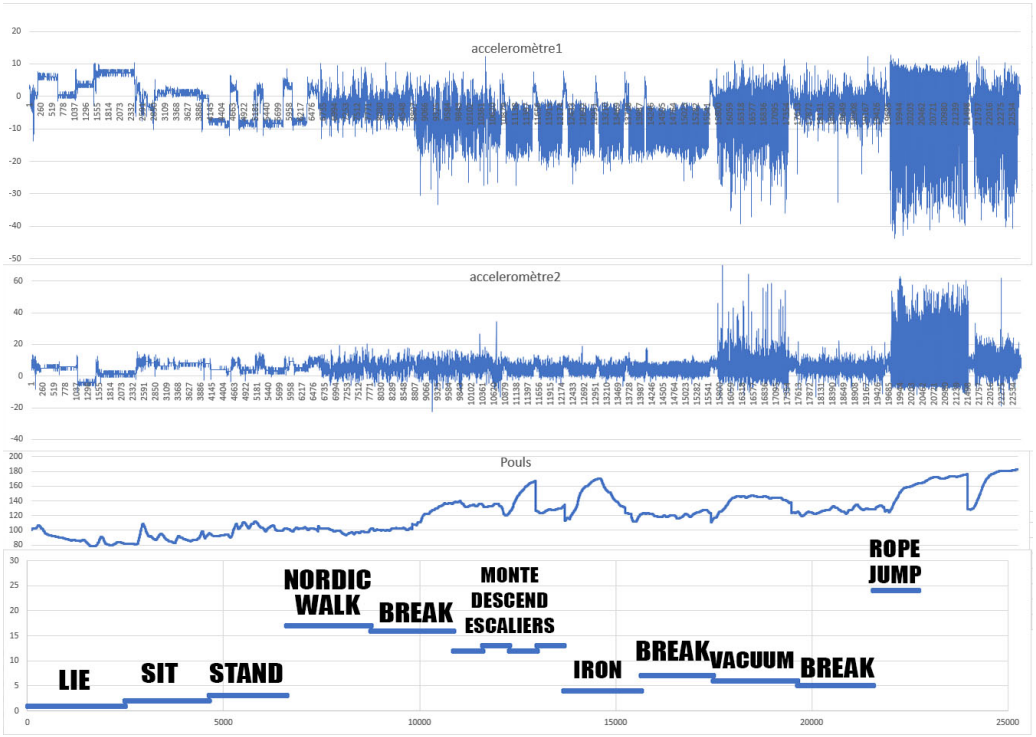

Figure 2 – Signaux corrélés avec l'activité

terminer l'activité en cours.

D'un autre coté, le pouls ne semble nous donner intuitivement aucune information : il est en fait le résultat d'activités antérieures, et l'activité présente n'agira sur le pouls qu'après qu'elle soit terminée. Cela paraît logique, car s'astreindre uniquement au pouls pour l'analyse revient à faire une analyse en « une dimension », peu discriminante. Naïvement, le pouls semble être un paramètre peu déterminant pour le résultat que nous souhaitons obtenir.

Mais si le pouls ne nous donne pas d'information sur l'activité en cours, il nous en donne en revanche sur celles qui viennent d'avoir lieu, et donc sur l'état physique du patient. Or, les informations données par les accéléromètres semblent des indicateurs très forts de l'activité en cours. Seulement nous avons observé cela suite à un échantillon de données d'une personne suivant un protocole précis au cours d'une journée. La question naturelle est donc de se demander si les activités antérieures auraient pu influencer l'activité en cours.

Si l'apprentissage automatique marche pour une journée, là où les activités ne sont faites qu'une seule et unique fois, et donc toujours à la même fatigue, il est pas sûr qu'il fonctionne aussi bien pour des échelles de temps plus grandes, là où une même activité sera corrélée avec des pulsations sensiblement différentes en fonction des états de fatigues de l'individu.

Récapitulons nos observations et hypothèses permises par ce graphique : un apprentissage automatique permettrait assez facilement de déterminer une activité grâce aux données de l'accéléromètre et sur une fenêtre de temps assez courte. La donnée du pouls qui n'a pas le temps de varier lors d'une activité nous donnerait alors une indication sur l'état de fatigue

de l'individu, et donc d'affiner (ou plutôt de rendre possible) la catégorisation en fonction de l'accéléromètre.

IV.2.2 • Accéléromètre seul, pouls seul

Afin de pouvoir discuter de l'importance du pouls et de l'accélérométrie pour l'étiquetage des données, nous avons procédé à plusieurs tests. En utilisant SVC, nous avons testé l'accélérométrie seule pour l'entraînement, avec des fenêtres de 1 et 10 respectivement. En incluant des fenêtres de 10, les erreurs sont presque divisées par deux, tenant ainsi compte des oscillations que nous avons mentionnées.

SVC, accéléromètre uniquement,
fenêtres de taille 1
f1 = 0.56

Id	precision	recall	f1-score	support
1	0.86	0.90	0.88	603
2	0.55	0.81	0.65	554
3	0.55	0.86	0.67	484
4	0.45	0.67	0.54	517
5	0.63	0.50	0.56	88
6	0.62	0.72	0.67	521
7	0.68	0.43	0.53	433
12	0.40	0.15	0.22	344
13	0.57	0.27	0.37	362
16	0.49	0.35	0.41	535
17	0.54	0.38	0.44	559
Avg.	0.58	0.58	0.56	5000

SVC, accéléromètre uniquement,
fenêtres de taille 10
f1 = 0.76

Id	precision	recall	f1-score	support
1	0.83	0.96	0.89	52
2	0.60	0.59	0.59	70
3	0.60	0.89	0.72	54
4	0.75	0.84	0.79	55
5	1.00	0.43	0.60	7
6	0.83	0.86	0.85	51
7	0.89	0.88	0.88	48
12	0.88	0.57	0.69	37
13	0.88	0.79	0.84	29
16	0.85	0.75	0.80	52
17	0.73	0.51	0.60	43
Avg.	0.77	0.76	0.76	498

Nous avons également procédé au même protocole pour le pouls seul. Voici les résultats obtenus, ci-dessous.

SVC, pouls uniquement,
fenêtres de taille 1
f1 = 0.58

Id	precision	recall	f1-score	support
1	0.82	056	0.67	603
2	0.67	0.71	0.69	554
3	0.51	0.69	0.58	484
4	0.59	0.81	0.68	517
5	0.00	0.00	0.00	85
6	0.45	0.62	0.52	520
7	0.76	0.76	0.76	437
12	0.62	0.47	0.54	344
13	0.00	0.00	0.00	362
16	0.63	0.71	0.67	535
17	0.61	0.67	0.64	559
Avg.	0.57	0.61	0.58	5000

SVC, pouls uniquement,
fenêtres de taille 10
f1 = 0.58

Id	precision	recall	f1-score	support
1	0.71	0.58	0.64	52
2	0.68	0.71	0.70	70
3	0.55	0.54	0.54	54
4	0.56	0.69	0.62	55
5	0.00	0.00	0.00	7
6	0.49	0.67	0.57	51
7	0.77	0.75	0.76	48
12	0.67	0.38	0.48	37
13	0.00	0.00	0.00	29
16	0.61	0.71	0.65	52
17	0.53	0.79	0.64	43
Avg.	0.57	0.61	0.58	498

De ces résultats expérimentaux on peut en conclure qu'effectivement les données de l'accéléromètre sont les plus utiles pour déterminer l'activité en cours, surtout avec des fenêtres contenant plusieurs oscillations de la montre. On remarque également qu'à cette échelle, les fenêtres ont peu d'importance lorsque le pouls est seul.

Les seules données du pouls semblent nous donner des résultats relativement corrects, même si bien en deçà de nos attentes. Toutefois, il convient d'étudier ces résultats avec circonspection. Dans notre PSC, les données que nous avons utilisées en l'absence des données de nos montres sont des données très particulières. Elles sont fondées sur quelques jours de suivi de protocole par des personnes portant une montre. Ainsi les activités ne sont réalisées qu'un petit nombre de fois par chaque personne. Or les données en temps réel telles que le pouls dépendent énormément des activités passées, les données mesurées lors d'une activité sont donc censées être assez différentes suivant le jour où elles sont collectées.

Les données que nous avons utilisées manquent manifestement de diversité. Pour simplifier, c'est comme si dans nos données, à chaque fois que le porteur de montre courait le pouls était entre 110 et 120, tandis qu'à chaque fois qu'il marchait le pouls était entre 60 et 70. Comme nous ne contrôlons pas la manière de raisonner de nos algorithmes, nous ne pouvons pas l'empêcher d'établir comme raisonnement que si le pouls est entre 110 et 120 alors le porteur de montre est en train de courir. En effet en procédant de cette manière l'algorithme peut facilement obtenir très bons résultats aux tests. C'est le phénomène d' « overfitting », c'est-à-dire que l'algorithme apprend « par cœur »

Cependant, en situation réelle, nous aurions un apprentissage sur plusieurs journées, et alors cet « overfitting » ne fonctionnerait plus du tout, et les résultats avec le pouls seul seraient sans doute bien moins satisfaisants.

Maintenant si nous utilisons à la fois les données de l'accéléromètre et celles du pouls pour des fenêtres de taille 10 on obtient un f1-score moyen de 0.8434.

SVC, pouls et accéléromètre, fenêtres de 10

Id	precision	recall	f1-score	support
1	0.84	0.94	0.89	52
2	0.97	0.84	0.90	70
3	0.82	0.91	0.86	54
4	0.76	0.85	0.80	55
5	1.00	0.43	0.60	7
6	0.88	0.98	0.93	51
7	0.84	0.90	0.87	48
12	0.83	0.54	0.66	37
13	0.68	0.66	0.67	29
16	0.84	0.79	0.81	52
17	0.89	0.93	0.91	43
Avg.	0.85	0.84	0.84	498

Le pouls donne donc des informations différentes que celles de l'accéléromètre car en combinant ces 2 sources d'informations, on obtient de bien meilleurs résultats. Attention, il

faut là aussi remarquer qu'il y a pu y avoir un over-fitting sur les données du pouls, et donc que cela marcherait beaucoup moins bien sur plusieurs jours.

IV.3 Réflexions sur les fenêtres d'échantillonnage

IV.3.1 • Pouls et accéléromètre

D'après nos précédentes hypothèses et expérimentations, l'accéléromètre donne des informations très importantes sur des fenêtres contenant plusieurs oscillations tandis que le pouls qui ne varie pas beaucoup donne à peu près les mêmes informations selon les tailles de fenêtres : par exemple, avec des fenêtres de taille 1 ou 10 nous obtenons des résultats très semblables. La suite logique de notre démarche scientifique est donc de s'intéresser à l'influence de la forme et la taille des fenêtres.

Intuitivement, le pouls ne variant pas très vite, il n'y a que très peu de fluctuations dans une fenêtre donnée. Or la variation du pouls nous donne des informations précieuses quand aux activités assez récentes. En effet si une activité épuisante ne se ressent pas en temps réel sur les valeurs du pouls, elle doit être discernable sur les valeurs de la dérivée du pouls. Pour calculer ces dérivées nous avons malheureusement besoin de fenêtres beaucoup plus grandes, alors que le pouls varie lentement. En réalité nous avons besoin de quelques mesures assez espacées dans le temps.

Essayons notre apprentissage automatique avec des fenêtres de petites tailles ou de grandes tailles avec des valeurs espacées sur notre base de donnée PAMAP2, uniquement avec le pouls.

SVC, pouls uniquement
fenêtres de taille 1
f1 = 0.58

Id	precision	recall	f1-score	support
1	0.82	0.56	0.67	603
2	0.67	0.71	0.69	554
3	0.51	0.69	0.58	484
4	0.59	0.81	0.68	517
5	0.00	0.00	0.00	85
6	0.45	0.62	0.52	520
7	0.76	0.76	0.76	437
12	0.62	0.47	0.54	344
13	0.00	0.00	0.00	362
16	0.63	0.71	0.67	535
17	0.61	0.67	0.64	559
Avg.	0.57	0.61	0.58	5000

SVC, pouls uniquement
fenêtres de taille 100
f1 = 0.68

Id	precision	recall	f1-score	support
1	0.79	0.58	0.67	566
2	0.71	0.79	0.75	562
3	0.59	0.58	0.58	493
4	0.66	0.79	0.72	525
5	0.00	0.00	0.00	90
6	0.48	0.77	0.59	525
7	0.88	0.75	0.81	440
12	0.93	0.63	0.75	351
13	1.00	0.42	0.59	349
16	0.73	0.71	0.72	522
17	0.67	0.86	0.75	552
Avg.	0.71	0.69	0.68	4975

L'approche avec des fenêtres de taille 100 est plus précise, mais cela se paie dans la complexité temporelle de l'algorithme.

En prenant cette fois des fenêtres de 10 mesures chacune espacées de 10 mesures, couvrant ainsi l'étendue d'une fenêtre de 100 mesures, nous obtenons un f1-score moyen de 0.68.

Id	precision	recall	f1-score	support
1	0.80	0.58	0.67	566
2	0.71	0.80	0.75	562
3	0.59	0.58	0.59	493
4	0.66	0.79	0.72	525
5	0.00	0.00	0.00	90
6	0.48	0.77	0.59	525
7	0.89	0.75	0.81	440
12	0.93	0.63	0.75	351
13	1.00	0.41	0.58	349
16	0.73	0.70	0.71	522
17	0.67	0.84	0.75	552
Avg.	0.71	0.69	0.68	4975

C'est très encourageant, d'une part car le score est presque le même, quoiqu'en réalité légèrement inférieur, et car la complexité temporelle est largement amoindrie.

À titre uniquement instructif sur le phénomène d'overfitting, si nous procédons au même protocole, mais cette fois-ci avec des mesures espacées de 100, couvrant ainsi 1000 mesures, le f1-score moyen est de 0.97.

Id	precision	recall	f1-score	support
1	0.99	1.00	1.00	363
2	1.00	0.99	1.00	535
3	0.94	0.91	0.92	484
4	1.00	1.00	1.00	505
5	1.00	0.55	0.71	92
6	0.90	1.00	0.95	510
7	1.00	1.00	1.00	462
12	1.00	0.95	0.97	361
13	0.99	1.00	1.00	366
16	1.00	0.94	0.97	524
17	0.92	0.99	0.96	548
Avg.	0.97	0.97	0.97	4750

De si bons résultats avec des données si restreintes laissent penser que nous avons sans aucun doute un cas d'overfitting, c'est-à-dire que l'algorithme se contente « d'apprendre par c œur » les données. Toutefois, le reste du temps, quand nous utilisons des fenêtres de tailles modérées, avec des résultats de lordre de 80% ou même 60% de réussite, nous ne pouvons affirmer avec certitude que notre algorithme ne fait pas un overfitting.

En conclusion, hormis le cas de l'overfitting, nous avons montré qu'il valait mieux prendre des mesures espacées pour le pouls, afin d'obtenir des variations significatives et exploitables par l'apprentissage. Cependant nous pouvons aller plus loin dans l'évolution de nos fenêtres.

Plutôt que de garder des fenêtres de données espacées linéairement, une approche intéressante consisterait à favoriser des fenêtres espacées non linéairement. En effet, intuitivement, les données éloignées sont moins importantes que les données proches.

À partir de ce constat, nous avons essayé une autre forme de fenêtre : des fenêtres de types « carré » qui considérait pas exemple les données pouls-accéléromètre aux temps $t-1$, $t-4$, $t-9$, $t-16$, etc. Intuitivement, cela permet à l'algorithme de hiérarchiser les données selon leur importance, qui décroît avec le temps.

Voici les résultats de notre algorithme pour des fenêtres carrées de 5 mesures, sur PAMAP2, avec pouls et accéléromètre, avec un f1-score moyen de 0.90 ; contre 0.92 pour des fenêtres linéaires de 25 mesures.

SVC, pouls et accéléromètre
fenêtres carrées de taille 5
f1 = 0.90

Id	precision	recall	f1-score	support
1	0.97	0.96	0.97	598
2	0.98	0.96	0.97	552
3	0.89	0.94	0.91	488
4	0.84	0.93	0.88	523
5	0.98	0.73	0.84	88
6	0.92	0.94	0.93	512
7	0.87	0.90	0.89	472
12	0.87	0.78	0.82	352
13	0.87	0.77	0.82	334
16	0.83	0.84	0.84	529
17	0.93	0.95	0.94	546
Avg.	0.90	0.90	0.90	4994

SVC, pouls et accéléromètre
fenêtres linéaires de taille 25
f1 = 0.92

Id	precision	recall	f1-score	support
1	0.97	0.94	0.95	598
2	0.97	0.95	0.96	554
3	0.89	0.92	0.91	484
4	0.90	0.92	0.91	516
5	1.00	0.62	0.77	98
6	0.95	0.94	0.95	514
7	0.86	0.97	0.91	469
12	0.93	0.89	0.91	355
13	0.96	0.86	0.90	361
16	0.90	0.91	0.91	493
17	0.91	0.95	0.93	554
Avg.	0.93	0.92	0.92	4996

Les résultats sont sensiblement les mêmes. Nous avons donc « effacé » des données n'apportant pas plus d'information à l'algorithme. L'avantage est la vitesse d'exécution de l'algorithme, manifestement améliorée. Les fenêtres de formes non linéaires sont donc presque aussi efficaces que des fenêtres de formes linéaires, mais elles permettent de réduire considérablement le temps d'entrainement de l'algorithme. Cette méthode semble donc très prometteuse.

IV.3.2 • Pouls seul

Avec les paragraphes ci-dessus, on pressent que le plus efficace est d'utiliser des fenêtres non linéaires, afin de suivre une d'évolution temporelle, qui est due à l'activité en cours.

En testant notre algorithme sur PAMAP2 avec des fenêtres carrées de taille 5, on a un f1-score de 0.62 avec le pouls seul.

Id	precision	recall	f1-score	support
1	0.82	0.56	0.66	598
2	0.66	0.72	0.69	552
3	0.49	0.61	0.54	488
4	0.61	0.81	0.70	523
5	0.00	0.00	0.00	88
6	0.47	0.71	0.56	512
7	0.79	0.74	0.76	472
12	0.79	0.45	0.57	352
13	0.89	0.15	0.26	334
16	0.64	0.69	0.67	529
17	0.58	0.69	0.63	546
Avg.	0.65	0.62	0.61	4994

Ces résultats n'ont pas l'air si mauvais, l'algorithme semble être en mesure d'obtenir des informations de ces 5 mesures.

Il convient toutefois d'être critique vis-à-vis de ces derniers résultats. Comme nous l'avions vu grâce à l'analyse fréquentielle en début d'étude, l'algorithme se contente « seulement » de découper les plages de pouls et d'y associer une activité unique, qui est manifestement due au protocole de PAMAP2, assez restreint temporellement. Nous pouvons émettre l'hypothèse que l'apprentissage avec le pouls seul devienne de plus en plus difficile à mesure que la quantité de données augmente, car c'est une prédiction unidimensionnelle, mais nous devrions pour cela disposer de plus de données.

CONCLUSION

Les algorithmes que nous avons mis en place permettent de prédire relativement efficacement l'activité suivie par le patient. Cependant, plusieurs écueils demeurent :

— une difficulté majeure se trouve dans la récupération de données fiables, tant au niveau des grandeurs physiologiques mesurées qu'au niveau de la vraisemblance des temps de mesures. C'est une difficulté que nous n'avons pas su surmonter ;

— l'importance du traitement que doivent subir ces mêmes données avant d'être exploitables est également conséquent, comme l'identification des activités pour les données du set d'entraînement ou le recoupement des différentes grandeurs, mesurées à des fréquences distinctes ;

— en l'état, la quantité de données requise pour obtenir un niveau de précision correct est conséquente, ce qui est un processus lourd ;

— les montres actuelles sont aussi un facteur limitant important. Leur mémoire limitée ont poussé Healsy à ne recueillir que des données de pouls et d'abandonner les données d'accélération linéaire et de gyrométrie, saturant les mémoires des montres. Notre étude comparative montre que la prédiction est théoriquement faisable, toutefois avec des degrés de précision décevants.

Toutefois, l'étude que nous avons menée et les résultats que nous avons obtenu permettent sans aucun doute d'établir une démonstration de faisabilité forte. Le taux de prédiction légèrement décevant que nous avons obtenu uniquement avec le pouls reste satisfaisant et peut être amélioré. De même, les résultats que nous avons obtenus avec le réseau de neurones (CNN) permettent de quantifier de manière significative la précision que l'on peut obtenir de façon réaliste, ainsi que ses rapports avec le coût temporel.

BIBLIOGRAPHIE

[1] Charu C. Aggarwal et Chandan K. Reddy. *Data Clustering : Algorithms and Applications*. Chapman & Hall/CRC, 2013.

[2] David Anguita et al. *A Public Domain Dataset for Human Activity Recognition Using Smartphones (HARUS)*. url : https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones.

[3] Muhammad Arshad Awan et al. "Human Activity Recognition in WSN : A Comparative Study". In : *International Journal of Networked and Distributed Computing* 2.4 (oct. 2014), p. 221-230.

[4] Joachim Connault. 2004. url : http://lacote.ensae.net/SE206/Cours/Joachim.Connault.pdf.

[5] Tensor Flow. url : https://github.com/tensorflow/tensorflow.

[6] Ian Goodfellow, Yoshua Bengio et Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[7] Tâm Huynh et Bernt Schiele. "Analyzing Features for Activity Recognition". In : *Proceedings of the 2005 Joint Conference on Smart Objects and Ambient Intelligence : Innovative Context-aware Services : Usages and Technologies*. sOc-EUSAI '05. Grenoble, France : ACM, 2005, p. 159-163.

[8] INRIA. 2015. url : http://scikit-learn.org/stable/.

[9] Jafet Morales et David Akopian. "Physical activity recognition by smartphones, a survey". In : *Biocybernetics and Biomedical Engineering* (mai 2017).

[10] Attila Reiss. *PAMAP2 : Physical Activity Monitoring Dataset*. url : https://archive.ics.uci.edu/ml/datasets/pamap2+physical+activity+monitoring.

[11] Attila Reiss et Didier Stricker. "Creating and Benchmarking a New Dataset for Physical Activity Monitoring". In : *Proceedings of the 5th International Conference on PErvasive Technologies Related to Assistive Environments*. PETRA '12. Heraklion, Crete, Greece : ACM, 2012, 40 :1-40 :8.

[12] Attila Reiss, Didier Stricker et Gustaf Hendeby. "Towards Robust Activity Recognition for Everyday Life : Methods and Evaluation". In : *Proceedings of the 7th International Conference on Pervasive Computing Technologies for Healthcare*. PervasiveHealth '13. Venice, Italy : ICST (Institute for Computer Sciences, Social-Informatics et Telecommunications Engineering), 2013, p. 25-32.

[13] Aqib Saeed. url : https://aqibsaeed.github.io/2016-11-04-human-activity-recognition-cnn/.

[14] Health Data Science. url : https://github.com/healthDataScience/deep-learning-HAR.

[15] scipy.stats. url : https://docs.scipy.org/doc/scipy/reference/stats.html.

[16] Luis Miguel Soria Morillo et al. "Low Energy Physical Activity Recognition System on Smartphones". In : *Sensors* (2015).

[17] Benyue Su et al. "A Novel Method for Short-time Human Activity Recognition Based on Improved Template Matching Technique". In : *Proceedings of the 15th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry - Volume 1*. VRCAI '16. Zhuhai, China : ACM, 2016, p. 233-242.

[18] Jindong Wang. url : https://github.com/tensorflow/tensorflow.

[19] Jindong Wang et al. "Deep Learning for Sensor-based Activity Recognition : A Survey". In : *Pattern Recognition Letters* (juil. 2017).

[20] Jian Bo Yang et al. "Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition". In : IJCAI'15 (2015), p. 3995-4001.

[21] Yonglei Zheng et al. "Physical Activity Recognition from Accelerometer Data Using a Multi-scale Ensemble Method". In : *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*. AAAI'13. Bellevue, Washington : AAAI Press, 2013, p. 1575-1581.