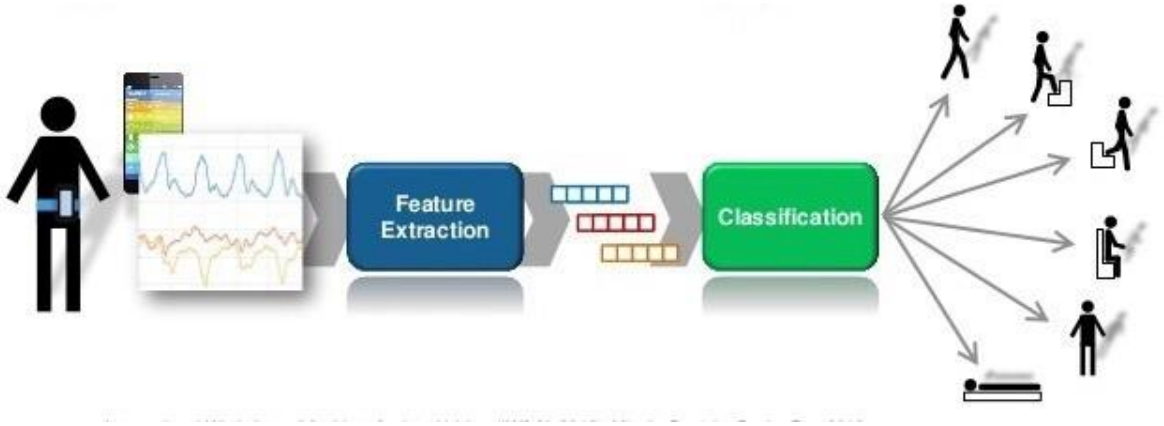
UNIVERSITE PAUL SABATIER



DETECTION D'ACTIVITE HUMAINE - DTW ET CLASSIFICATION AVEC REDUCTION DE DIMENSION

PROJET MCS

LÉA BERQUEZ
M1 INF-DC
VICTOR CATHALA
M1 INF-IARF

Sommaire

I. Evaluation des résultats initiaux

Nous allons dans cette partie présenter les résultats de bases obtenus et les améliorations envisagées et effectuées sur ce code. Tout d'abord nous avons réalisés des fonctions pour chaque partie du code. Cela nous permettra ultérieurement de changer rapidement les paramètres et de faire un maximum de tests pour améliorer nos analyses.

Nous avons pu remarquer que l'algorithme qui effectue le découpage des données en base de test et en base d'apprentissage peut engendrer des résultats biaisés. En effet si TIME_STEP > SEGMENT_TIME_SIZE, il y a un risque de chevauchement des données. On pourra alors retrouver plusieurs fois les mêmes données dans différents segments.

Sur le calcul de la DTW, il nous semble possible de changer la distance et de jouer sur les poids. Sur l'évaluation des résultats de la DTW, on remarque que l'on a une classification d'environ 50%. Ce résultat est donc améliorable. Cependant, le modèle à tendance à confondre certaines classes comme parfois Walking et Downstairs.

Pour en revenir aux résultats, cela peut changer suivant les données utilisées. Il peut y avoir de légères différences, plus au moins 5% de différences sur la précision en fonction des jeux de données utilisées. Cela est dû au fait que les données d'apprentissage et de test sont tirées aléatoirement, donc on n'a pas toujours les mêmes à chaque exécution. Le modèle n'apprend pas sur la même base à chaque fois.

Le prétraitement par l'ACP permet de réduire le nombre de composantes des données d'apprentissage et de test. Ce traitement permet de traiter les composantes qui sont corrélées et permet de réduire le temps de calcul car cela réduit la taille des données. Nous avons constaté qu'il est possible de faire varier le nombre de composantes principales. Nous allons donc chercher à trouver le meilleur nombre afin d'améliorer le temps d'exécution et les résultats.

Nous avons pu remarquer que la classification par les K plus proche voisins est légèrement moins performante que la DTW (environ 38%). Mais cela dépend des données utilisées. Nous pensons qu'il est possible d'améliorer le résultat. Ici, la classe des KPPV de la librairie est utilisée avec 1 seul voisin, il est donc possible de trouver les meilleurs paramètres pour améliorer les résultats. Ce classifieur est mis en place grâce à la librairie sklearn. Cette librairie contient d'autres classifieurs ainsi que GridSearchCV. Nous voyons donc la possibilité de découvrir d'autres classifieurs et de voir s'ils sont plus performants, ce qu'ils peuvent nous apporter. GridSearchCV nous permettra de trouver les meilleurs paramètres afin d'optimiser nos résultats.

Voici donc nos conclusions après la lecture de ce projet. Nous allons essayer de mettre toutes ces choses en places et d'analyser les résultats pour en sortir les éléments les plus pertinents. Vous pourrez retrouver l'ensemble des résultats dans le notebook.

II. Résultats obtenus avec des changements sur les bases de tests, d'apprentissage et/ou de la durée des segments

Le tableau suivant présente les résultats obtenus en fonction des paramètres choisis. Pour chacun des tests nous avons fait des observations et comparé les résultats obtenus avec la DTW et les résultats obtenus avec le KPPV. Vous pouvez retrouver tous ces résultats sur le notebook.

Tests	*Résultat DTW*	*Résultat KPPV*	*Observations*
Test 1 : Avec peu de données *Nombre de classes = 6* *Taille base de test = 10* *Taille base apprentissage = 10* *Durée du segment = 30*	Score = 0,46 Observations : très peu de classes sont reconnues	Score = 0,36 Observations : certaines classes sont confondues avec toutes les autres	Ces résultats sont dû au manque de données d'apprentissage. En effet, il n'y a pas assez de données d'apprentissage pour bien représenter chaque classe. Les résultats varient énormément en fonction du tirage des bases de données.
Test 2 : Avec une base de test bien plus grande *Nombre de classes = 6* *Taille base de test = 200* *Taille base apprentissage = 10* *Durée du segment = 30*	Score = 0,48 Observations : idem que précédemment	Score = 0,37 Observations : idem	C'est le même résultat que le test 1, il n'y a pas assez de données d'apprentissage. Donc la précision de la classification n'est pas très importante.
Test 3 : Avec base d'apprentissage plus grande *Nombre de classes = 6* *Taille base de test = 100* *Taille base d'apprentissage = 1000* *Durée du segment = 30*	Score = 0,49 Observations : on obtient un résultat proche du Test 1. On remarque que certaines classes sont confondues. La précision moyenne varie moins que dans les tests précédents.	Score = 0,82 Observations : on remarque que l'on obtient de bien meilleurs résultats que le Test 1 et 2.	On obtient de meilleurs résultats que précédemment. Néanmoins cela peut être encore améliorer. En effet, il se peut que les données qui ont été choisis pour chaque classe ne sont pas assez représentatives de celle-ci. En théorie, plus on a de données d'apprentissage, plus la représentation de chaque classe est proche de la réalité.
Test 4 : Avec une durée de segment plus petite *Nombre de classes = 6* *Taille base de test = 20* *Taille bas d'apprentissage = 500* *Durée du segment = 10*	Score = 0,47 Observations : la confusion entre walking et upstairs est moins grande que dans le test 1 et 2.	Score = 0,71 Observations : pas de changements significatifs	On ne remarque pas de changements significatif.
Test 5 : Avec une durée de segment plus grande *Nombre de classes = 6* *Taille base de test = 20* *Taille bas d'apprentissage = 500* *Durée du segment = 100*	Score = 0,49 Observations : idem	Score = 0,75 Observations : les résultats semblent légèrement meilleurs	Les résultats semblent meilleurs. Un segment plus grand permet d'avoir plus de variations et donc de mieux identifier la classe.

Ces changements de paramètres ne nous permettent pas de faire de grandes conclusions. On remarque seulement que plus la base d'apprentissage sera grande plus les résultats seront meilleurs. Cela s'explique par le fait que le modèle sera plus entrainé et plus apte à trouver des solutions.

L'amélioration de la DTW est moins significative que sur KPPV car les données d'apprentissage ne servent qu'à calculer le profil moyen, la variation sur le profil moyen n'est pas importante donc on a du mal à améliorer les résultats.

On remarque que la méthode des KPPV est plus performante que la DTW sur nos données.

On a pu souvent remarquer que certaines classes étaient parfois confondues comme par exemple walking/upstairs. Nous allons donc voir si le nombre de classes à un impact sur les résultats de la classification.

III. Résultats obtenus en modifiant le nombre de classes

Précédemment, nous avons remarqué que certaines classes étaient confondues entre elles, nous allons donc essayer de voir l'impact du nombre de classes sur la classification.

Numéro Test	**Nombre de classes**	**DTW**	**KPPV**
1	2 Labels : 'Downstairs' 'Jogging'	Score = 0,75	Score = 0,93
2	2 Labels : 'Jogging' 'Sitting'	Score = 0,96	Score = 0,98
3	4 Labels: 'Downstairs' 'Jogging' 'Sitting' 'Standing'	Score = 0,68	Score = 0,78
4	6 Labels: 'Downstairs' 'Jogging' 'Sitting' 'Standing' 'Upstairs', 'Walking'	Score = 0,47	Score = 0,69

On remarque avec les Test 1 et 2 que le modèle est plus performant quand les données sont significativement différentes. Les courbes de Jogging et Sitting sont vraiment très différentes puisque la première action est en mouvement rapide et l'autre est statique. Il est donc cohérent que le modèle est moins de mal à différencier ces deux classes qu'à différencier 'Downstairs' et 'Jogging' qui peuvent provoquer des courbes similaires. On obtient donc un meilleur score avec le test 2 qu'avec tous les autres tests. Plus on va ajouter des classes, plus le modèle aura à traiter de différences. Il est donc plus susceptible de confondre certaines classes. On remarque également que la méthode des KPPV produit de meilleurs résultats.

IV. Changement de distance pour la DTW

Nous avons essayé de changer le calcul de la distance dans la DTW. Pour cela nous avons utilisé la fonction cdist de la librairie scipy. Pour chacun de nos tests, nous avons la même base de test et d'apprentissage donc les données tirées n'influent pas sur le résultat.

Distance utilisée	**Score obtenu**
Distance initiale euclidienne	0,47
Distance euclidienne avec cdist	0,47
Distance de racine carrée euclidienne	0,50
Distance de Manhattan	0,43

On n'observe pas de différences entre la distance euclidienne de numpy et de scipy, ce qui est logique puisque le calcul des distances est effectué de la même manière. La distance racine carrée euclidienne est meilleure que le calcul avec la distance euclidienne classique. Nous avons effectué des tests avec d'autres calculs de distance mais aucun n'a apporté d'amélioration, nous avons mis dans le notebook l'exemple du calcul de la distance de Manhattan. Ce calcul est la représentation de la norme 1.

V. Autres méthodes de classifications

Nous avons voulu regarder si d'autres classifieurs seraient plus performants sur nos données. Nous avons donc utilisé de nouveaux classifieurs. Le tirage des données n'influe pas dans les résultats puisque chacun des nouveaux classifieurs utilisent les mêmes bases.

Pour ces autres classifieurs on va utiliser les paramètres suivants :

- Taille base de test égale à 100
- Taille base d'apprentissage égale à 500
- Nombre de components égal à 25
- La taille des segments égale à 100

Classifieur	**Précision**	**Analyse**
SVC	Précision de 64%	On remarque qu'il y a une bonne classification pour la classe sitting mais il y a des petites erreurs Upstairs/Downstairs
Gaussian	Précision de 69%	On remarque une bonne classification en général mais il y a des erreurs sur Upstairs/Downstairs
Bernoulli	Précision de 49%	La matrice de confusion n'est pas très bien déssinée. On remarque de nombreuses confusions sur certaines données.
Centroid	Précision de 39%	Cette méthode n'apporte une bonne classification seulement pour la classe sitting. Pour les autres il y a beaucoup de confusion. Par exemple upstairs n'est pas du tout reconnu.
Random Forest Classifieur	Précision de 86%	On obtient un bonne classification avec un résultat rapide

Après ces premiers tests, on remarque que le classifieur random forest produit les meilleurs résultats en un temps rapide. Il est suivi par le classifieur Gaussian. Nous souhaiterions faire une remarque sur les classifieurs : ils sont plus ou moins performants suivant le type de données qu'il traite. En effet, lors de l'UE Fondement de la Recherche d'Information (FRI), Léa a utilisé certains classifieurs pour faire de la classification de page wikipedia sur des films. Elle a pu remarquer que sur les données de FRI le classifieur Gaussian était bien moins performant, elle a préféré dans le cadre de ce projet de FRI utiliser le classifieur de Bernoulli alors qu'ici ce classifieur produit les moins bons résultats.

On remarque donc que certaines méthodes ne sont pas performantes sur nos données. Cela peut être dû aux paramètres passés en entrée. Pour la DTW, l'ACP, la KPPV et d'autres classifieurs nous avons essayé de trouver les meilleurs paramètres pour avoir des résultats optimums.

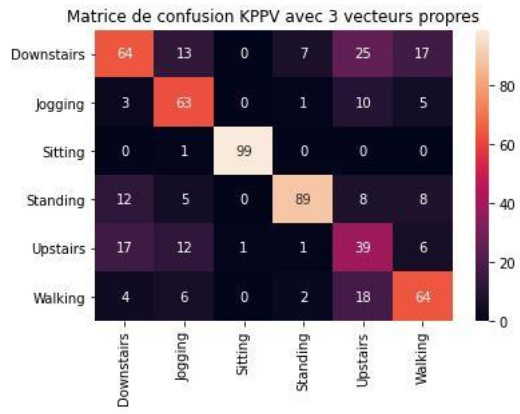
VI. Recherche de paramètres optimaux

1) Nombre de composantes optimales du prétraitement ACP

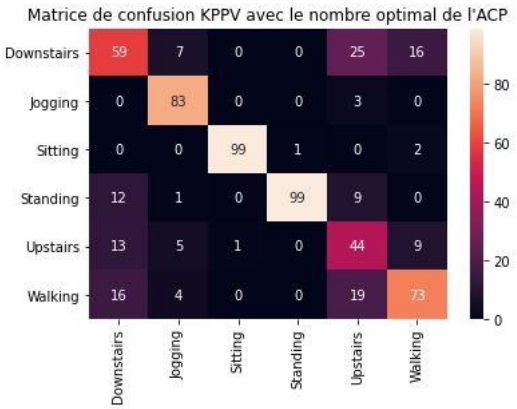
Nous avons créé une fonction pour trouver le nombre de vecteurs propres optimal pour l'ACP. On remarque que nos résultats varient suivant le nombre de vecteurs propres. Ceci s'explique par le fait que on va chercher à trouver la meilleure base pour utiliser les composants principaux et ignorer les données peu pertinentes. On va donc réduire la dimension et effectuer les méthodes sur plus de données en moins de temps. On va donc avoir des données plus précises.

Voici un exemple de résultats :

KPPV avec 3 vecteurs propres dans l'ACP
Score = 69%



KPPV avec optimisation pour l'ACP
Score = 76%



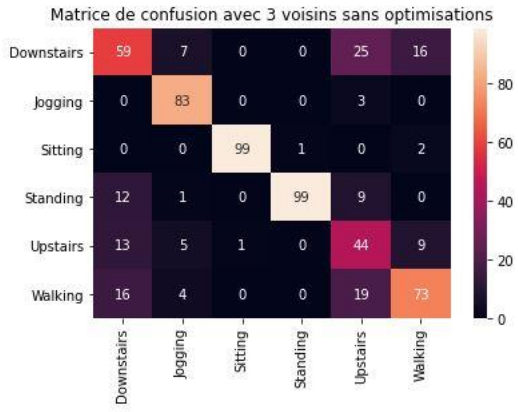
On a obtenu 9 comme meilleur paramètre ACP

2) Nombre de voisins optimal pour KPPV

Nous avons réalisé cette fonction dans le but de trouver le nombre optimal de voisins à regarder. Nous avons choisi de chercher ce nombre entre 1 et 15 voisins. Nous effectuons ce test sur les mêmes jeux de données. Nous avons donc ci-dessous les résultats avec seulement le meilleur ACP et les résultats avec le meilleur ACP et les meilleurs voisins.

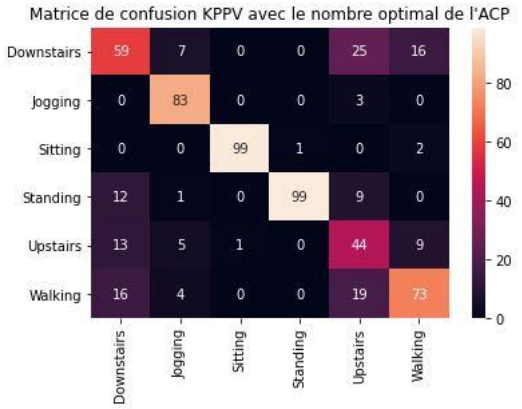
KPP sans optimisation voisins mais avec optimisation ACP

Score = 76%



KPPV avec optimisation pour l'ACP et voisins

Score = 81%



On a obtenu 1 comme meilleur paramètre voisins

On obtient alors une augmentation de la précision lorsque l'on effectue toutes ses améliorations.

3) Utilisation de GridSearchCV

Nous avons utilisé la méthode GridSearchCV de la librairie sklearn. Cette méthode permet de contourner le tri des données. En effet, si les données sont triées, on va chercher à tester sur une base d'apprentissage non entrainée sur ces données. Par exemple si la base d'apprentissage ne se base que sur Walking il va avoir du mal à reconnaitre les autres classes. GridSearchCV permet de tester un certain nombre de paramètres du classifieur sur l'ensemble des données d'apprentissage afin d'obtenir de meilleurs paramètres pour le classifieur et ainsi avoir de meilleurs résultats de classification de manière générale.

Le GridSearchCV utilise la validation croisée. La base d'apprentissage est découpée en k blocs. Une sous base d'apprentissage est formé contenant l'union de k-1 blocs, le dernier étant utilisé pour les tests, et on calcule le score de bonne classification du bloc de test choisi. On effectue ces opérations jusqu'à avoir un score pour chaque bloc en modifiant successivement le bloc de test choisi et la sous base d'apprentissage. On obtient alors une moyenne du score avec un paramètre.

Le GridSearchCV effectue cette validation croisée sur chaque paramètre et retourne celui ayant en moyenne le meilleur score de bonne classification.

Avec ces méthodes, on va essayer de trouver les meilleurs paramètres parmi ceux passé en entrée (choisis par nous). Les résultats obtenus sont présentés dans la section suivante.

4) Conclusion

Classifieur	**Précision paramètres par défaut**	**Précision meilleurs paramètres**	**Meilleurs paramètres trouvés**
ACP	69%	76%	Meilleur ACP à 9
KPPV	76%	81%	Nombre de voisins égal à 1 Meilleur ACP à 9
SVC	64%	70%	`{'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}`
Gaussian	69%	69%	`{'priors': None, 'var_smoothing': 0.001}`
Bernoulli	49%	60%	`{'alpha': 0.1, 'binarize': 2.5, 'class_prior': None, 'fit_prior': True}`
Centroid	39%	41%	`{'metric': 'cityblock', 'shrink_threshold': 0.8}`
Random Forest	86%	86%	`{'bootstrap': False, 'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100, 'oob_score': False}`

La recherche de meilleurs paramètres est soumise à nos choix à nous, on obtient donc des améliorations peu significatives. Il faudrait faire les tests avec des milliers de combinaisons de paramètres différentes mais cela serait très longs. On remarque tout de même une amélioration sur les KPPV avec toutes les améliorations.

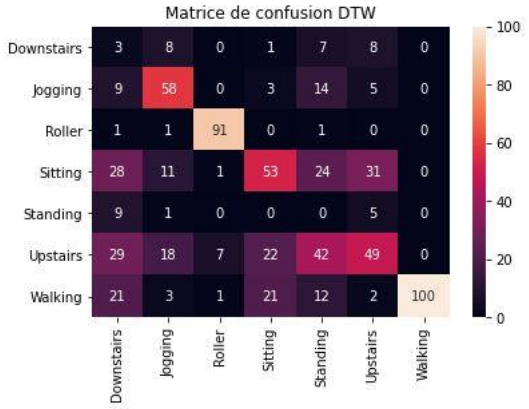
VII. Evaluation sur nos données

Pour aller plus loin, nous avons décidé de créer nos propres données. Nous avons rajouté une classe « roller ». Nous avons donc téléchargé une application pour l'accéléromètre, Léa a enfilé ses rollers, Victor ses chaussures de course et nous avons récoltés quelques données. Pour avoir ces données nous avons régler la fréquence à 20Hz comme les données initiales et nous avons simuler la montre en tenant le téléphone au niveau du poignet.

Nous avons donc rajouté ces données aux données du projet. Pour rajouter ces données nous avons dû légèrement les modifier pour effectuer un changement d'unité. Nous avons appliqué la méthode DTW et KPPV sur ces données.

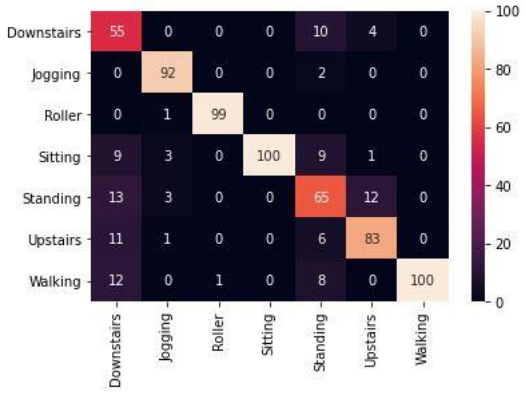
DTW

Score = 51%

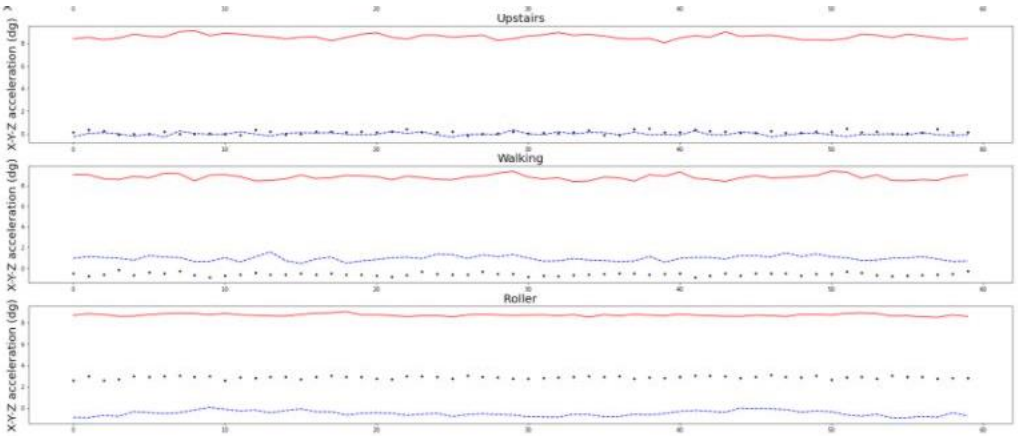


KPPV

Score = 84%



On remarque que cette classe est très bien retrouvée. Malheureusement, il est possible que nos données ne soient pas « conformes » aux autres. En effet, les données de bases ont été prises dans de certaines conditions que nous ne connaissons pas. Si on regarde la courbe, elle semble cohérente avec les autres.



VIII. Conclusion

En conclusion, nous avons pu découvrir les méthodes de classification et ce qu'elles peuvent nous apporter. Nous avons pu nous rendre compte que la méthode des KPPV est assez performante et qu'en général lorsque l'on trouve les meilleurs paramètres on peut obtenir de meilleurs résultats. Nous savons que plus l'on entrainera un modèle plus il sera performant. En revanche nous avons constaté que cela à un coût et que l'on peut avoir des exécutions très longues.

Nous aurions aimé pouvoir avoir le temps de faire plus de classes nous-même et d'ajouter par exemple la fréquence cardiaque aux données.