HES-SO : Multimodal Processing, Recognition and Interaction

Kemboco

Gesture recognition challenge

Pauline Boukhaled et Jonathan Cornaz
08/06/2015

Table des matières

Introduction

Dans le cadre du module MPRI, nous avons appliqué les notions vues au cours via un cas pratique; la reconnaissance gestuelle. Notre but était d'utiliser différents algorithmes de Machine Learning; HMM et ANN et d'obtenir le meilleur score.

Le but était de créer un classifier capable de reconnaitre des gestes humains sur la base d'informations provenant de la Kinect ainsi que d'accéléromètres placés respectivement à l'épaule, le coude, le poignet et la main.

Il y avait 11 gestes différents à reconnaitre :

- Calibration
- Swipe left
- Swipe right
- Push to screen
- Take from screen
- Palm-up rotation
- Palm-down rotation
- Draw a circle I
- Draw a circle II
- Wave hello
- Shake hand

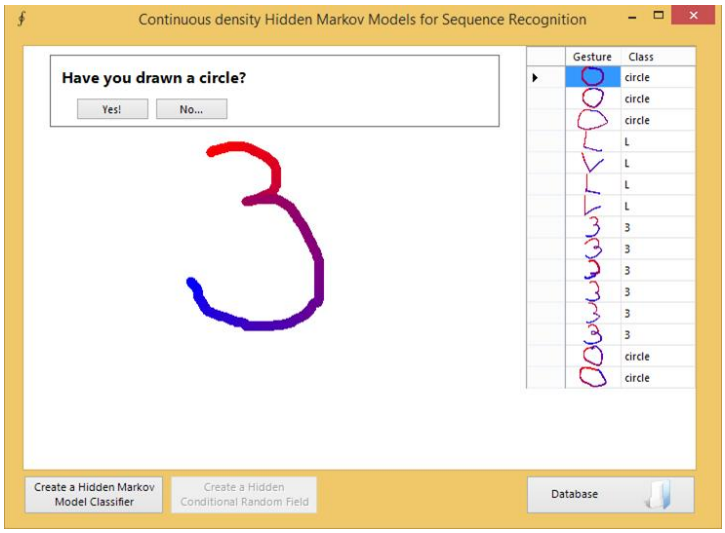
Enfin, il nous fallait déposer nos algorithmes sur la plateforme OPEGRA afin que leurs performances soient évaluées.

HMM a été implémenté par Pauline Boukhaled et ANN par Jonathan Cornaz

Modèle de Markov caché

Analyse

Pour le Modèle de Markov caché, j'ai décidé d'utiliser #C sur Visual Studio plutôt que MatLab car je n'étais pas très à l'aise avec dans les derniers TPs. J'avais besoin d'une librairie pour le Machine Learning. J'ai opté pour la Librairie Accord-framework qui est combinée avec AForge (http://accord-framework.net). J'ai décidé de l'utiliser car elle est pas mal utilisée et sur son site on pouvait trouver de nombreux exemples d'application.
J'ai donc cherché à tester les exemples mis à disposition pour comprendre comment elle fonctionne. Je me suis basée sur l'exemple du mouse gesture recognition qui était plus ou moins similaire au projet MPRI (https://www.youtube.com/watch?v=jn6rjAf3HeQ).



DataSet et Séquence

Le Data Set qui était fourni comprenait une matrice à trois dimensions de cette compositions; [Gestes][Temps][Caractéristiques].
Les caractéristiques sont des accélérations situées à différentes positions. En l'occurrence, 4 points; l'épaule, le coude, le poignet et la main.
On a 11 gestes différents, donc 11 labels.
Il s'agit de savoir quelles caractéristiques on conserve.

Traitement des features

Afin de déterminer quelles caractéristiques garder, j'ai procédé par deux étapes.

- Par observation: j'ai reproduit les gestes en constatant quels points étaient inutiles comme l'épaule par exemple. J'ai essayé de définir le nombre d'états pour chaque geste.
- Par tests: j'ai implémenté mon algorithme en utilisant différents paramètres et en gardant certaines caractéristiques. En choisissant la main, la main et le poignet, la main le poignet et le coude, ainsi que les 4, j'ai obtenu des mauvaises performances. Tandis qu'avec la main et le coude, le score était le meilleur.
- Par matrice de confusion: l'ajustement des paramètres et la détection des erreurs fréquentes ont pu se faire avec cette matrice qui sera décrite plus bas dans les chapitres.

Algorithme

Training

J'ai d'abord échantillonné les caractéristiques afin qu'elles soient toutes de la même taille.
J'ai retenu 60% du dataset et j'ai gardé 100 observations du testset pour regarder les performances.
J'ai créé un classifier différent pour chaque labels en me servant des conclusions obtenues lors de l'analyse des features.
Les classifiers ont été créés en utilisant la distribution normale car c'est celle qui était disponible dans les exemples d'application mais aussi qui donnait le meilleur score. En effet, on a utilisé d'autre distribution comme `GeneralDiscreteDistribution`, mais aucune n'a donné de meilleurs résultats.

```
// Create a new learning algorithm to train the sequence classifier
var teacher = new
HiddenMarkovClassifierLearning<NormalDistribution>(classifier,

    // Train each model until the log-likelihood changes less than 0.001
    modelIndex => new
BaumWelchLearning<NormalDistribution>(classifier.Models[modelIndex])
    {
        Tolerance = 0.0001,
        Iterations = 0
    }
);
```

Evaluate

Sur ma séquence du testset, j'ai échantillonné comme sur le dataset. Je passe cette même séquence à chaque classifier qui me retournent une probabilité sur son appartenance au label. Le classifier qui a la meilleure probabilité donne son label à la séquence.

Tests et résultats

Le meilleur score qu'on a pu obtenir en local se situait entre 45% et 50%. Sur la plateforme OPEGRA par contre, on arrivait à peine à 34%.
Pour améliorer les paramètres, j'ai créé la matrice de confusion qui m'a permis de mieux déterminer quels gestes étaient confondus et dont l'affinement des états devait être fait.
On peut constater que dans cette matrice les gestes les plus justes sont le 0, 2, 4, 6, 7 et 10. Au contraire du geste 3 qui n'est même jamais prédit et le 9 qui est tout le temps confondu avec le 10.

Afin d'augmenter mon score, j'ai essayé d'augmenter le nombre d'états mais malheureusement les résultats n'étaient pas meilleur. Du coup, je suis restée au nombre d'état 2 qui donnait le meilleur score actuellement.

		Predicted											
		0	1	2	3	4	5	6	7	8	9	10	
Actual	0	4											4
	1			4			1	3					8
	2		1	3		2		2		1			9
	3					5			2	2			9
	4					10			1				11
	5		1	1				7	1				10
	6		1			1		8					10
	7								9				9
	8			1		4		1	2	1			9
	9										1	10	11
	10		1									9	10
		4	4	9	0	22	1	21	15	4	1	19	100

Problèmes rencontrés

Les problèmes que j'ai rencontrés étaient au niveau de la plateforme OPEGRA. Lorsque je lançais mon algorithme, il tournait plus de 15 minutes alors qu'en local à peine 1 minute. C'est lorsque j'ai modifié mon échantillonnage que ça a fonctionné.
J'ai testé pas mal de choses avant de modifier cela parce que OPEGRA ne retourne pas de message d'erreurs sur l'algorithme en lui-même.

Réseau de neurones artificiels

Traitement des features

Avant de commencer à choisir les features il convient de déterminer comment elles seront considérées et traitées. En effet, certaines features peuvent être fondamentale avec un certain traitement comme elles peuvent devenir inutiles avec un autre.

Les données brutes à disposition étaient des représentations de capteurs sur la durée du mouvement. L'évolution de ces valeurs peuvent être considérés comme des signaux avec pour chaque geste une "forme" de signal différente. Si l'on pense de cette manière, on en vient très vite à penser à des méthodes d'analyse de signaux et notamment aux transformée de Fourier.

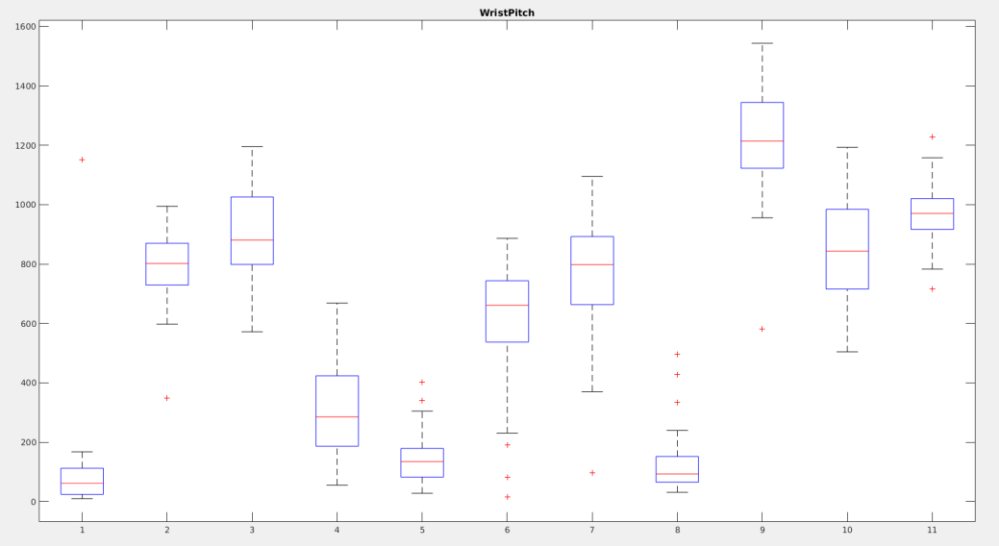
Ensuite, les données brut sont accessible (pour chaque occurrence de geste) sous forme matrices de N lignes et F colonnes pour N échantillons et F features. Le nombre d'échantillons pouvant varier d'une occurrence à une autre, il a fallu déterminer comment représenter chaque matrice en un vecteur de taille fixe qui servira alors de vecteur d'entrée (inputs) au réseau de neurones. Les transformée de Fourier, permettent une représentation intéressante du signal, mais ne résolvent pas cette problématique. Par contre comme elles font passer le signal du domaine temporel au domaine des fréquences, il peut être intéressant d'utiliser des fonctions d'analyse de dispersion, comme la moyenne, la médiane ou l'écart-type. En l'occurrence, l'écart type semble le plus approprié, car deux personne effectuant le même geste, ne le feront pas forcément à la même vitesse, ce qui va nécessairement changer les valeurs médianes et moyennes de la transformée de Fourier. Par contre l'écart type de la transformée de Fourier ne devrait pas beaucoup changer en fonction de la vitesse à laquelle le geste est effectué, mais uniquement en fonction de quel geste est effectué, ce qui est exactement ce que l'on cherche à discriminer.

C'est pour ces raisons que j'ai choisi de traiter les features en appliquant l'écart-type de la transformée de Fourier. Par la suite, en tentant d'optimiser l'algorithme, j'ai ajouté la moyenne de la transformée de Fourier, car cela améliorait tout de même les performances.

Choix des features

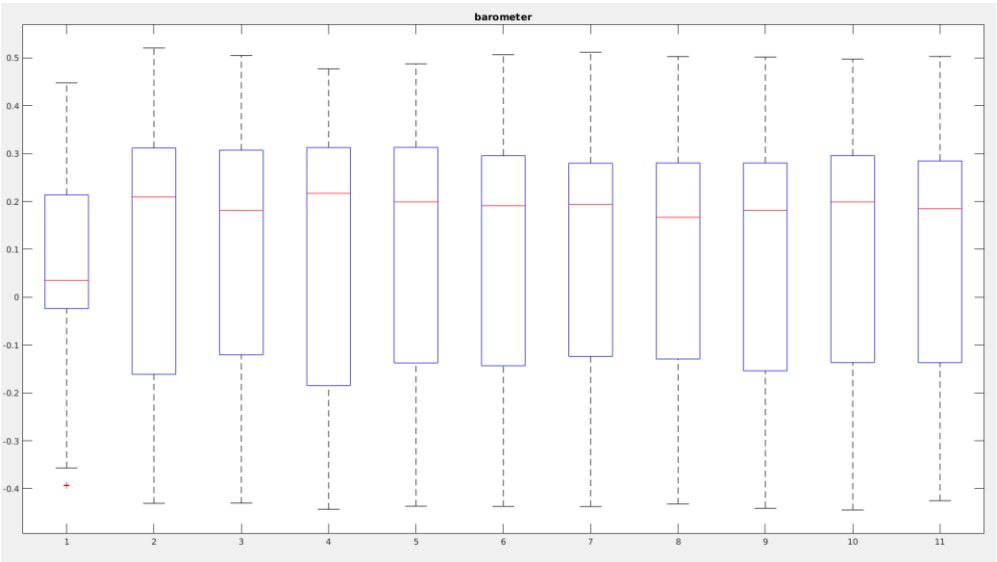
Afin de choisir les features, j'ai commencé par développé les fonctions MATLAB qui permettent d'afficher une dispersion des valeurs de chaque features pour chaque geste. J'ai ainsi créé un graphique par features qui permettent de comparer les dispersions de celle-ci pour chaque geste sous forme de boites à moustaches. Ces graphiques permettent de manière assez simple et rapide de se rendre compte de la pertinence des features.

Par exemple le graphique suivant montre les dispersions des écart-types des transformées de Fourier pour le "pitch" du poignet :



On constate que c'est une excellente feature, puisque les dispersions sont très différentes pour chaque geste et certains d'entre eux (comme le 7 par rapport au 8) peuvent même être discriminé uniquement avec cette feature.

Cet autre exemple permet de confirmer que le baromètre est une feature complétement inutile, puisque chaque geste présente les mêmes dispersions de valeurs :



Sur la base de ces analyses de dispersion, il a été possible de créer les trois groupes de features suivants :

- Les features à utiliser absolument
- Les features à ne pas utiliser
- Les features dont l'effet sur les performances est incertain.

Sur la base de ces trois groupes de features, j'ai utilisé l'algorithme suivant :

- features à utiliser = features indispensables
- features à essayer = features dont l'effet est incertain
- tant qu'il y a des features à essayer :
    - pour chaque features F à essayer :
        - entraine 5 de réseaux de neurones pour des subsets aléatoires* avec les features à utiliser + la feature F
        - evaluer les 5 réseaux de neurones entrainés
        - si le score moyen des 5 réseaux de neurones entrainés et le meilleur de cette itération :
            - définir la feature F comme meilleur feature à ajouter au set de donnée
            - conserver le meilleur score de cette itération
    - si le meilleur score de cette itération est meilleur que le meilleur score global :
        - ajouter la meilleure feature aux feature à utiliser
        - enlever la meilleure feature des features à tester
        - conserver le meilleur résultat global
    - sinon :
        - sortir de la boucle principale
- retourne les features à utiliser

* Les sets de données étaient à chaque fois séparés aléatoirement de la manière suivante :
64 % pour le set d'entrainement
16 % pour le set de validation (termine l'entrainement)
20 % pour le set de test utilisé lors de l'évaluation du score

Cet algorithme permet donc de détecter, parmi les features dont l'effet sur le résultat est incertains, quelles sont celles qui améliorent le plus le résultat. De plus, il "sait" s'arrêter lorsque ajouter des features n'améliorerait plus le résultat, mais au contraire le dégraderait. Ceci est très pratique, car s'il est relativement facile de faire la différence entre une excellente feature, et une très mauvaise feature, il est plus difficile de faire la différence entre deux features très similaire, et de savoir s'il serait plus intéressant d'ajouter quelques features ou d'en enlever.

Features selectionnées

Au final les features suivante ont été utilisées :

- Epaule :
    - Ecart-types de : XAcc
- Coude :
    - Ecart-types de : XAcc, YAcc, YawSpeed
    - Moyennes de : XAcc, YAcc, ZAcc
- Poignet :
    - Ecart-types de : YAcc, YawSpeed, Pitch, XMag, QuatX, QuatZ, QuatW
    - Moyennes de : Pitch, ZMag, QuatX, QuatW
- Main :
    - Ecart-types de : YAcc, YawSpeed, Pitch, XMag, ZMag, QuatX, QuatW
    - Moyennes de : Pitch, ZMag, QuatX, QuatW

Il est à noter que certaines moyennes des résultats des transformées de Fourier ont été ajoutées. Elles n'ont été ajoutées que parce qu'elles ont permis de meilleur résultats observés.

C'est bien sûr les features du poignet et de la main qui sont les plus utilisées, car c'est les parties qui bougent le plus et donc pour lesquelles les valeurs sont les plus distinctes et différente à chaque geste.

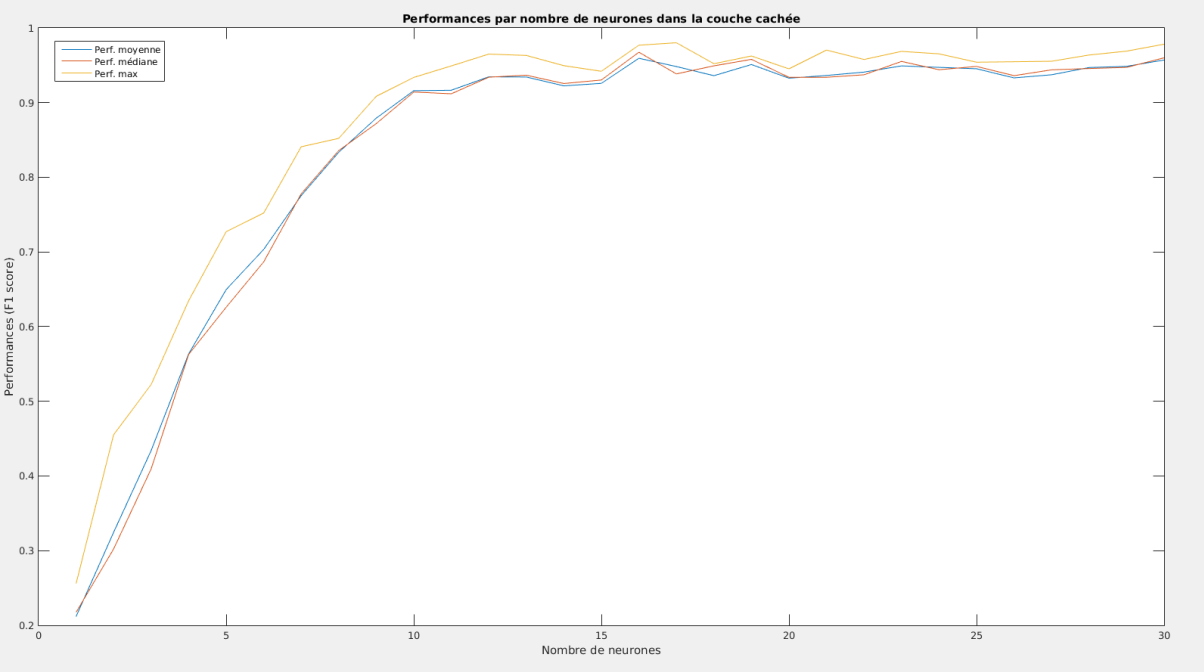
De plus aucune feature de la kinect n'a été selectionnée. Certaines d'entre elles avaient pourtant êtée placées dans le set de features dont l'effet sur les performances est incertain, mais l'algorithme de selection de features n'a pas trouvé qu'elles amélioreraient le résultat.

Enfin, de manière analogue l'écart-type de la transformée de Fourier des Accélérations en X pour l'épaule a été ajouté par l'algorithme de selection de feature, car il semble que cela améliore les résultats, même si l'épaule ne bouge pas beaucoup.

Apprentissage

Dans le cadre d'un réseau de neurones artificiels, il est nécessaire de déterminer combien de neurones mettre dans la couche cachée. Il faut en mettre le moins possible pour optimiser les performances en ressource (mémoire et temps d'execution) mais en mettre suffisamment pour que les résultats soient de bonne qualité. Afin de déterminer quel était le nombre de neurones optimal, j'ai créé une fonction MATLAB qui effectue 5 entrainements de réseaux de neurones pour chaque valeur nombre de neurones de la couche cachée entre 1 et 30.

Le graphique des performances (F1 score) en fonction du nombre de neurones dans la couche cachée est le suivant :



On remarque qu'entre 1 et 10 neurones dans la couche cachée, la performance du réseau s'améliore énormément à chaque neurone ajouté. Par contre, elle se stabilise ensuite et il ne semble pas pertinent de mettre plus de 15 neurones dans la couche cachée. C'est pourquoi j'ai choisi de mettre 11 neurones dans la couche cachée.

Après cela, tout est prêt pour entrainer le réseau de neurone final. Dans ce but, j'ai créé une fonction MATLAB qui entraine 200 réseaux de neurones différents avec l'entier du set de donnée (80% pour l'entrainement et 20% pour la validation). Chacun de ces réseaux sont évalué sur le set ayant servi à l'entraînement, et le meilleur d'entre eux est sauvegardé pour servir de classifier.

Résultats

Durant le développement, les résultats oscillèrent rapidement entre 94% et 98% (testé avec 20% du set de donnée n'ayant pas servi à l'entrainement). Le score final sur OPEGRA a donné les résultats suivants :

Fitness (f1 score) : 97.37

Matrice de confusion :

	Calibration	Swipe left	Swipe right	Push to screen	Take from screen	Palm-up rotation	Palm-down rotation	Draw a circle 1	Draw a circle 2	Wave hello	Shake hand	
Calibration	9	0	0	0	0	0	0	0	0	0	0	9
Swipe left	0	27	0	0	0	0	0	0	0	0	0	27
Swipe right	0	1	26	0	0	0	0	0	0	0	0	27
Push to screen	0	0	0	26	0	0	0	0	0	0	0	26
Take from screen	0	0	0	2	25	0	0	0	0	0	0	27
Palm-up rotation	0	0	0	0	0	23	4	0	0	0	0	27
Palm-down rotation	0	0	0	0	0	1	26	0	0	0	0	27
Draw a circle 1	0	0	0	0	0	0	0	27	0	0	0	27
Draw a circle 2	0	0	0	0	0	0	0	0	27	0	0	27
Wave hello	0	0	0	0	0	0	0	0	0	27	0	27
Shake hand	0	0	0	0	0	0	0	0	0	0	27	27
	9	28	26	28	25	24	30	27	27	27	27	

Les résultats obtenus sur OPEGRA furent conforme aux résultats obtenus localement. On remarque une légère confusion de reconnaissance entre le "swipe left" et le "swipe right", ainsi qu'entre le "push to screen" et le "take from screen". Il y a une plus grande confusion entre le "palm-up rotation" et le "palm-down rotation".

Les problèmes qui subsistent sont donc des problèmes de différenciation entre un geste et son geste inversé. Et il est normal que différencier le "palm-up" du "palm-down" soit plus difficile, puisqu'il y a moins de capteur et donc d'information qui entre en jeu pour ces gestes. Au vu des choix effectué pour traiter les informations, il est normal que le classifier confonde les gestes avec leurs inverses. En effet deux gestes qui serait parfaitement inversé devraient produire des résultats de la transformée de Fourier similaires. La raison pour laquelle les résultats sont tout de mêmes bons, c'est que dans la pratique les

mouvements inversés ne vont jamais être de parfait miroir de leur inverse, et le réseau de neurone arrive donc à “identifer” les subtiles différences qu’il y a et ainsi classifier correctement la plus grande partie d’entre eux.

Pour combler ce manque il pourrait être intéressant de séparer temporellement les données brut et (par exemple) prendre l’écart-type de la transformée de Fourier de la première moitié du mouvement séparément de la deuxième. Cela devrait permettre de diminuer la confusion sur les gestes inversés et donc d’améliorer les résultats globaux.

Conclusions

Pauline

Même si je n'ai pas obtenu le meilleur score, je suis satisfaite de ce projet. Cela m'a permis de bien réviser la matière du cours et pas seulement ce qui concernait mon algorithme mais les autres aussi; via mon collègue et les autres présentations. J'ai trouvé ce projet très intéressant car il m'a permis d'utiliser #C et de découvrir la librairie Accord.
Le principal problème que j'ai eu, était le temps à cause de tous les projets en même temps. Dans les améliorations futures, il faudrait faire tourner un algorithme qui va générer tous les paramètres possibles (nombre d'états, k-fold, etc) en récupérant le score. Et voir ceux qui génèrent le meilleur score.

Jonathan

Ce projet fut intéressant et a permis de mettre en évidence que tout se joue au niveau des features, et qu'il ne suffit pas de choisir celle-ci, mais également de bien définir comment les traiter et les considérer de manière cohérente avec l'algorithme d'apprentissage qui sera utilisé.

Effectuer ce travail dans le cadre d'un challenge a permis de montrer à la fois les avantages et les inconvénients de ces derniers. On notera l'envie de surpasser les autres que cela donne, et donc de dépasser ses propres limites et de faire preuve d'ingéniosité. Par contre on ne manquera pas de remarquer la baisse de collaboration que cela apporte dans la classe, puisqu'un challenge va limiter les partages entre les groupes.

Enfin, on ne manquera pas de noter le temps que prends l'optimisation des résultats de la solution par rapport au développement d'une solution non-optimisée. En effet, je suis assez rapidement supérieure à 93% en local, puis à 95% sur OPEGRA. Mais ce travail étant un challenge, il convenait d'essayer d'améliorer un maximum la solution. C'est notamment à cette fin que j'ai développé un algorithme de sélection des features. J'ai ainsi beaucoup travaillé pour atteindre au final que 97% sur OPEGRA, soit à peine 3% de mieux que la version non-optimisée. Ce travail est bien entendu gratifiant, car il a permis d'avoir de bons résultats, et a même été le deuxième meilleur réseau de neurones (parmi ceux rendu dans les temps).