Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene
Faculté d'électronique et d'informatique
Département d'informatique

Rapport de TP

Module : Apprentissage automatique et réseaux de neurones

Master 1 SII

Mini-projet

Traduction automatique des gestes de la main

- Réalisé par :

  BENHADDAD Wissam
  BOURAHLA Yasser

23-02-2018

Table des matières

1 Introduction

L'objectif de ce mini-projet est de nous familiariser avec les concepts et techniques d'apprentissage automatiques supervisé (plus particulièrement les réseaux de neurones) afin de mettre en pratique ces aspects théoriques, notamment en utilisant un modèle entrainé sur un ensemble de données pour résoudre un problème en particulier.

2 Problématique

Dans ce projet, nous avons choisis de tenter de réaliser une application qui permettra de traduire différents gestes de la main en un texte ou une action.

Cette traduction automatisé peut servir par exemple à

- Orientation à distance d'un robot.
- Traduction du langage des signes pour faciliter la communication avec les muets.
- **Air-Gesture** Communiquer une action à sa maison, son téléphone ou sa voiture avec la main ...

Il devient évident que réussir a traduire(avec un taux d'exactitude assez raisonnable) des geste de la main en temps réel(ou différé) s'avère être une tache irréalisable avec des algorithmes classiques [1], en raison de la complexité de la relation entre les données, cela nous a donc conduit à développer un module d'apprentissage automatique basé sur les réseaux de neurones pour accélérer l'aide à la décision, plus de détails dans MATENSASH TCITER HNA SECTION TA3 NEURAL NEEEETTTTTTTTTTTTT

3 Ensemble de données

Afin de développer le module d'apprentissage automatique mentionné dans précédemment (voir 2), nous avons choisit le data-set [2] disponible dans [**?**], il s'agira donc d'analyser ces données, de les pré-traiter éventuellement afin de les préparer pour la session d'apprentissage CITER FUCKING LEARNING HEEEEEERRE

3.1 Description des données

Comme expliqué dans [**?**], les données d'apprentissage on été récupéré a l'aide l'application Vicon-Tracker [**?**] ainsi que celle de marqueurs(au total de 11) placés sur la face arrière d'un gant, ces derniers servent de source de données envoyé à des capteurs positionnés sur les deux flancs(pour calculer la profondeur), la figure suivante illustre le procédé :

1. Algorithmes naïfs n'ayant pas recours à l'intelligence artificielle pour la résolution de problèmes
2. Ensemble de données d'apprentissage

FIGURE 1 – Environement ou les données on été capturées [**?**]

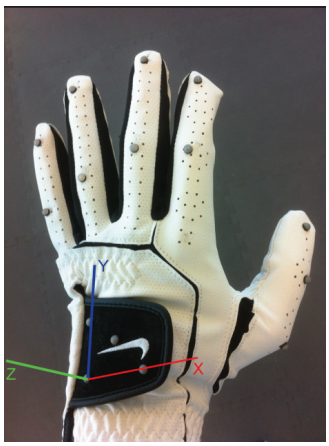


FIGURE 2 – Le gant utilisé pour y attaché les marqueurs(la source des données) [**?**]

Dans [**?**], on peut notamment trouvé une brève description des données brutes, elles sont sous forme d'un fichier **.csv** avec le délimiteur de cellules **,** (virgule), l'entête est structurée de la manière suivante :

- Les deux premières colonnes Class/User représentent répsectivement l'identifiant du geste observé (1 à 5) et l'identifiant de l'utilisateur(cobaye) qui a porté le gant durant cette session de collecte de données.
- Les 36 colonnes suivantes contiennent des nombres réels qui représentent les coordonnées cartésiennes en 3-dimensions (x_i, y_i, z_i) des différents marqueurs $Marker_i$.
  Il est a noté d'après [**?**] et [**?**] que les marqueurs sont non-étiquetés, c'est à dire que pour deux instances $I_1 et I_2$ du data-set, les coordonnés $(x_i^{I_1}, y_i^{I_1}, z_i^{I_1})$ et $(x_i^{I_2}, y_i^{I_2}, z_i^{I_2})$ ne désignent pas toujours les coordonnées du même marqueurs i.
  En raison des conditions de captage des données certaines instances ont des données manquantes représentées par **?**

3.2 Analyse et prétraitement des données

Une étape primordiale avant de se lancer dans les essaies d'apprentissage est l'analyse et la codification des données, nous avons d'abord effectué une analyse manuelle pour essayer de comprendre comment les données variaient, en nous inspirant des remarques faites dans [**?**] nous avons remarqué que durant l'enregistrent de la position des marqueurs, certains d'entre eux ne devenait plus visibles par les capteurs, fortement causé par la nature du geste et non pas a cause d'éventuelles perturbations, pour nous en assurer nous avons dresser le tableau suivant :

		Minimum number of Markers								
		4	5	6	7	8	9	10	11	12
Gestures	1	15604	14190	9191	2639	52	48	48	48	0
	2	14978	14950	14909	14761	14668	14550	13524	10524	31
	3	16322	15380	10746	6423	3556	81	0	0	0
	4	14774	14769	14578	13018	5767	2349	13	0	0
	5	15727	15686	15648	15406	14900	13535	10382	4180	0

TABLE 1 – Nombre d'instances par classe de geste dont le nombre minimum de marqueurs est visible (donnée non manquante dans le data-set)

Nous pouvons observer que par exemple, pour le geste 2 **tous les doigts levés** chaque instance étiquetée par cette classe possède quasiment tout les marqueurs visible à l'exception du dernier(puisque les marqueurs ne sont étiquetés on ne peut pas savoir le quel manque), il en est presque de même pour le geste 5 **(geste de saisie un sorte de poing ouvert légèrement)**, nous pouvons voir que qu'il y a une petite différence avec les données de la classe 2, cela est du au fait que les marqueurs placés aux ongles ne sont pas toujours visibles.

Pour le reste des classes, la classe 3 et 4 diffèrent en un seul marqueurs ou deux car les deux sont très similaires(**3** : un doigt levé, **4** : deux doigts levés), enfin pour les geste de la classe 1 **(Poing fermé)** le manque apparent de marqueurs visibles est expliqué par le fait que les marqueurs au jointure et extrémité des doigts soient masqués par la nature du geste en lui même.

Bien que les valeurs manquantes ont une grande signification pour nous humains, il faut tout de même traduire cette information pour le modèle, deux approches ont été pensée.

3.2.1 Approche naïve

Affecter une(des) valeur(es) aux données manquantes est une tâche souvent ardue, une multitude de théories existent sur le sujet, mais parfois la solution la plus simple semble être la plus efficace, nous avons simplement remplacé le valeurs manquantes par des **zéros**, ce choix est appuyé par un raisonnement fait à partir de l'analyse faite préalablement, en effet chaque classe de geste suit un certain motif(pattern) de valeurs manquantes, autrement dit si nous devions remplacer ses valeurs manquante par une approximation(une moyenne sur les valeurs de la colonne par exemple) cela pourrait rapprocher les instances entre elles et donc pourrait fausser l'apprentissage, car les valeurs des attributs seraient trop proches l'une d'entre elles, les remplacer par une valeur nulle pourra en théorie aider le modèle a mieux distinguer les classes, pour ainsi mieux approximer le pattern que suivent ces données.
Cependant, le pattern résultant d'un tel remplacement de données dépend surtout de la méthode dont les données ont été collectées. Comme expliqué précédemment, les données manquantes sont dû au fait que la méthode utilisée pour les collecter ne détectait pas tous les marqueurs, certains gestes éclipsaient certains marqueurs, d'où le pattern de données manquantes.

3.2.2 Approche par Clustering

L'approche par cluster essaye de minimiser la dépendance entre les données et la méthode avec la quelle elles sont collectées.1 L'idée c'est d'appliquer une sorte d'étiquetage de marqueurs en regroupant les coordonnées susceptibles d'appartenir au même marqueur dans un cluster. Nous avons utilisé un modèle de main 3D pour simuler les cinq gestes et collecter les coordonnées des marqueurs étiquetés.

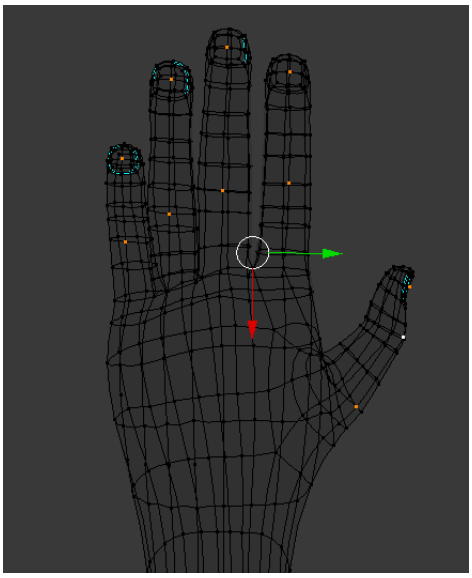


FIGURE 3 – Modèle de main 3D avec marqueurs étiquetés

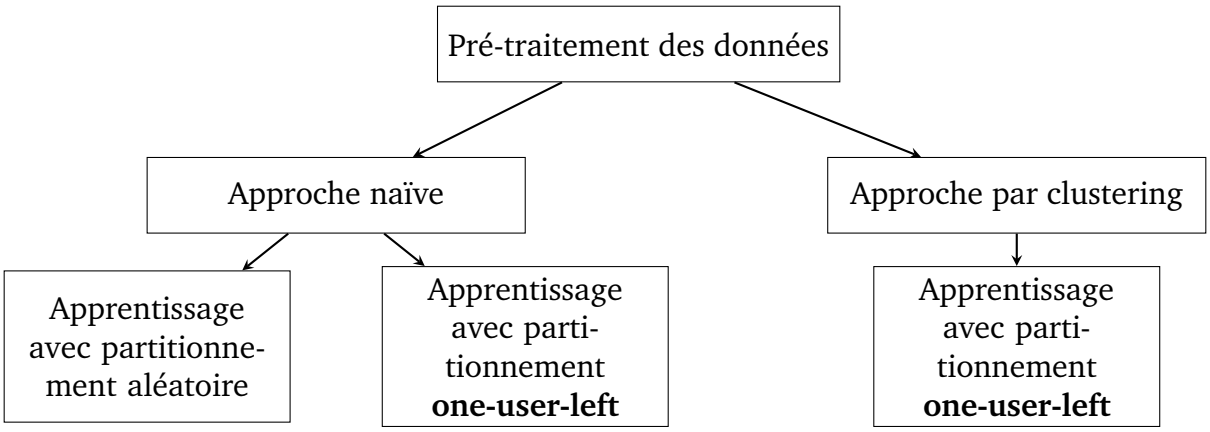
L'ensemble des données est ensuite parcourue en affectant chaque point (x_i, y_i, z_i) au marqueur le plus proche.
Le modèle 3D n'étant pas très fiable, ainsi que la différence entre les mains des utilisateurs génèrent un nombre important de collision, c'est à dire plusieurs point sont affectés au même marqueur. Pour y remédier, à l'affectation d'un point à un marqueur si ce dernier est déjà pris, on affecte le point au prochain marqueur le plus proche.
Évidemment à l'arriver d'une nouvelle donnée, soit en utilisant la même méthode de collection de données que le data-set d'apprentissage ou bien une autre méthode, le même traitement est effectué afin que cette donnée devienne conforme avec ceux qu'on génère en utilisant cette approche.

4 Solution proposée

Le problème ne pouvant être résolu à l'aide de techniques d'algorithmique classique(notamment à cause de sa trop forte complexité), le recours à l'apprentissage automatique s'est vu être la meilleure option.

4.1 Apprentissage sur les données

Après avoir analysée et traité les données, nous avons ensuite entamer la conception de notre modèle, étant donnée que nous avons utilisé deux approches pour le pré-traitement des donnés, la façon dont notre modèle va apprendre ces données pourrait différer, c'est pourquoi cette section sera diviser en deux sous-sections :



4.1.1 Codification

Pour ce qui est de la codification, nous avons opté pour une la codification un-parmis-N(dans notre cas N = 5) pour toutes les approches :

1	→	1	0	0	0	0
2	→	0	1	0	0	0
3	→	0	0	1	0	0
4	→	0	0	0	1	0
5	→	0	0	0	0	1

TABLE 2 – Correspondance entre geste$_i$ et codif(geste$_i$)

4.1.2 Variation des architectures

Pour trouver la meilleure architecture pour chacune des approches, nous avons écrit un script pour faire varier les différents paramètres, nous avons jugé bon de varier les suivants :

- Nombre de couches cachées, en effet comme vu en cours et en TP, se contenter d'une seule couche cachées peut s'avérer être un handicap pour le bon apprentissage de neurones, mais en rajouter trop peut nous conduire à un sur-apprentissage [3], nous avons donc fixé ce paramètre au maximum de 3.
- Nombre de neurones par couches cachées, aussi étroitement lié à la complexité du problème, ce paramètre joue un rôle clé dans la puissance d'apprentissage du modèle, en contre partie, il est sujet au phénomène de sur-apprentissage(trop de neurones dans une couche caché), mais aussi à celui de sous-apprentissage [4]
- Fonction d'apprentissage(Optimiseur), Élément central lors de la phase d'apprentissage, se contenter d'une ou deux fonctions ne serait pas très judicieux d'un point de vue scientifique, en effet il existe une multitude d'optimiseurs adapté a différents types de problèmes(SGD[**?**],Adadelta[**?**],RMSprop[**?**] ...)

4.1.3 Apprentissage avec partitionnement aléatoire

Danse cette approche, nous avons décidé de partitionné les donnés dans leur intégralité en 3 sous ensembles :

- A Ensemble des instances sur les quelles sera lancé l'apprentissage.
- V Ensemble des instances pour contrôler l'avancement de l'apprentissage.
- T Ensemble des instances qui servirons à l'évaluation de l'approximation fournie par le modèle après sont apprentissage

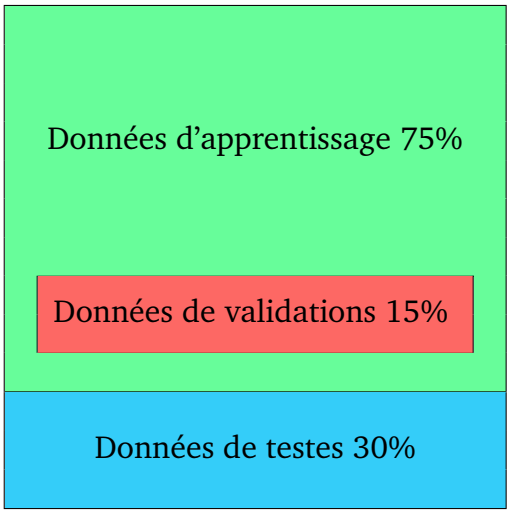


TABLE 3 – Partitionnement des données

3. Phénomène ou le modèle s'adapte beaucoup trop bien aux données d'apprentissage au dépens des données de testes

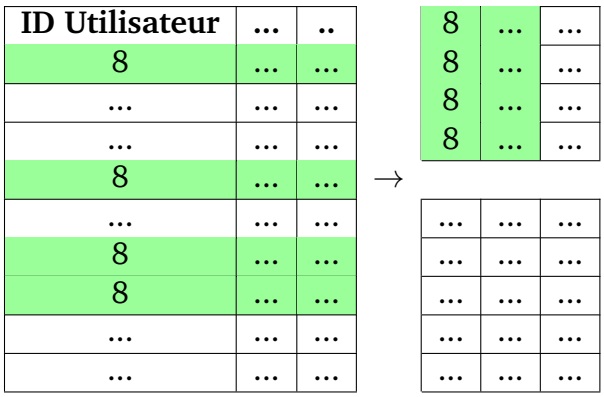
4. Phénomène ou le modèle n'arrive pas à apprendre la relation entres le données présentées durant l'apprentissage.

4.1.4 Apprentissage avec partitionnement one-left-user

L'idée (comme suggérée dans [**?**])est de séparer l'ensemble des instances D en deux sous ensembles **Train** et **Test** en fonction de l'identifiant d'une utilisateur id tel que :

- **Train** contient l'ensemble des instances ou $D[user] \neq id$
- **Test** contient l'ensemble des instances ou $D[user] = id$

Un utilisateur servira donc de testeur pour le modèle après apprentissage.



5 Comparaison entre des approches

Nous commençons d'abord par donner les tableaux récapitulatifs suivants :

Remarques :

- Tous les essayes se sont faite en 4 ré-apprentissages
- chaque couche de sortie dispose de la fonction d'activation **softmax**[5], en raison de la nature de la codification choisie.
- La liste des fonctions d'apprentissages utilisées au complet est cité dans SJHDSJHDJ-SHDHSDJHSHSHHD CITE ANNEX HERE

Architecture	Régression moyenne		
	Apprentissage	Validation	Évaluation
[40, 50, 50, 10] ['relu', 'relu', 'relu', 'relu'] : Nadam	0.9719	0.9538	0.9574
[40, 50, 50, 20] ['relu', 'relu', 'relu', 'relu'] : Nadam	0.9717	0.9614	0.9568
[40, 50, 50, 30] ['relu', 'relu', 'relu', 'relu'] : Nadam	0.9729	0.95705	0.9559
[40, 50, 20] ['relu', 'relu', 'relu'] : Nadam	0.9703	0.9515	0.9550
[40, 50, 50, 20] ['relu', 'relu', 'relu', 'relu'] : Nadam	0.9723	0.95113	0.9547

TABLE 4 – Meilleures architectures sur les données de teste pour l'approche naïve (3.2.1) avec partitionnement aléatoire(4.1.3)

5. Fonction mathématiques de normalisation `https://en.wikipedia.org/wiki/Softmax_function`

Commentaires : La première approche(voir 3.2.1) avec partitionnement aléatoire a donné d'assez bon résultats, il n'y a pas eu un sur-apprentissage apparent, les données de testes on été plutôt bien prédites, comme expliqué dans 3.2.1, la discrimination a été effectué tant bien que mal, principalement dû au motif(pattern) des valeurs manquantes qui dépend grandement des geste de chaque utilisateur, cette approche reste a être testé avec des instances non-présentes dans le data-set.

Architecture	Régression		
	Apprentissage	Validation	Évaluation
[40, 20] ['relu', 'relu'] : Adagrad	0.8948	0.8966	0.7918
[40, 20] ['relu', 'relu'] : Adagrad	0.8638	0.8639	0.7906
[40, 40] ['relu', 'relu'] : Adagrad	0.9057	0.8965	0.7797
[40, 10] ['relu', 'relu'] : Adadelta	0.9396	0.9365	0.7744
[40, 30] ['relu', 'relu'] : Adagrad	0.8856	0.8827	0.7708

TABLE 5 – Meilleures architectures sur les données de teste pour l'approche avec naïve (3.2.1) avec partitionnement one-user-left(4.1.4)

Commentaires : La même approche que précédemment (voir 3.2.1) mais avec un partitionnement one-user-left, malheureusement cette approche n'a pas donnée d'aussi bon résultats que la précédente, principalement dû au fait que chaque utilisateur avait une manière différente de réaliser les gestes demandés ainsi qu'au non-étiquetage des marqueurs, les données manquantes sont donc très différentes selon l'utilisateur, ainsi les données du testeur sont très mal prédites, ceci est un cas typique de sur-apprentissage.

Architecture	Régression		
	Apprentissage	Validation	Évaluation
[40, 20] ['relu', 'relu'] : Adadelta	0.9988	0.9968	0.9632
[40, 40] ['relu', 'relu'] : SGD	0.9990	0.9961	0.9620
[40, 50, 10] ['relu', 'relu', 'relu'] : Adadelta	0.9986	0.9977	0.9602
[40, 30] ['relu', 'relu'] : RMSprop	0.9982	0.9967	0.9577
[40, 10] ['relu', 'relu'] : Adam	0.9983	0.9963	0.9569

TABLE 6 – Meilleures architectures sur les données de teste pour l'approche avec Clustering (3.2.2) avec partitionnement one-user-left(4.1.4)

Commentaires : Les remarques qui sautent aux yeux sont bien entendu le très bon score à l'évaluation (meilleur que dans **??**) ainsi que le faible écart entre données d'apprentissage, de validation et de teste, malgré un partitionnement identique à celui de **??**, la différence faite par le clustering pour le ré-étiquetage des marqueurs et la gestion des valeurs manquantes a grandement aidé a donné un pattern similaires à ces valeurs pour chaque utilisateur, cette approche a donc couvert un peu plus l'espace de données possibles.

6 Application dédié

Nous avons opté à exploiter ce réseau de neurones avec une application mobile qui traduit le langage des signes en mots. L'application se connecte avec un gant qui contient des marqueurs. Ce dernier envoi à l'application les positions des marqueurs en utilisant des accéléromètres pour qu'elle puisse reconnaitre le geste effectué afin de le traduire.

Pour simuler une telle application, nous avons codé les lettres latins avec des combinaisons des cinq gestes reconnus par le réseau de neurones.

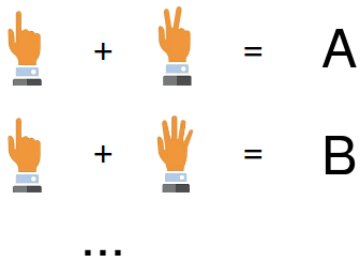


FIGURE 4 – Codage des lettres latins en utilisant les gestes

6.1 Schéma de l'application

L'application se devise en deux parties :

- Une pour simuler localement la traduction, c'est à dire elle utilise des données de testes déjà prêtes.
- L'autre recherche un gant sur le réseau, et récupère les données de ce dernier. Pour réaliser cette partie nous utilisons une application desktop pour simuler le gant afin d'envoyer les données à l'application.

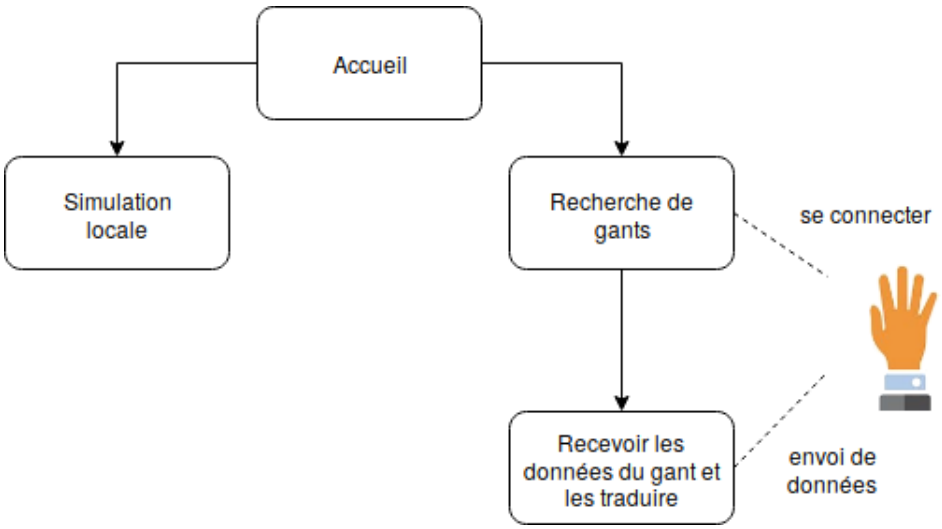


FIGURE 5 – Schéma générale de l'application

Table des figures

Liste des tableaux