Projet CLANU 2022-2023: Réseau de neurones appliqué à la classification d'images IRM

E. Bretin et C. Reichert

22 février 2023

Voici l'énoncé du projet CLANU. Ce projet a pour objectif de mettre en pratique une méthode présentée en analyse numérique dans un contexte de données réelles. Plus précisément, nous nous intéresserons cette année à l'utilisation d'une méthode d'apprentissage de type réseau de neurones généralisée au cadre multi-classes dans le but d'identifier types d'imagerie par résonance magnétique (IRM) du cerveau. Il s'agira plus précisément de reconnaître de façon automatique l'orientation (axial, coronal, sagittal) et le type d'acquisition IRM (T1,T2,PD), c'est à dire reconnaître 1 classe parmi 9.

Lors de ce projet, vous devrez :
- étudier une généralisation du code de réseaux de neurones vu lors du TP5 du module de MA1 dans le cadre d'une classification multiclasse
- mettre en forme une base de données à partir de données brutes d'IRM
- déterminer un réseau de neurones efficace, permettant de bien distinguer le type de pondération utilisé ainsi que la prise de vue tout en évitant des phénomènes de sur-apprentissage (situations où l'apprentissage intensif sur une base d'entraînement produit des résultats qui se dégradent sur des bases de données de validation ou de test).

Le contexte mathématique est introduit dans cet énoncé et différentes implémentations sont à faire pour répondre aux questions. La qualité des résultats et de l'analyse mathématique d'une part, ainsi que la qualité de la conception informatique d'autre part, seront prises en considération lors des différentes évaluation qui entour ce projet.

Concernant l'évaluation de ce projet, il s'effuera exclusivement lors d'un QCM moodle.

1 Base de données : imagerie par résonance magnétique

L'IRM permet d'étudier les propriétés des tissues biologiques de façon non-invasive. Nous nous intéresserons ici à des images du cerveaux. Ces images sont classiquement affichées sur l'écran d'un scanner en niveau de gris, comme illustré dans la figure ci-dessous. Plus précisément, nous affichons ici 9 images correspondant à chacune des classes considérées.

Le but du projet est d'apprendre via un réseau de neurones à reconnaître de façon automatique le type d'images fournie en entrée de l'algorithme. Plus précisément, vous disposez d'une base de

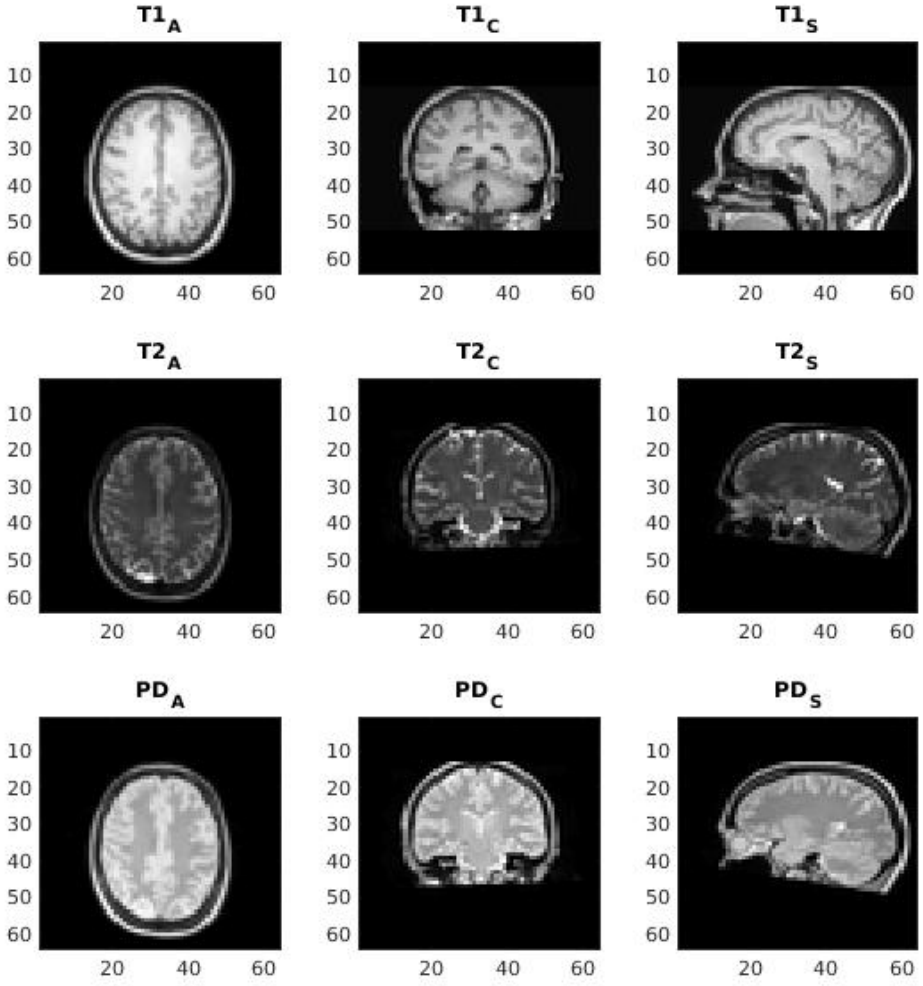

FIGURE 1 – Exemples d'images d'IRM réparties en 9 classes : T1A, T1C, T1S, T2A, T2C, T2S, PDA, PDC, PDS

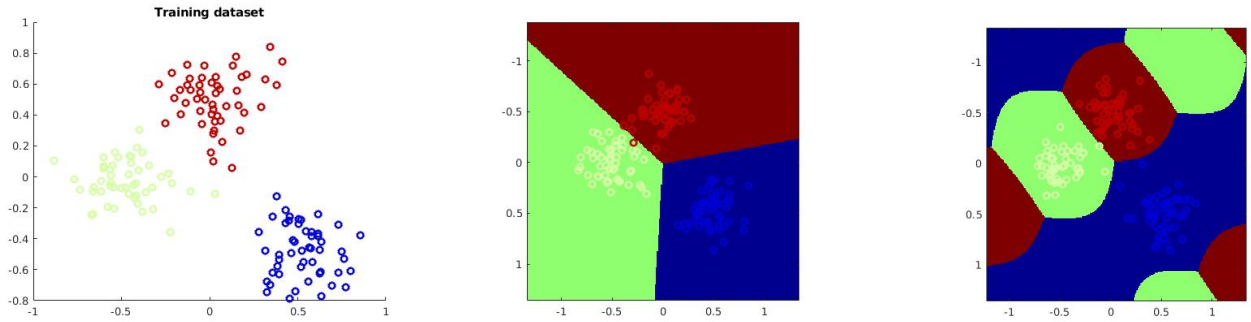

FIGURE 2 – Exemple d'un problème de classification avec 3 classes : données et 2 exemples de classification obtenues avec des structures différentes de réseaux de neurones

données constituée de 100 images de chacun des type T1A, T1C, T1S, T2A, T2C, T2S, PDA, PDC, PDS. Afin de pouvoir effectuer cet apprentissage en des temps raisonnables sous `Matlab`, toutes les images de la base de données ont été redimensionnées à une même taille fixe de 64x64 pixels. De plus, pour toutes les images, chaque pixel possède une valeur normalisée entre 0 et 1. Cette base de données vous est fournie via le fichier `data_Clanu_22.mat` présent sous *moodle*. Lors de la lecture de ce fichier sous `Matlab`, vous disposerez de 9 variables, ou sont sauvegardé les images IRM de chaque type sous forme d'un tableau de cellules. A partir de ces variables, vous pouvez simplement accéder à la $i^{ème}$ image pour chaque type de vue via les commandes `Matlab` suivantes `img = T1_A{i}` ...

2 Liste des questions liées au module MA2

2.1 Réseau de Neurones et classification multi-classe

Vous disposez d'un code `Matlab` fonctionnel qui est une extension du TP 5 et qui permet de traiter le cas de données avec un nombre de classes supérieur à 2.
En particulier le script `scrip_1_classification_R2.m` a été utilisé pour traiter le jeu de données illustré à la figure (2)

1. Etudier ce code en regardant
   — La structure des données database.X_train, database.Y_train
   — Les modifications du fichier model.m et notamment l'expression de la nouvelle fonction d'entropie à minimiser
   — Les modifications du fichier predict.m et la structure de ce que renvoie cette fonction.
2. Utiliser ce code pour classifier un autre exemple de jeu de données dans $\mathbb{R}^2$ réparties par exemple en 4 classe. Tester différentes structures de réseaux et différents choix du paramètre de descente.

2.2 Base de données d'IRM

1. Afficher quelques-unes de ces images et entraînez-vous à reconnaître visuellement le type d'images ainsi que son orientation. Afin d'afficher des images en niveau de gris, vous pouvez utiliser la commande `Matlab` suivante :
   `figure; imagesc(img); axis image; colormap(gray); colorbar;`
2. A partir du fichier `data_Clanu_22.mat`, écrire une fonction `structure_database` qui permet de séparer l'ensemble des images en trois groupes :
   — les données d'apprentissage (groupe `Train`) constituées de 80 images de chaque type, soit 2700 images au total ;

— les données de validation (groupe `Valid`) constituées de 10 images de chaque type, soit 90 images au total ;
— les données de test (groupe `Test`) constituées de 10 images de chaque type, soit 90 images au total.

Vous pourrez utiliser la fonction `Matlab randperm` pour sélectionner de façon aléatoire les images pour chaque groupe.

Pour chaque groupe, vous devrez créer une matrice dans laquelle seront stockées par colonne les images ré-organisées en vecteur colonne (`X_train`, `X_valid`, `X_test`) ainsi qu'un vecteur associé permettant de stocker la classe attachée à chaque image (`Y_train`, `Y_valid`, `Y_test`). L'expression d'une image sous forme vectorielle est réalisée simplement sous matlab via la commande suivante : `img(:)`. L'ensemble de ces données sera structuré via la variable unique `database` de la manière suivante :

— `database.X_train` $\Rightarrow$ matrice de taille $4096 \times 80 * nC$
— `database.Y_train` $\Rightarrow$ vecteur de taille $nC \times 80 * nC$
— `database.X_valid` $\Rightarrow$ matrice de taille $4096 \times 10 * nC$
— `database.Y_valid` $\Rightarrow$ vecteur de taille $nC \times 10 * nC$
— `database.X_test` $\Rightarrow$ matrice de taille $4096 \times 10 * nC$
— `database.Y_test` $\Rightarrow$ vecteur de taille $nC \times 90$

Ici nC représente le nombre de classes utilisées égale à 9 si on considère la base complète. Il sera cependant intéressant de faire varier le nombre de classes dans les différents tests en ne considérant qu'une partie de la base de données.

2.3 Données d'IRM et apprentissage du réseau de neurone

1. Tester le script précédent dans le cas d'un réseau à 1 couche sur la base de données d'IRM obtenue à partir de la fonction `structure_database`. On pourra commencer par essayer de classifier uniquement les images de type $T1$ avec $nC = 3$ puis considérer la base de données complètes avec $nC = 9$.
2. Afin d'éviter des phénomènes de sur-apprentissage (situations où l'apprentissage intensif sur une base d'entraînement produit des résultats qui se dégradent sur des bases de données de validation ou de test), il est intéressant d'afficher au cours du processus d'optimisation (c'est à dire au cours des itérations lors de la descente de gradient) à la fois l'énergie calculée sur les données d'entraînement mais également l'énergie calculée sur les données de validation (à partir des paramètres mis à jour en tenant compte uniquement des données d'entraînement). En effet, il est communément accepté que la situation de sur-apprentissage apparaît lorsque l'énergie diminue sur les données d'entraînement (processus d'apprentissage) alors que dans le même temps, l'énergie sur les données de validation augmente (perte de généralisation de l'algorithme au cours de l'apprentissage). Modifier la fonction `L_layers_nn.model` de votre programme afin de pouvoir afficher au cours des itérations un tel graphique.
3. A partir d'une étude structurée de l'influence de l'architecture de votre réseau (nombre de couches, nombre de neurones par couches, type de fonction d'activation, paramètres d'optimisation, etc...), quelle configuration vous semble la plus efficace afin de classifier le type d'orientation en IRM ? Il s'agira de trouver un réseau suffisamment performant tout en évitant des phénomènes de sur-apprentissage.