Université de Poitiers
M2 IOI, M2 info, M2 Math

2022–2023

Projet 2 : les réseaux de neurones

1 Développement d'un perceptron

Note : durant ce volet, le coefficient d'apprentissage α sera égal à 0.1.

1.1 Mise en place d'un perceptron simple

— *Créer une fonction*

```
[y]=perceptron_simple(x,w,active);
```

Ce programme doit évaluer la sortie d'un perceptron simple (1 neurone) pour une entrée élément de R^2.
Les paramètres :
- La variable `w` contient les poids synaptiques du neurone. C'est un vecteur à 3 lignes. La première ligne correspond au seuil.
- La variable `x` contient l'entrée du réseau de neurones. C'est un vecteur à 2 lignes.
- La variable `active` indique la fonction d'activation utilisée. Si `active==0` $\varphi(x)$ =sign(x) si `active==1` $\varphi(x) = \tanh(x)$

Le résultat :
- La variable `y` est un scalaire correspondant à la sortie du neurone.

— *Tester votre perceptron avec l'exemple du OU logique vu en cours (en utilisant $\varphi(x)$ =sign(x)).*

— *Afficher dans le cadre de l'exemple du OU logique sur la même figure les différents éléments de l'ensemble d'apprentissage et la droite séparatrice associée aux poids du neurone sur la même figure.* Par exemple sous Python, pour afficher une droite de type $y = ax + b$, par exemple pour $x \in [0, 1]$ on peut faire :

```
a=2 b=5 t=[-3,3] z=[a*t[0]+b,a*t[1]+b] plt.plot(t,z) plt.axis([t[0],t[1],z[0],z[1])])
```

1.2 Etude de l'apprentissage

1.2.1 Programmation apprentissage Widrow-hoff

— *Créer une fonction*

```
w,erreur=apprentissage_widrow(x,yd,epoch,batch_size);
```

Ce programme retourne le vecteur de poids w obtenu par apprentissage selon la règle d'apprentissage utilisant la descente du gradient.
Les paramètres :
- La variable `x` contient l'ensemble d'apprentissage. C'est une matrice à 2 lignes et n colonnes.
- La variable `yd[i]` indique la réponse désirée pour chaque élément `x[:,i]`. `yd` est un vecteur de 1 ligne et n colonnes de valeurs +1 ou -1 (classification à 2 classes).
- Epoch : le nombre d'itérations sur l'ensemble d'apprentissage.
- Batch_size : le nombre d'individus de l'ensemble d'apprentissage traités avant mise à jour des poids.

Le résultat :
- La variable `w` contient les poids synaptiques du neurone après apprentissage. C'est un vecteur à 3 lignes. La première ligne correspond au seuil.
- La variable erreur contient l'erreur cumulée calculée pour le passage complet de l'ensemble d'apprentissage à savoir

$$erreur = \sum_{i=0}^{N-1} (yd(i) - y(i))^2$$

La variable erreur est donc un vecteur de taille égale au nombre d'itération.

La droite séparatrice et les points d'apprentissage seront affichés à chaque itération (une itération correspond à la présentation de tous les individus de l'ensemble d'apprentissage), ainsi que l'erreur de classification. L'algorithme s'arrêtera dès que l'erreur est nulle, ou au bout de Epcoh itérations. Les poids initiaux seront générés de manière aléatoire.

1.2.2 Test 1 simple

— *Charger des données p2_d1.txt.*
`Data = np.loadtxt('p2_d1.txt')`
Ce fichier comprend une variable qui contient l'ensemble d'apprentissage constitué de 2 classes de 25 individus chacune en dimension 2. Un "professeur" nous a indiqué que les 25 premiers individus sont issus de la classe 1 et les 25 derniers de la classe 2.
— *Appliquer votre algorithme d'apprentissage. Afficher l'évolution de l'erreur. Vérifier que la frontière est correcte*

1.2.3 Test 2

— *Charger des données p2_d2.txt.*
Ce fichier comprend une variable qui contient l'ensemble d'apprentissage constitué de 2 classes de 25 individus chacune en dimension 2. Un "professeur" nous a indiqué que les 25 premiers individus sont issus de la classe 1 et les 25 derniers de la classe 2.
— *Appliquer votre algorithme d'apprentissage de Widrow-Hoff. Comparer avec les premières données et conclure.*

1.3 Perceptron multicouches

1.3.1 Mise en place d'un perceptron multicouche

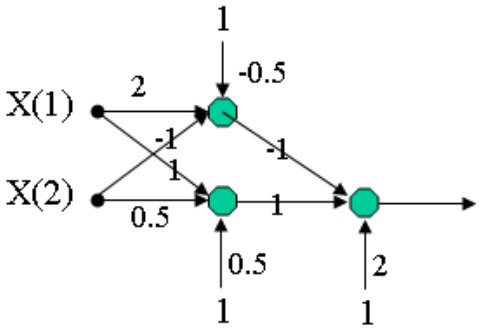
— *Créer une fonction*

```
y=multiperceptron(x,w1,w2);
```

Ce programme calcul la sortie d'un perceptron multicouches à 1 neurone sur la couche de sortie et deux neurones sur la couche cachée pour une entrée élément de R^2. La fonction d'activation est $\varphi(x) = \frac{1}{1+e^{-x}}$.
Les paramètres :
- La variable `w1` contient les poids synaptiques des 2 neurones de la couche cachée. C'est une matrice à 3 lignes et 2 colonnes. La première colonne `w1(:,1)` correspond au 1^{er} neurone de la couche cachée et `w1(:,2)` correspond au $2^{ème}$ neurone de la couche cachée.
- La variable `w2` contient les poids synaptiques du neurone de la couche de sortie. C'est un vecteur à 3 lignes.
- La variable `x` contient l'entrée du réseau de neurones. C'est un vecteur à 2 lignes.

Le résultat :
- La variable `y` est un scalaire correspondant à la sortie du neurone.

— *Tester votre perceptron multicouches avec l'exemple ci-dessous pour une entrée $x = [1\ 1]'$:*



Comparer le résultat "informatique" avec la sortie attendue calculée sur "le papier".

1.3.2 *Programmation apprentissage multicouches*

— *Créer une fonction*

```
w1,w2,erreur=multiperceptron_widrow(x,yd,Epoch,Batch_size);
```

Ce programme retourne les poids $w1$ et $w2$ obtenus par apprentissage selon la règle d'apprentissage utilisant la descente du gradient. Nous étudions le cas d'un perceptron multicouches à 1 neurone sur la couche de sortie et deux neurones sur la couche cachée pour une entrée élément de R^2. La fonction d'activation est $\varphi(x) = \frac{1}{1+e^{-x}}$.
Les paramètres :

- La variable `x` contient l'ensemble d'apprentissage. C'est une matrice à 2 lignes et n colonnes.
- La variable `yd(i)` indique la réponse désirée pour chaque élément `x(:,i)`. `yd` est un vecteur de 1 ligne et n colonnes de valeurs +1 ou 0 (classification à 2 classes).
- Epoch : le nombre d'itérations sur l'ensemble d'apprentissage.
- Batch_size : le nombre d'individus de l'ensemble d'apprentissage traités avant mise à jour des poids.

Le résultat :

- La variable `w1` contient les poids synaptiques des 2 neurones de la couche cachée. C'est une matrice à 3 lignes et 2 colonnes. La première colonne `w1(:,1)` correspond au 1^{er} neurone de la couche cachée et `w1(:,2)` correspond au $2^{ème}$ neurone.
- La variable `w2` contient les poids synaptiques du neurone de la couche de sortie. C'est un vecteur à 3 lignes.
- La variable erreur contient l'erreur cumulée calculée pour le passage complet de l'ensemble d'apprentissage à savoir

$$erreur = \sum_{i=0}^{N-1} (yd(i) - y(i))^2$$

La variable erreur est donc un vecteur de taille égale au nombre d'itération.

L'erreur de classification sera affichée à chaque itération (une itération correspond à la présentation de tous les individus de l'ensemble d'apprentissage). L'algorithme s'arrêtera dès que l'erreur est nulle, ou au bout de Epoch itérations. Les poids initiaux seront générés de manière aléatoire. Nous utiliserons comme coefficient d'apprentissage $\alpha = 0.5$.

Vous allez tester votre algorithme d'apprentissage pour le cas du XOR. La table de cette fonction est

$x(1)$	0	1	0	1
$x(2)$	0	0	1	1
$ydesire$	0	1	1	0

— *Créer l'ensemble d'apprentissage. Afficher cet ensemble avec la fonction affiche_classe. Pensez-vous que ce problème puisse être traité par un perceptron simple ?*
— *Appliquer votre algorithme d'apprentissage.*
— *Tester, à partir de votre fonction* `multiperceptron`*, le réseau de neurones ainsi obtenu sur l'ensemble d'apprentissage.*
— *Afficher les droites séparatrices associées aux différents neurones et les points de l'ensemble d'apprentissage.*

2 Deep et Full-connected : discrimination d'une image

L'objectif de ce mini-projet est de réaliser un système discrimination d'images réelles. La base de données d'images est une base de données classique de test (base Wang) et est composée de 10 types d'images : Jungle, Plage, Monuments, Bus, Dinosaures, Eléphants, Fleurs, Chevaux, Montagne et Plats.

Bien sûr à l'intérieur de chaque classe (100 images par classe), les images présent une forte variabilité. Je vous propose de mettre en place une application de classification. Dans le cadre du système de classification, notre système de base de données d'images fonctionnera de la façon suivante :

— l'utilisateur proposera au système une nouvelle image "inconnue" ;
— le système affichera alors la classe d'appartenance de l'image.

Afin de mettre en place cette, nous devons mesurer des paramètres sur chaque image afin de caractériser ces différentes images. Vous allez développer deux approches, l'une reposant sur des descripteurs "Image" l'autre suivant une stratégie Deep donc à base de descripteurs CNN.

2.1 Approche basée Descripteurs (basée modèle)

2.1.1 Calcul des descripteurs

Cette étape n'est pas forcement complexe mais nécessite du temps de développement et des connaissances en traitement d'images. Dans le cadre de ce projet, nous vous fournissons directement les mesures de caractéristiques préalablement faites sur les images de la base Wang dans un fichier Excel. Nous proposons d'utiliser 5 ensembles de mesures :

— JCD,
— PHOG,
— CEDD,
— FCTH,
— Fuzzy Color Histogram.

Pour chacun de ces vecteurs de mesures devant caractériser chaque image, nous fournissons l'article fondateur expliquant le principe de ces mesures. Elles vont mesurées des paramètres de couleur, de géométrie, de texture ... Nous faisons ces 5 types de mesures pour chacune des images de la base, ainsi nous obtenons 5 tableaux de mesures ayant un nombre de colonnes variable selon le descripteur et comprenant 1000 lignes pour les 1000 images. Les mesures sont stockées dans le fichier wang.xlsx. Ce fichier contient un onglet par type de mesure.

Attention à l'ordre de rangement des images : comme nous l'avons dit, la base est composée de 10 types d'images : Jungle, Plage, Monuments, Bus, Dinosaures, Eléphants, Fleurs, Chevaux, Montagne et Plats. Comme vous pouvez le constater en visualisant les images, les classes sont regroupées à travers leur numéro (par exemple de 200 à 299, ce sont les images "Monuments").

— *Importer les différents tableaux de mesure et créer un vecteur de label indiquant la classe sous forme d'un chiffre de chaque image : la classe de l'image de nom "i.jpg" est contenue dans label[i].*

2.1.2 Mise en place d'un système de discrimination basée structure Full-Connected

Nous proposons de mettre en place un système de discrimination qui doit déterminer à partir d'un ensemble d'apprentissage la catégorie de l'image inconnue. Vous allez extraire aléatoirement quelques images de la base de données ainsi que les mesures associées. Ces images tests vont constituer vos images "inconnues", le reste du tableau constitue l'ensemble d'apprentissage.

La structure de discrimination à mettre en place sera de type Perceptron Multi-couches. Pour cela, vous utilisez le framework Keras/TensorFlow, les hyperparamètres sont laissés libres. Pour cela, vous devez paramétrer votre environnement pour pouvoir utiliser Keras (qui a différentes dépendances) (conda install -c anaconda keras).

— Mettre en place un système de discrimination qui pour la présentation d'une image inconnue et de son vecteur de mesures associé propose une classe (Jungle, Plage, Monuments, Bus, Dinosaures, Eléphants, Fleurs, Chevaux, Montagne et Plats).
— Tester la procédure de classification avec les 5 types de mesures et différentes images inconnues. Analyser les résultats (matrice de confusion, taux d'erreur).
— Comparer avec différents hyperparamètres.

2.1.3 Approche "Deep" (basée Data)

Nous proposons de mettre en place une stratégie entièrement basée sur l'apprentissage, à savoir que les descripteurs vont être construits pas des couches de convolution.

— *Mettre en place un système de classification qui a en données une image calcul les descripteurs par des couches de convolution.*

Par exemple :

```
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(256,256, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

Nous appliquons ensuite une structure "Full Connected " pour prendre la décision donc pour la présentation d'une image inconnue propose une classe (Jungle, Plage, Monuments, Bus, Dinosaures, Eléphants, Fleurs, Chevaux, Montagne et Plats).

```
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
```

- *Dans un premier temps tester des structures simples. Etudier l'influence des paramètres, l'évolution de la fonction de cout.*
- *Comparer avec les résultats avec les méthodes basées caractéristiques.*
- *Tester avec des structures plus complexes. La "Data augmentation" doit probablement être utilisée.*