



**STATISTIQUE  
SCIENCE DES DONNÉES**  
UNIVERSITÉ DE MONTPELLIER

# Apprentissage Statistique

## TP n°5 : Perceptron multicouche

Rédigé par

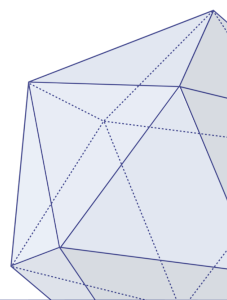
CÔME OLIVIER

PRALON Nicolas

SENE Assane

**IMAG**

INSTITUT MONTPELLIERAIN  
ALEXANDER GROTHENDIECK



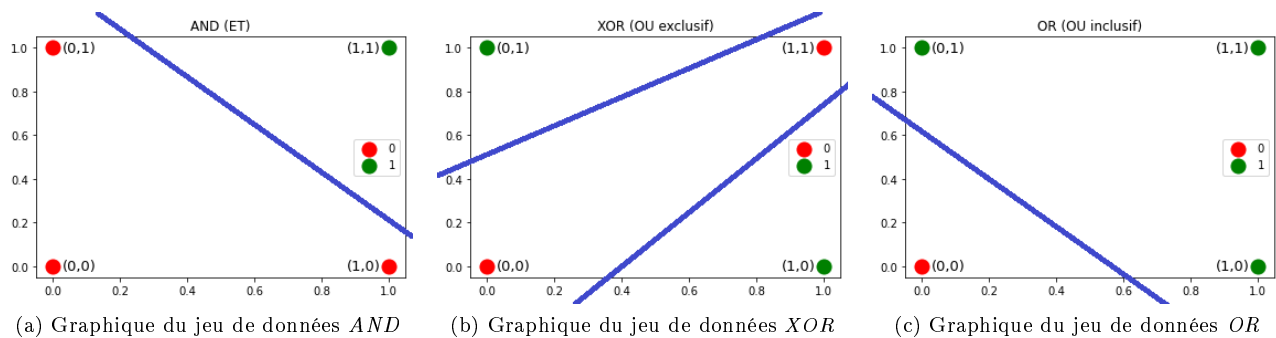
# Table des matières

Introduction	2
Question 1	2
Question 2	2
Question 3	2
Question 3	3
0.0.1 a) . . . . .	3
Question 4	3
0.0.2 b) . . . . .	3
0.0.3 c) . . . . .	3
Question 5	3
Question 6	4

# Introduction

## Question 1

Présentons très synthétiquement le jeu de données pour le *AND*, *XOR* et *OR* :



Les difficultés de classification résident dans le fait que les données du *XOR* ne sont pas linéairement séparables. Sur le graphique (b) on voit qu'il y a deux droites de séparation et non une seule. De ce fait, un seul neurone ne peut réussir à classer les données.

## Question 2

Définissons un classifieur *MLP* pour apprendre l'opérateur *AND* :

```
Score obtenu sur les données de test pour l'apprentissage de l'opérateur AND: 1.0
```

FIGURE 1 – Score obtenu par le classifieur *MLP* sur les données de test de l'opérateur *AND*

Dans le cas où l'on souhaite apprendre l'opérateur *AND*, le classifieur *MLP* ne comprenant aucune couche cachée fournit de très bons résultats en terme de prédiction. Comme on peut le voir sur la figure 1, le classifieur ne fait aucune erreur de prédiction (le score vaut 1). Pour calculer ce score nous avons utilisé la fonction `score(x_test, y_test)` de *sklearn*.

## Question 3

Définissons un classifieur *MLP* pour apprendre l'opérateur *OR* :

```
Score obtenu sur les données de test pour l'apprentissage de l'opérateur OR: 1.0
```

FIGURE 2 – Score obtenu par le classifieur *MLP* sur les données de test de l'opérateur *OR*

Dans le cas où l'on souhaite apprendre l'opérateur *OR*, le classifieur *MLP* ne comprenant aucune couche cachée fournit de très bons résultats en terme de prédiction. Comme on peut le voir sur la figure 2, le classifieur ne fait aucune erreur de prédiction (le score vaut 1). Pour calculer ce score nous avons utilisé la fonction `score(x_test, y_test)` de *sklearn*.

## Question 4

### 0.0.1 a)

Définissons un classifieur *MLP* sans couche cachée pour apprendre l'opérateur *XOR* :

```
Score obtenu sur les données de test pour l'apprentissage de l'opérateur XOR sans couche cachée: 0.5
```

FIGURE 3 – Score obtenu par le classifieur *MLP* sur les données de test de l'opérateur *XOR* sans utiliser de couches cachées

Dans le cas où l'on souhaite apprendre l'opérateur *XOR*, le classifieur *MLP* ne comprenant aucune couche cachée fournit de mauvais résultats en terme de prédiction. Comme on peut le voir sur la figure 3, la précision de prédiction du classifieur est de 50% (le score vaut 0.5), ce qui est assez bas.

Cela s'explique par le fait qu'un perceptron ne peut apprendre que sur des données linéairement séparables or ce n'est ici pas le cas.

### 0.0.2 b)

Définissons un classifieur *MLP* avec deux couches cachées pour apprendre l'opérateur *XOR* :

```
Score obtenu sur les données de test pour l'apprentissage de l'opérateur XOR avec les 2 couches cachées: 0.75
```

FIGURE 4 – Score obtenu par le classifieur *MLP* sur les données de test de l'opérateur *XOR* en utilisant 2 couches cachées

Le classifieur *MLP* comprenant deux couches cachées (4 neurones sur la première couche et 2 neurones sur la deuxième) fournit de meilleurs résultats (en terme de prédiction) que le classifieur ne comprenant aucune couche cachée. En effet, comme on peut le voir sur la figure 4, la précision de prédiction de ce nouveau classifieur est de 75% contre 50% pour le précédent.

Cela s'explique par le fait que l'ajout de couches cachées dans un perceptron permet de transformer un problème non linéairement séparable en un problème linéairement séparable. Ce nouveau classifieur (perceptron multi-couche) sera donc en mesure d'apprendre l'opérateur *XOR* contrairement au précédent.

### 0.0.3 c)

Classifieur *MLP* avec deux couches cachées et des fonctions d'activation hyperbolique pour apprendre l'opérateur *XOR* :

```
Score obtenu avec la fonction d'activation tanh: 1.0
```

FIGURE 5 – Score obtenu par le classifieur *MLP* sur les données de test de l'opérateur *XOR* en utilisant 2 couches cachées et des fonctions d'activation *tanh*

Les résultats obtenus ici sont encore mieux que ceux obtenus dans les deux cas précédents. En effet, comme on peut le voir sur la figure 5, la précision de prédiction de ce nouveau classifieur est de 100% contre 75% et 50% pour les deux précédents. Les fonctions d'activation hyperbolique (*tanh*) cherchent des séparateurs non linéaires or nos données sont ici non linéairement séparables. Par conséquent, ces fonctions d'activation seront bien adaptées à notre situation. Cela explique pourquoi les résultats obtenus ici sont mieux que ceux obtenus précédemment.

## Question 5

Notre jeu de donnée est composé d'images (de taille  $8 \times 8$  c'est à dire 64 pixels) de chiffre manuscrit. On veut identifier le chiffre contenu dans une image. On cherche donc ici à mettre en place un réseau de neurones qui sera capable de classer ces images. Étant donné qu'il y a 10 chiffres allant de 0 à 9, nous aurons ici un nombre total de 10 classes.

Nous allons comparer les performances de plusieurs classifieur que nous aurons crée en jouant sur différents paramètres de *MLPclassifieur*. Nous regrouperons ces performances dans un tableau puis nous regarderons quel classifieur fournit les meilleures performances.

```
In [5]: M
Out[5]:
```

	identity	logistic	tanh	relu
(lbfgs, ())	0.944444	0.955556	0.955556	0.944444
(lbfgs, 7)	0.966667	0.694444	0.861111	0.766667
(lbfgs, (4, 2))	0.733333	0.350000	0.355556	0.072222
(lbfgs, (3, 3, 2))	0.766667	0.222222	0.277778	0.516667
(sgd, ())	0.977778	0.972222	0.966667	0.966667
(sgd, 7)	0.966667	0.872222	0.788889	0.422222
(sgd, (4, 2))	0.666667	0.183333	0.377778	0.316667
(sgd, (3, 3, 2))	0.633333	0.072222	0.333333	0.127778
(adam, ())	0.966667	0.966667	0.977778	0.961111
(adam, 7)	0.966667	0.955556	0.883333	0.900000
(adam, (4, 2))	0.650000	0.161111	0.444444	0.344444
(adam, (3, 3, 2))	0.522222	0.166667	0.233333	0.338889

(a) Dataframe des scores des différents classifieurs utilisés

```
In [6]: M.idxmax()
Out[6]:
```

identity	(sgd, ())
logistic	(sgd, ())
tanh	(adam, ())
relu	(sgd, ())

dtype: object

(b) Meilleurs *MLP* en fonctions des différentes fonctions d'activations

FIGURE 6

D'après la figure 6, on constate que les classifieurs sans couches cachées sont les plus performant. Ce résultat est étonnant puisqu'ici, les données sont linéairement non séparables. Les classifieurs *MLP* ayant les meilleures performances sont ceux ayant pour fonctions d'activation la fonction tangente hyperbolique (*tanh*) et la fonction linéaire (*identity*). Étant donné que les données sont linéairement non séparables, on se serait plutôt attendu à avoir de meilleurs résultats avec la fonction d'activation *tanh* et l'optimiseur *adam*.

## Question 6

```
Score svm linear: 0.9722222222222222
Score svm polynomial: 0.9888888888888889
Score neuronal: 0.9722222222222222
```

FIGURE 7 – Comparaison entre le classifieur *SVM* et *MLP*

Dans cette dernière question nous comparons le classifieur *MLP* obtenu dans la question précédente avec deux classifieurs *SVM*. Afin de déterminer le meilleur de ces trois classifieur, nous évaluons les performances des deux classifieur *SVM* en validation croisé avec un k-fold ( $k = 5$ ).

D'après la figure 7, on constate que le classifieur *SVM* avec le noyau polynomial est le meilleur car c'est celui ayant le score le plus élevé.