ECOLE NATIONALE
SUPÉRIEURE
D'INFORMATIQUE

المدرسة الوطنية العليا للإعلام الآلي
(المعهد الوطني للتكوين في الإعلام العالي سابقا)
Ecole Nationale Supérieure d'Informatique
Ex.INI (Institut National de formation en Informatique)

Rapport TP FAS

Faire la classification et la régression sur les plantes de l'iris en utilisant les réseaux de neurones avec Tensor flow.

CHAOUCHE Yasmine - G01-

AZIZ Rezak - G03-

2020/2021

Table de matières

I. Introduction

Depuis une dizaine d'années, l'intelligence artificielle a connu un flagrant développement surtout avec l'apparition des réseaux de neurones. Ces derniers font l'objet de plusieurs travaux dans de nombreuses disciplines.

S'inspirant du cerveau humain, les réseaux de neurones donnent de meilleurs résultats en apprentissage. Ils sont appliqués pour résoudre des problèmes de classification, de prédiction, d'optimisation et de la reconnaissances : d'image, de voix, de visages, de texte .. et cela en s'appuyant sur des méthodes d'apprentissages profonds pour effectuer un ensemble de tâches qui sont de plus en plus complexes.

Notre étude s'intéresse à un problème particulier qui est le problème de classification, et cela en utilisant les réseaux de neurones. Ainsi, notre but est de faire une classification de données de fleurs d'iris à travers le développement d'un modèle d'apprentissage automatique avec les réseaux de neurones, et ensuite faire une évaluation de ce modèle.

Dans un souci d'organisation, nous avons séparé notre rapport en deux parties : une partie théorique qui consiste en une étude documentaire sur les réseaux de neurones et les outils à utiliser. Ensuite, une application pratique sur les fleurs d'iris.

II. Partie théorique

1. Le deep learning :

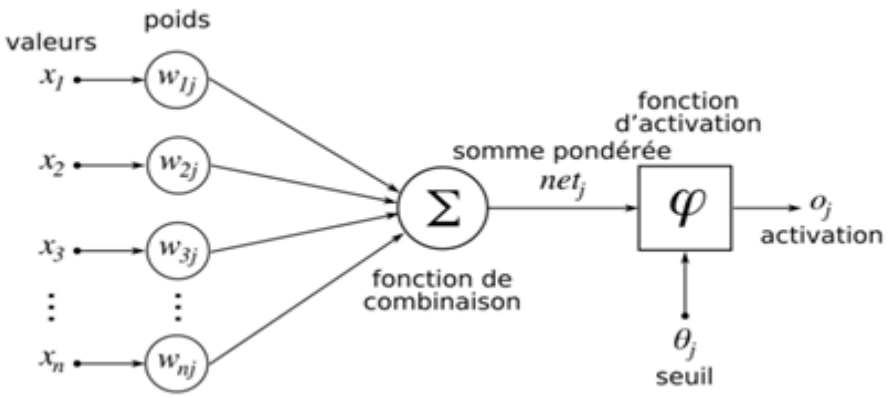
L'apprentissage profond ou deep learning est une branche du machine learning particulièrement adaptée à l'apprentissage de données complexes en vue de réaliser des modèles avancés supervisés ou non supervisés. Il permet à un réseau de neurones d'apprendre grâce à un grand nombre de couches permettant d'identifier des caractéristiques. De nombreuses couches sont dissimulées entre l'entrée et la sortie du réseau.

C'est quoi un réseau de neurones?

Les réseaux de neurones sont des imitations simples des fonctions d'un neurone dans le cerveau humain pour résoudre des problématiques d'apprentissage de la machine (Machine Learning)

Le neurone est une unité qui est exprimée généralement par une fonction sigmoïde : $f(x) = \frac{1}{1+e^{-x}}$

Un réseau de neurones peut prendre des formes différentes selon l'objet de la donnée qu'il traite et selon sa complexité et la méthode de traitement de la donnée. Voici l'architecture d'un réseau neuronal :

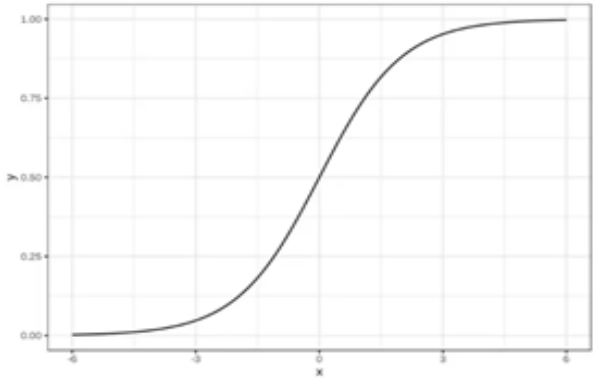


2. Régression logistique

La régression logistique est un modèle statistique permettant d'étudier les relations entre un ensemble de variables qualitatives Xi et une variable qualitative Y. Il s'agit d'un modèle linéaire généralisé utilisant une fonction logistique comme fonction de lien. Il permet aussi de prédire la probabilité qu'un événement arrive (valeur de 1) ou non (valeur de 0), donc le résultat varie toujours entre 0 et 1.

La fonction utilisée est donc la fonction sigmoïde, définie sur R à valeurs dans [0,1] et décrit par :

$$f(x) = \frac{1}{1 + e^{-x}}$$



Sigmoid function

3. Apprentissage automatique:

L'apprentissage automatique est un champ d'étude de l'intelligence artificielle qui se fonde sur des approches mathématiques et statistiques pour donner aux ordinateurs la capacité d'apprendre à partir de données, c'est-à-dire d'améliorer leurs performances à résoudre des tâches sans être explicitement

programmés pour chacune.

En général, deux grands types d'algorithmes d'apprentissage automatique sont utilisés aujourd'hui : l'apprentissage supervisé et l'apprentissage non supervisé:

- **Apprentissage supervisé**:

L'apprentissage supervisé s'intéresse aux données étiquetées.Des algorithmes tels que la régression linéaire ou logistique, la classification multiclasse et les machines à vecteurs de support sont des exemples d'apprentissage supervisé.

- **Apprentissage non-supervisé**:

L'apprentissage non supervisé traite des données non-étiquetées. L'objectif est d'identifier automatiquement des caractéristiques communes aux observations. Les méthodes de réduction de dimension, comme l'analyse en composantes principales, ou d'estimation de densités de probabilité font partie de l'apprentissage non supervisé.

III. Outils et techniques utilisés:

1. Tensor Flow:

Tensor flow est un outil open source d'apprentissage automatique développé par Google.

2. Keras:

Keras est une API pour l'apprentissage profond écrite en python. Elle permet d'interagir avec les algorithmes de réseaux de neurones profonds, notamment Tensor Flow.

Problématique: le problème de classification des fleurs d'iris.

Démarche:

- ➢ Recherche sur la classification et régression en utilisant tensor flow.
- ➢ Appliquer sur le dataset.

IV. Partie pratique:

1. Présentation de données

Le jeu de données des Fleurs d'iris comprend 150 observations, réparties en 3 espèces : setosa, virginica et versicolor. Il y a 50 observations de chaque type. Sur ces derniers, 4 caractéristiques sont mesurées qui sont: la longueur et la largeur du sépale et du pétale.

Dans un premier temps, on analyse les données qu'on a à notre possession:

5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5	3.4	1.5	0.2	Iris-setosa

2. Implémentation

Cette partie d'implémentation est divisée en trois étapes principales:

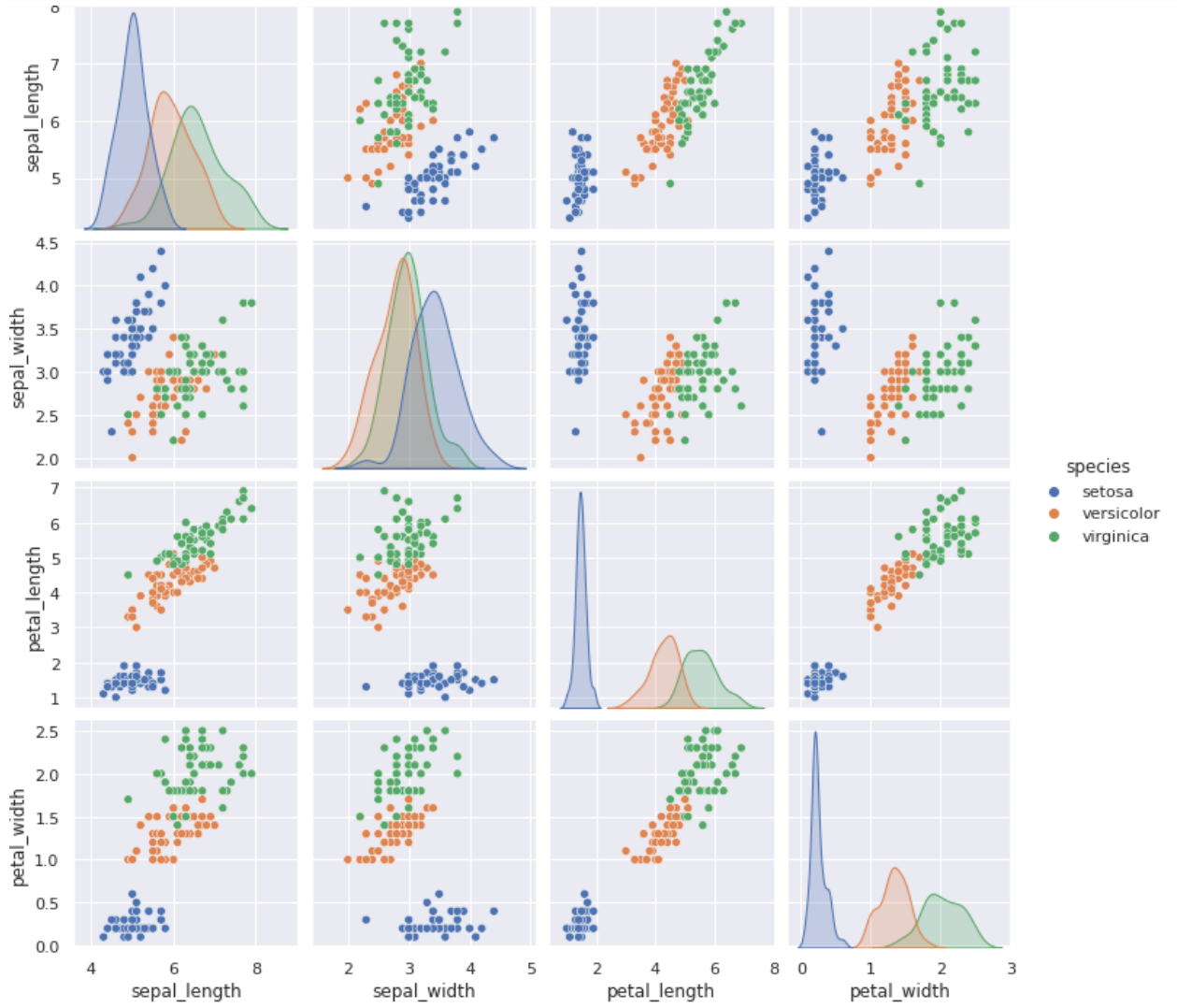
- charger les données d'iris (fichier .CSV) et les faire adaptées avec Keras.
- préparer les données de classification multi-classes pour la modélisation avec les réseaux de neurones.
- évaluer le modèle de réseau neurones Keras avec scikit-learn.

1. Visualisation des données

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import tensorflow as tf
import cProfile
tf.compat.v1.enable_eager_execution()
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
```

```
#importitt id depuis le menu a gauche
iris_data=pd.read_csv("iris.csv")
import seaborn as sns
sns.set()

iris = sns.load_dataset("iris")
iris.head()
sns.pairplot(iris, hue='species', height=2.5);
```



Après avoir représenté graphiquement les caractéristiques dans un diagramme de paires, il est clair que la relation entre les paires de caractéristiques d'un iris-setosa (en bleu) est nettement différente de celles des deux autres espèces.

2. Transformation des données

Dans un second temps, nous devons transformer nos données pour les adapter à l'API Keras et pour donner la même chance aux différentes classes de fleurs.

```
x
[[4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
 [4.8 3.  1.4 0.1]
 [4.3 3.  1.1 0.1]
 [5.8 4.  1.2 0.2]
 [5.7 4.4 1.5 0.4]
```

```
y
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
            ...
144    Iris-virginica
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
```

3. **Fractionnement des données:**

   Ensuite, les données doivent être divisé en deux partie: une partie qui sera utilisé pour entraîner notre modèle et une autre pour le test des performances de notre modèle

```
encoder = LabelEncoder()
y1=encoder.fit_transform(y)
Y=pd.get_dummies(y1).values
X_train, X_test, y_train, y_test=train_test_split(x,Y,test_size=0.2,random_state=0)
```

4. Choix d'un modèle:

   Dans notre cas, on doit utiliser un réseau de neurones pour cela on exploite l'API Keras qui nous permet de choisir un modèle et le paramétrer selon:

   - le nombre de Layer a utiliser
   - le nombre de noeud dans le Layer
   - La fonction d'activation des noeuds
   - La fonction de perte qu'on cherche a minimiser
   - la métrique à utiliser pour évaluer notre modèle
   - un algorithme d'optimisation pour notre modèle

```
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam
model=Sequential()
model.add(Dense(4,input_shape=(4,),activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(3,activation='softmax'))
model.compile(optimizer="Adam",loss="categorical_crossentropy",metrics=['accuracy'])
model.fit(X_train,y_train,batch_size=5,epochs=200)
```

- Il n'existe pas une méthode précise pour déterminer le nombre de Layers et de nœud à utiliser. Cela

se détecte par expérimentation et par intuition. Nous avons donc utilisé 3 nœuds dans le layer de sortie car nous avons 3 classes de fleurs et nous avons utilisé 4 nœuds dans le layer d'entrée. nous avons ajouté un autre layer pour avoir de meilleures performances.

- Pour la fonction d'activation des nœuds nous avons utilisé ReLU (Rectified Linear Unit). Cette fonction donne de meilleurs résultats que sigmoid. Pour le layer de sortie nous avons utilisé softmax est une généralisation de la fonction logistique qui prend en entrée un vecteur de K nombres réels et qui en sort un vecteur de K nombres réels strictement positifs et de somme 1. elle sera utile pour distinguer la classe ayant la plus grande probabilité.
- On utilise aussi l'algorithme d'optimisation qui s'appelle Adam. Ce dernier est une extension de la descente du gradient.
- Pour évaluer nos données pendant l'entraînement, on utilise la précision comme métrique et catégorical cross entropy comme fonction de perte.

```
24/24 [==============================] - 0s 1ms/step - loss: 0.1088 - accuracy: 0.9684
Epoch 194/200
24/24 [==============================] - 0s 1ms/step - loss: 0.0837 - accuracy: 0.9669
Epoch 195/200
24/24 [==============================] - 0s 1ms/step - loss: 0.0882 - accuracy: 0.9646
Epoch 196/200
24/24 [==============================] - 0s 2ms/step - loss: 0.1362 - accuracy: 0.9422
Epoch 197/200
24/24 [==============================] - 0s 1ms/step - loss: 0.1189 - accuracy: 0.9591
Epoch 198/200
24/24 [==============================] - 0s 1ms/step - loss: 0.1099 - accuracy: 0.9665
Epoch 199/200
24/24 [==============================] - 0s 2ms/step - loss: 0.0783 - accuracy: 0.9857
Epoch 200/200
24/24 [==============================] - 0s 1ms/step - loss: 0.0911 - accuracy: 0.9745
<tensorflow.python.keras.callbacks.History at 0x7f555186fd90>
```

Une fois notre modèle est entraîné on remarque qu'on a une perte de 0.0911 qui est une perte assez petite et une précision de 0.9745. On pourra dire que notre modèle donne d' assez bons résultats sur les données de l'entraînement. Il nous reste à confirmer cela sur les données de tests.

5. Evaluation de modèle:

```
z=model.predict(X_test)
y_pred=(z>0.5)
sklearn.metrics.accuracy_score(y_test,y_pred)
```

```
1.0
```

Pour évaluer notre modèle, on l'applique pour prédire les valeurs de tests, on remarque que notre modèle prédit les valeurs de tests avec une précision de 100%. Cela ne veut pas dire que notre modèle est parfait car la précision en utilisant toutes les données (entrainement + tests) ne donnent pas une précision de 100% mais on pourra dire que notre modèle est assez bon pour le généraliser.

V. Conclusion

En conclusion, ce TP a été une opportunité pour découvrir l'utilisation des réseaux de neurones et les appliquer dans un cas pratique. Ça nous a permis de toucher à des outils puissants pour la modélisation des réseaux de neurones et l'évaluation de nos résultats.

VI. Références

https://www.juripredis.com/fr/blog/id-19-demystifier-le-machine-learning-partie-2-les-reseaux-de-neurones-artificiels

https://datascientest.com/regression-logistique-quest-ce-que-cest

https://members.loria.fr/FSur/enseignement/apprauto/poly_apprauto_FSur.pdf

https://www.tensorflow.org/?hl=fr

https://keras.io/

Cours Machine Learning de Madame Benatchba et Monsieur Aries