Contenido

[REDES NEURONALES 1](#_Toc524447655)

[Keras 1](#_Toc524447656)

# REDES NEURONALES

## Keras

Keras es una plataforma de alto nivel para redes neurales escrita en Python. Esta plataforma esta enfocada en permitir una experimientación rápida de los datos de entrada. Keras soporta redes backpropagation, recurrentes, convolucionadas, entre otras y además combinaciones entre ellas; está diseñada para correr tanto en PCs como en computadoras avanzadas de multiprocesadores.

La API de Keras tiene dos modos de construir redes neuronales. El más simple es el modelo secuencial que solo permite que las capas se agreguen en secuencia. El modo Modelo, permite construir redes de manera mas flexible e incluso robusta.

$ pip3 install tensorflow

$ pip3 install keras

1. Importacion de librerías a usar

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

import numpy as np

import pandas as pd

import os

import matplotlib.pyplot as plt

import pylab as pl

from sklearn.preprocessing import RobustScaler

from sklearn.preprocessing import label\_binarize

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

import keras

from keras.models import Sequential

from keras.layers import Dense, Activation

from keras.optimizers import SGD

|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

1. Carga del archive de datos

seed = 155

np.random.seed(seed)

data = pd.read\_csv("Clasificacion.txt",sep="\t")

data.head(3)

1. Distinguir en un dataframe distinto el atributo de clase

y = data["Clase Vino"]

x = data.drop("Clase Vino", axis=1)

x = x.drop("ID Caso", axis=1)

y = label\_binarize(y, classes=[1,2,3])

n\_classes= y.shape[1]

1. Obtener los nombres de atributo

headers = x.dtypes.index

headers

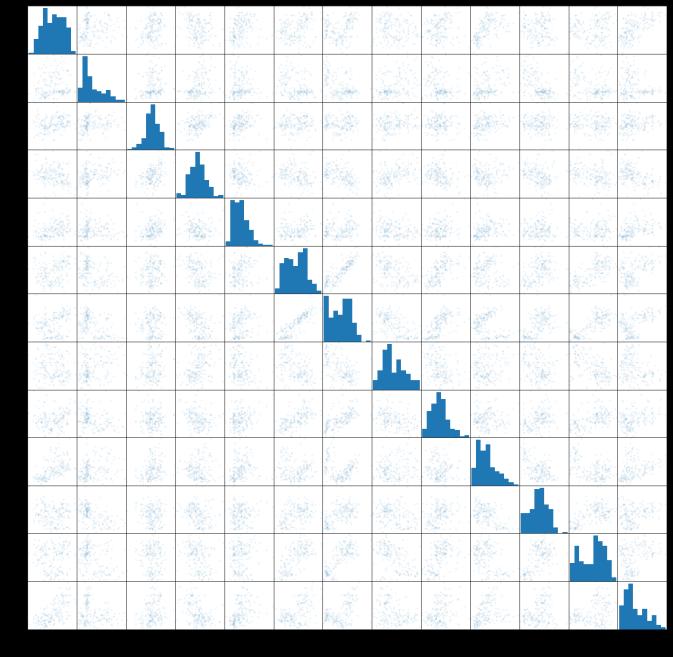
1. **Graficar los diagramas de dispersión**

**from pandas.tools.plotting import scatter\_matrix**

**fig, ax = plt.subplots(figsize=(20, 20))**

**scatter\_matrix(x[headers], alpha=0.1, diagonal='hist', ax=ax);**

**La grafica de salida es como la que sigue:**

****

1. Normalizacion

xt = RobustScaler(quantile\_range=(25, 75)).fit\_transform(x)

plt.scatter(xt[:,2], xt[:,4], s=40, c=y, cmap=plt.cm.Spectral)

1. Definicion de los conjuntos de entrenamiento y de test

X\_train, X\_test, y\_train, y\_test = train\_test\_split(xt, y,

test\_size=0.33,

random\_state=0

)

1. Creacion de la red neuronal

Definimos un modelo Secuencial. El modelo se construye apilando capas. La primera capa es una capa densa que recibe 2 entradas y tiene 20 neuronas con activación relu. La segunda y última capa tiene una neurona con activación lineal.

modelo = Sequential()

modelo.add(Dense(20, input\_dim=13))

modelo.add(Activation('relu'))

modelo.add(Dense(units=3))

modelo.add(Activation('linear'))

1. La estructura de la red neuronal puede ser visualizada mediante:

modelo.summary()

1. El proceso de aprendizaje se configura con la función .compile(). Aquí usaremos una función de pérdida de entropía cruzada y un optimizador SGD. A el proceso de aprendizaje hará un seguimiento de la precisión.

modelo.compile(loss='binary\_crossentropy',

optimizer = SGD(lr=0.01, momentum=0.9, decay=0.0, nesterov=False),

metrics=['accuracy'])

1. Entrenar la red neuronal

batch\_size = 32

epochs = 5

history = modelo.fit(X\_train, y\_train,

batch\_size=batch\_size,

epochs=epochs,

validation\_data=(X\_test, y\_test),

verbose=2)

1. Métricas

score = modelo.evaluate(X\_test, y\_test)

print('Test score:', score[0])

print('Test accuracy:', score[1])

(incluya tablas de confusion para cada clase)

Explique por favor lo que evidencian las siguientes graficas para la red neuronal::

pl.plot(history.history['loss'], label='loss')

pl.plot(history.history['val\_loss'], label='val\_loss')

pl.legend()

pl.xlabel('Epoch')

pl.ylabel('Loss')

pl.plot(history.history['acc'], label='acc')

pl.plot(history.history['val\_acc'], label='val\_acc')

pl.legend()

pl.xlabel('Epoch')

pl.ylabel('Accuracy')

Su objetivo es mejorar el rendimiento del clasificador de Keras. Estas son las cosas que debes probar:  
  
 Diferentes funciones de activación para la capa oculta (<https://keras.io/activations/>)

Diferentes optimizadores (<https://keras.io/optimizers/>)

Diferentes inicializadores para las capas densas (<https://keras.io/initializers/>)  
  
Pruebe diferentes combinaciones e informe sus hallazgos al final. ¿Qué configuración obtuvo la mejor precisión en la prueba?

Podria incluir tablas comparativas..