UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Estudios Superiores
Acatlán

Practica 5:

Redes Neuronales

Aprendizaje de maquina y minería de datos avanzados

Integrantes:

Gustavo Adolfo Alvarez Hernández

Profesor:

Eduardo Eloy Loza Pacheco

FES Acatlán, 19 de abril 2023

Objetivo

Que el alumno aprenda a diseñar redes neuronales utilizando Keras para tareas de clasificación, aprendiendo la estructura básica, los hiperparámetros que tiene cada capa, así como las funciones de perdida y optimizadores.

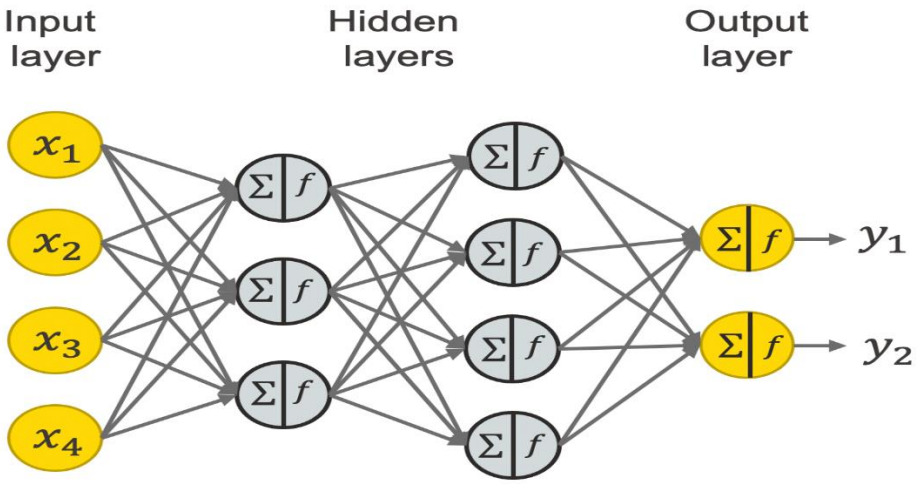
Resumen

En la presente practica se utilizará una red neuronal sobre el conjunto de datos churn que habla sobre personas que tienen una compañía, esto con el fin de realizar una clasificación y predecir si un usuario abandonara en este mes la compañía.

Antecedentes

Una red neuronal es una arquitectura diseñada para resolver tareas, la cual busca el hacer un simil sobre el funcionamiento del cerebro humano, al tener neuronas y al estar conectadas hacer sinapsis; Consta de 3 elementos principales los cuales son:

- Input layer:
  - La capa de entrada es la primera capa que solo consta de n neuronas para las n características que definen a nuestro dato, cada valor se aloja en cada neurona.
- Hidden layers:
  - Son aquellas capas interiores que se encargan de realizar operaciones sobre la capa de entrada, cada conexión contiene un peso y se genera una combinación lineal y es lo que recibe la neurona.
  - Luego esta pasa por una función de activación que determina si la neurona se activa o en términos generales pasa la salida de la función.
- Output layer:
  - La última capa tiene tantas neuronas como características tenga nuestra tarea a predecir, siendo donde se aloja la predicción.



En general ese es el funcionamiento y el trabajo esta en optimizar los pesos de cada capa para aproximarnos a la respuesta por los distintos métodos de optimización que existen.

Desarrollo

1. Cargamos las dependencias necesarias

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer

from keras.models import Sequential
from keras.layers import Dense

from sklearn.metrics import confusion_matrix
```

2. Cargamos los datos

```
data= pd.read_csv("Churn_Modelling.csv")
```

```
X= data.iloc[:,3:13].values
y= data.iloc[:,13].values
```

3. Generamos un One Hot Encoder para las variables categóricas

```
ls= ColumnTransformer([('Geography', OneHotEncoder(categories='auto'), [1])], remainder="passthrough")
X= ls.fit_transform(X)
X= X[:,1:]
```

```
ls= ColumnTransformer([('Gender', OneHotEncoder(categories='auto'), [3])], remainder="passthrough")
X= ls.fit_transform(X)
X= X[:,1:]
```

4. Dividimos los datos en entrenamiento y prueba

```
x_train,x_test,y_train,y_test= train_test_split(X,y, test_size=.20, random_state=0)
```

5. Escalamos los datos para tenernos en una proporción adecuada

```
sx= StandardScaler()
x_train = sx.fit_transform(x_train)
x_test = sx.fit_transform(x_test)
```

6. Generamos el modelo de clasificación con un optimizador Adam

```
clasificador = Sequential()
clasificador.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

   a. Añadimos las capas de entrada, salida y ocultas

```
clasificador.add(Dense(units=6,kernel_initializer='uniform',activation='relu',input_dim=11))
clasificador.add(Dense(units=6,kernel_initializer='uniform',activation='relu'))
clasificador.add(Dense(units=1,kernel_initializer='uniform',activation='sigmoid'))
```

7. Procedemos a entrenar el modelo

```
clasificador.fit(x_train,y_train,batch_size=10,epochs=100)
```

8. Generamos las predicciones

```
y_predice = clasificador.predict(x_test)
y_predBooleano = (y_predice>0.5)
```

```
63/63 [==============================] - 0s 2ms/step
```

9. Generamos la matriz de confusión para ver que tan bien clasifica el modelo

```
cm = confusion_matrix(y_test,y_predBooleano)
cm
```

```
array([[1530,   65],
       [ 213,  192]])
```

Conclusiones

Las redes neuronales para la clasificación son un modelo adecuado para la clasificación por su buena forma de optimizar los pesos, teniendo buenos resultados que en generalizan, esto es hay errores que son mínimos, pero no genere inquietud acerca del sobre ajuste.

Referencias

Knoklein, O. (5 de Junio de 2019). *Towards Data Science*. Obtenido de Classification Using Neural Networks: https://towardsdatascience.com/classification-using-neural-networks-b8e98f3a904f#:~:text=Neural%20networks%20are%20complex%20models,passing%20outputs%20to%20further%20layers.

Zhou, Z.-H. (2016). *Machine Learning.* Singapore: Springer Nature Singapore.