Red Neuronal

Una red neuronal es un modelo simplificado que emula el modo en que el cerebro humano procesa la información: Funciona simultaneando un número elevado de unidades de procesamiento interconectadas que parecen versiones abstractas de neuronas.

Las unidades de procesamiento se organizan en capas. Hay tres partes normalmente en una red neuronal : una capa de entrada, con unidades que representan los campos de entrada; una o varias capas ocultas; y una capa de salida, con una unidad o unidades que representa el campo o los campos de destino. Las unidades se conectan con fuerzas de conexión variables (o ponderaciones). Los datos de entrada se presentan en la primera capa, y los valores se propagan desde cada neurona hasta cada neurona de la capa siguiente. al final, se envía un resultado desde la capa de salida.

La red aprende examinando los registros individuales, generando una predicción para cada registro y realizando ajustes a las ponderaciones cuando realiza una predicción incorrecta. Este proceso se repite muchas veces y la red sigue mejorando sus predicciones hasta haber alcanzado uno o varios criterios de parada.

Al principio, todas las ponderaciones son aleatorias y las respuestas que resultan de la red son, posiblemente, disparatadas. La red aprende a través del entrenamiento. Continuamente se presentan a la red ejemplos para los que se conoce el resultado, y las respuestas que proporciona se comparan con los resultados conocidos. La información procedente de esta comparación se pasa hacia atrás a través de la red, cambiando las ponderaciones gradualmente. A medida que progresa el entrenamiento, la red se va haciendo cada vez más precisa en la replicación de resultados conocidos. Una vez entrenada, la red se puede aplicar a casos futuros en los que se desconoce el resultado.

Ejemplo

Se buscara clasificar el genero de un juego en base a los parametros de la plataforma, el publisher y sus ventas globales

Importar Librerias

In [20]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report
```

Cargar Datos

In [8]:
```
juegos = pd.read_csv('./vgsales.csv')
juegos = juegos.replace(np.nan,"0")
juegos['Platform'] = juegos['Platform'].replace("2600","Atari")

juegos.sample(10)
```

Out[8]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Otl
8429	8431	25 to Life	XB	2006.0	Shooter	Eidos Interactive	0.12	0.04	0.00	
10735	10737	MTX Mototrax	XB	2004.0	Racing	Activision	0.07	0.02	0.00	
14803	14806	Touch Shot! Love Application	PS3	2012.0	Adventure	Compile Heart	0.00	0.00	0.03	
12736	12738	Samurai Warriors 4: Empires	PS3	2015.0	Action	Tecmo Koei	0.00	0.00	0.06	
9764	9766	Ferrari Challenge Trofeo Pirelli	PS2	2008.0	Racing	System 3 Arcade Software	0.06	0.05	0.00	
15	16	Kinect Adventures!	X360	2010.0	Misc	Microsoft Game Studios	14.97	4.94	0.24	
9986	9988	NHL 2K3	XB	2002.0	Sports	Sega	0.09	0.02	0.00	
9314	9316	Red Orchestra 2: Heroes of Stalingrad	PC	2011.0	Shooter	Tripwire Interactive	0.04	0.07	0.00	
13928	13930	Shin Hayarigami	PS3	2014.0	Adventure	Nippon Ichi Software	0.00	0.00	0.04	
9694	9696	Blackwater	X360	2011.0	Shooter	505 Games	0.09	0.02	0.00	

Normalizacion

In [12]:
```
encoder = LabelEncoder()
juegos['plataforma'] = encoder.fit_transform(juegos.Platform.values)
juegos['publica'] = encoder.fit_transform(juegos.Publisher.values)
```

Definir Input/Output de la Red Neuronal

In [16]:
```
#Entradas a la red
X=juegos[['plataforma','publica','Global_Sales']]
#Salida
y = juegos['Genre']
```

Escalamiento

In [17]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y)
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

Crear y Entrenar Red Neuronal

In [18]:
```
mlp=MLPClassifier(hidden_layer_sizes=(10,10,10), max_iter=500, alpha=0.0001,
                     solver='adam', random_state=21,tol=0.000000001)

mlp.fit(X_train,y_train)
predictions=mlp.predict(X_test)
```

Revisamos la Red Neuronal

In [21]:
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

      Action       0.21      0.56      0.31       826
   Adventure       0.31      0.21      0.25       341
    Fighting       0.00      0.00      0.00       235
        Misc       0.07      0.01      0.01       419
    Platform       0.20      0.16      0.18       213
      Puzzle       0.00      0.00      0.00       148
      Racing       0.00      0.00      0.00       298
Role-Playing       0.20      0.13      0.16       388
     Shooter       0.24      0.02      0.04       320
  Simulation       0.00      0.00      0.00       211
      Sports       0.21      0.45      0.29       578
    Strategy       0.30      0.09      0.14       173

    accuracy                           0.22      4150
   macro avg       0.15      0.14      0.11      4150
weighted avg       0.16      0.22      0.16      4150
```

```
C:\Users\jeanc\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in
labels with no predicted samples. Use `zero_division` parameter to control this behav
ior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\jeanc\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in
labels with no predicted samples. Use `zero_division` parameter to control this behav
ior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\jeanc\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in
labels with no predicted samples. Use `zero_division` parameter to control this behav
ior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [28]:

```
accuracy= mlp.score(X_test, y_test)
print('Accuracy: ',accuracy)
```

```
Accuracy:  0.21686746987951808
```

In [ ]: