ARQUITECTURA NEURONAL MULTICAPA PARA TAREAS DE AGRUPAMIENTO

Dr. Álvaro Anzueto Ríos, Dra. Blanca Tovar Corona
Unidad Interdisciplinaria en Ingeniería y Tecnologías Avanzadas
Instituto Politécnico Nacional CDMX México
aanzueto@ipn.mx, bltovar@ipn.mx

Anzueto, A. & Tovar, B. (1 de noviembre de 2023). Arquitectura neuronal multicapa para tareas de agrupamiento. Boletín UPIITA. 18 (99).

Resumen

La unidad neuronal artificial más sencilla es el perceptrón y básicamente es la representación numérica, simplificada, de una sola neurona biológica, que es empleada típicamente en la tarea de clasificación. Una arquitectura neuronal multicapa es nombrada así cuando se conectan múltiples unidades perceptrón, en serie, y en conjunto aumentan su capacidad para la solución de problemas más complejos, como lo es el agrupamiento de datos. En este trabajo se presenta una metodología para realizar el agrupado de datos y se emplea una arquitectura neuronal perceptrón multicapas, considerando una base de datos que contiene la información referente a tres tipos de vinos. Cada dato de entrada a la arquitectura neuronal es el resultado del análisis químico y de ello se han extraído 13 características. En la gráfica final, cada uno de los tres ejes corresponde al valor numérico de cada neurona en la capa intermedia o de "latenciaz se puede observar que la tarea de agrupar de los datos es realizada.

1. Introducción

El agrupamiento de datos es una terea compleja que ha sido abordado por técnicas de aprendizaje automático en máquinas, ayuda a comprender el comportamiento de eventos, de donde se registran muestras formadas por características o rasgos, y de ellos se pueden extraer o identificar patrones, tendencias, grupos de datos que comparten características similares o bien datos atípicos; pero todo ello sirve para analizar e interpretar un objetivo mayor como puede ser la toma de decisiones.

En este trabajo se ha considerado la base de datos "WINE"[1], la cual contiene, de manera general, el análisis químico de tres tipos de vinos cultivados en la misma región de Italia por tres casas diferentes. En la Tabla 1, se listan las 13 características consideradas y la cantidad de muestras para cada clase. Con esta tabla de datos se busca determinar cada tipo de vino.

Neurona Artificial Perceptrón: Las Redes Neuronales Artificiales (RNA) están compuestas de un conjunto de elementos de procesamiento (EP) muy simples que emulan a las redes neuronales biológicas y las conexiones entre ellas [2-3]. El perceptrón (ver Figura 1), es la representación más básica de una neurona biológica y su ecuación o EP está representada en (1).

En la Figura 1, se presenta la arquitectura básica de un perceptrón con múltiples entradas y una salida o EP. Esta arquitectura consiste de un grupo de nodos de entrada conectados a un nodo suma por medio de elementos de ponderación (pesos sinápticos w) y un valor de polarización que está representado por b. La función de activación f determina al valor numérico a la salida que corresponde a las entradas procesadas por el perceptrón.

$$a = f(W \cdot P + b) \quad \ldots \quad (1) \tag{1}$$

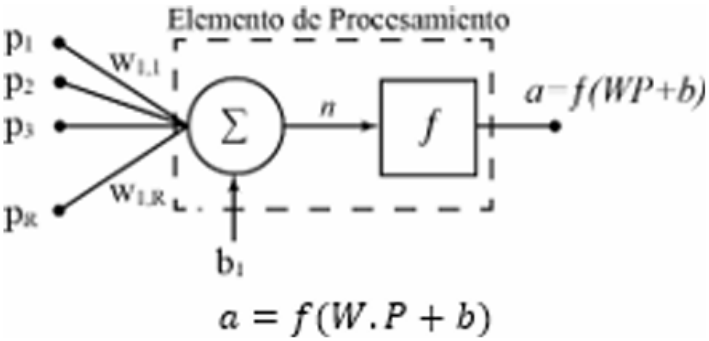

Figura 1 Neurona Perceptrón.

Una arquitectura de red neuronal multicapa se forma al colocar en cascada al menos dos neuronas, es decir, la salida de la primera capa son los datos de entrada de la segunda. Usualmente, las neuronas que se encuentran en la misma capa tienen la misma función de activación y el mismo patrón de conexiones hacia otras neuronas. En la Figura 2, se presenta una arquitectura multicapa.

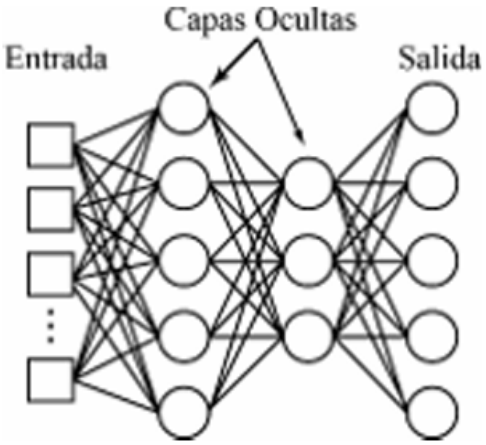

Figura 2 Neurona Perceptrón (Arquitectura multicapa).

Cuadro 1 Características de la base de datos Vinos.

Características	Número de Muestras
Alcohol, Ácido málico, Ceniza, Alcalinidad de la ceniza, Magnesio, Fenoles totales, Flavonoides, Fenoles no flavonoides, Proantocianinas, Intensidad de color, Matiz, OD280/OD315 residuo de vinos diluidos, Prolina	class_0 con (59) muestras, class_1 con (71) muestras, class_2 con (48) muestras

2. Desarrollo

Con la finalidad de desarrollar el proceso de agrupamiento se ha considerado el lenguaje de programación Python [4], además de emplear las bibliotecas "sklearn"[5] para la implementación de la arquitectura neu-

ronal y la extracción de la base de datos. Como primer paso, los datos son procesados y cada una de las características, que conforman una muestra, son normalizadas en un rango numérico de 0 a 1, evitando así, que una de las características influya en mayor medida que el resto.

Como siguiente paso, se subdivide la base de datos (ver comando `train_test_split`), para generar los apartados de entrenamiento, prueba y validación que son necesarios para el proceso de entrenamiento de la red neuronal. La Figura 3, contiene las líneas de código de los procesos que se han descrito hasta el momento.

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

from sklearn.neural_network import MLPRegressor
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_wine, load_iris
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
#<=========================================================
plt.close('all')

wine =  load_wine()
# wine = load_iris()
xx = wine.data
yy = wine.target
target_names = wine.target_names

scaler = MinMaxScaler()
scaler.fit(xx)
# separación de los datos para entrenamiento y validación
x_train, x_test, y_train, y_test = train_test_split(inputs, yy,
                                                    test_size=0.2,
                                                    random_state=1)
```

Figura 3 Líneas de código para la lectura y normalización de la base de datos.

Entrenamiento de la Arquitectura de Red Perceptrón Multicapa: Ahora es necesario definir la arquitectura neuronal. Se ha considerado la propiedad de una red completamente conectada, es decir, las salidas de una capa neuronal son todas conectadas a las neuronas que la prosiguen. Las líneas de código de la Figura 4, inician con este proceso.

La arquitectura neuronal multicapa tiene como parte inicial las 13 características que conforman una muestra, dos capas intermedias de 30 y 10 neuronas respectivamente, una capa profunda nombrada como capa de latencia (3 neuronas), de donde, cada neurona es considerada como un eje para un gráfico en tres dimensiones. Este gráfico sirve como representación del proceso del agrupado de los datos, se considera que la capa de latencia contiene la información esencial de cada muestra y por ende, las muestras con características similares tenderán a agruparse y diferenciarse del resto.

Finalmente se tiene dos capas de salida de 10 y 30 neuronas, con las cuales se reconstruye el mismo dato de entrada, es decir, el "targetü objetivo de la red, es presentar como resultado nuevamente el dato de entrada, esto indica, que la red neuronal en la capa de latencia extrae las componentes principales del dato de entrada y se comprueba, dado que, con esta información es posible reconstruir el mismo dato a la salida. El comando de entrenamiento de la red neuronal (`reg.fit`), necesita como parámetros el dato de entrada y el objetivo, que para nuestro caso será el mismo (`x_train`).

```
# Dimension de los datos de entrada
n_input = 13

# Estructura de entrada de la red (Encoder)
n_encoder1 = 30
n_encoder2 = 10

# Capa intermediam también nombrada capa de "LATENCIA"
# Capa para generar los gráficos en 3D (Neurona 1 = eje x)
n_latent = 3

# Estructura de salida de red (Decoder)
n_decoder2 = 10
n_decoder1 = 30

reg = MLPRegressor(hidden_layer_sizes = (n_encoder1, n_encoder2,
                                         n_latent,
                                         n_decoder2, n_decoder1),
                   activation = 'tanh',
                   solver = 'adam',
                   learning_rate_init = 0.0001,
                   max_iter = 9000,
                   tol = 0.0000001,
                   verbose = True)

reg.fit(x_train, x_train)
```

Figura 4 Líneas de código para la construcción y entrenamiento de la red neuronal.

3. Resultados

Como parte del entrenamiento se ha ejecutado el código con 9000 épocas de entrenamiento, con la búsqueda de un error mínimo de 0.001 y una función de actualización de tipo .ªdam". Las líneas de código en la Figura 5 ejecutan el proceso de extracción de datos en la capa de latencia y este resultado es graficado y presentado en la Figura 6. Es importante denotar que se logra la agrupación en tres clases, que corresponden a los tres tipos de vino.

```
#<-------------------------------------------------------------------------
def decoder(new_data):
    new_data = np.asmatrix(new_data)
    decoder2 = new_data*reg.coefs_[3] + reg.intercepts_[3]
    decoder2 = (np.exp(decoder2) - np.exp(-decoder2))/(np.exp(decoder2) + np.exp(-decoder2))

    decoder1 = decoder2*reg.coefs_[4] + reg.intercepts_[4]
    decoder1 = (np.exp(decoder1) - np.exp(-decoder1))/(np.exp(decoder1) + np.exp(-decoder1))

    reconst = decoder1*reg.coefs_[5] + reg.intercepts_[5]
    reconst = (np.exp(reconst) - np.exp(-reconst))/(np.exp(reconst) + np.exp(-reconst))

    return np.asarray(reconst)
#<-------------------------------------------------------------

test_latent = encoder(inputs)
plot3clusters(test_latent[:,:3], 'Linear AE', 'AE')
```

Figura 5 Líneas de código para la extracción de los datos en la capa de latencia.

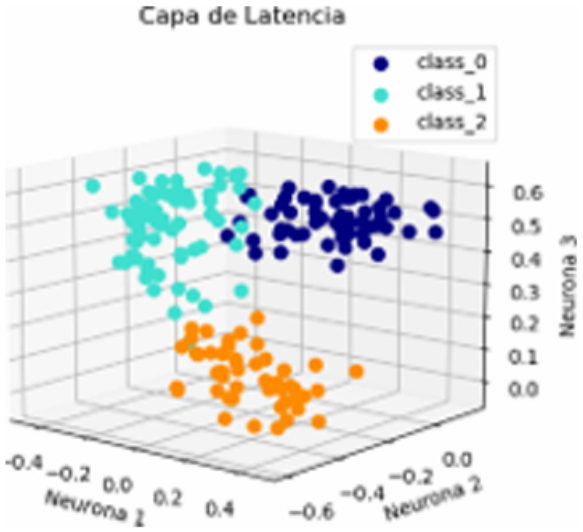

Figura 6 Representación de las tres clases de salida obtenida de la capa de latencia.

4. Conclusiones

La tarea de agrupado de datos nos puede explicar las tendencias, patrones y los que sucede en contexto de un problema más complejo del que únicamente se conocen los datos. Es por eso que en este trabajo se abordó una metodología que tiene la capacidad de realizar el agrupado de los datos que comparten características similares. Los resultados demuestran que la arquitectura neuronal perceptrón multicapa propuesta tiene la capacidad de extraer la información esencial de un conjunto de características y reconstruir los datos iniciales, empleando esta información para el agrupado y separación de clases en la misma actividad.

Referencias

[1] University of California, Irvine (s.a.). *Wine data set available on line*: `https://archive.ics.uci.edu/dataset/109/wine`. (Consultado el 28 de junio de 2023).

[2] M. Hagan, H. Demuth, M. Beale, and O. De Jesus, (2014). *Neural network design*, 2nd edition (2014).

[3] Y. D. Liang, (2013). *Introduction to programming using Python*.

[4] Autor (año). *Título del artículo, libro, revista o nombre de la página web, texto restante*. `https://www.lipsum.com/feed/html`

[5] Scikit learn 1.3.0, (s.a.). *User Guide, available on line*: `https://scikit-learn.org/stable/about.html`. (Consultado el 11 de julio de 2023).

Anzueto, A. & Tovar, B. (1 de noviembre de 2023). *Arquitectura neuronal multicapa para tareas de agrupamiento*. Boletín UPIITA. 18 (99).