ITI: MINERÍA DE DATOS APLICADA

ALUMNO: TORRES GRIMALDO LUIS ANGEL

EVALUACIÓN UNIDAD 3

TÍTULO: "IMPLEMENTACIÓN DE RBFN PARA TAREAS DE CLASIFICACIÓN EN PYTHON"

INTRODUCCIÓN:

¿Qué es "Clustering"?

En ciencias computacionales cuando se habla de "***Clustering***" o ***agrupamiento*** se hace referencia a técnicas de aprendizaje no supervisado. Los algoritmos de **Clustering** por medio de cierto criterio establecido identifican semejanzas entre los elementos pertenecientes a la población y los separan en grupos de elementos similares.

Aplicaciones:

Las técnicas de agrupamiento encuentran aplicación en diversos ámbitos.

- En biología para clasificar animales y plantas.
- En medicina para identificar enfermedades.
- En marketing para identificar personas con hábitos de compras similares.
- En teoría de la señal pueden servir para eliminar ruidos.
- En biometría para identificación del locutor o de caras.

Existen dos grandes técnicas para el agrupamiento de casos:

- Agrupamiento jerárquico, que puede ser aglomerativo o divisivo.
- Agrupamiento no jerárquico, en los que el número de grupos se determina de antemano y las observaciones se van asignando a los grupos en función de su cercanía. Existen los métodos de k-mean y k-medoid.

Existen diversas implementaciones de algoritmos concretos. Por ejemplo, el de las k-medias, de particionamiento. Es uno de los más antiguos pero uso extendido a pesar de sus carencias y falta de robustez.

Función de base radial.

Las funciones de base radial han sido utilizadas en diversas técnicas de reconocimiento de patrones como interpolación, aproximación de funciones, agrupamiento, mezclas de modelos, etc.

En 1988, Broomhead y Lowe propusieron las redes neuronales de función de base radial (RBFN, *radial basis function nwtwork*) como un método alternativo de perceptrón multicapa.

La RBFN utiliza una topología invariante de tres capas y puede realizar mapeos muy complejos, en donde una red perceptrón multicapa necesitaría múltiples capas ocultas.

La salida de la RBFN implementa una suma ponderada de las respuestas lineales de las neuronas en la capa oculta.

MARCO TEÓRICO:

K-MEANS:

Dado un conjunto de observaciones ($\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$), donde cada observación es un vector real de d dimensiones, k-means construye una partición de las observaciones en k conjuntos ($k \leq n$) a fin de minimizar la suma de los cuadrados dentro de cada grupo (WCSS): $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2$$

donde $\boldsymbol{\mu}_i$ es la media de puntos en S_i.

Agrupamiento (Clustering) como problema de optimización:

El agrupamiento se puede visualizar como un problema de optimización en el cual hay qué:

- Minimizar la DISPERSIÓN INTRA-GRUPO
- Maximizar la DISPERSIÓN INTER-GRUPO

Para evaluar estas 2 características se pueden utilizar índices de validación que describan el grado de cohesión (los patrones en un grupo deben ser similares) y la separación (los grupos deben estar bien separados).

La relación entre la dispersión intra-grupo y la dispersión inter-grupo puede ser descrita mediante un índice de valides de grupos (CVI), el cual se utiliza como función de costo o función objetivo.

RED DE FUNCIÓN DE BASE RADIAL (*Radial Basis Function Network, RBFN):*

El entrenamiento de la RBFN consiste en determinar los parámetros de los centros y radios de las RBFs así como los pesos sinápticos entre las capas oculta y salida.

1. Aprendizaje no supervisado: Los centros de las RBFs se determinan a partir de un algoritmo de agrupamiento y posteriormente se computan los radios de base en los centros obtenidos. El número de grupos corresponde al número de neuronas ocultas.
2. Aprendizaje supervisado: Dadas las respuestas de las neuronas ocultas, los pesos sinápticos se ajustan mediante un algoritmo supervisado que realice el mapeo desde la capa de entrada hasta la entrada de salida de las neuronas de salida. El número de neuronas de salida corresponde al número de clases.

Cada neurona de la capa de salida implementa una combinación lineal de las respuestas no loneales de las neuronas ponderadas con pesos sinápticos:

$$z_k = w_{0k} + \sum_{j=1}^{H} w_{jk}\phi_j(\mathbf{x})$$

Se deben minimizar las diferencias cuadradas entre las salidas de la red Z_K y las referencias y_K.

$$J(\mathbf{w}) = \frac{1}{2}\sum_{k=1}^{O}(y_k - z_k)^2$$

Donde **w** son todos los pesos entre las capas oculta y salida.

Para computar los pesos se puede utilizar el algoritmo de descenso de gradiente o mediante el método de la pseudoinversa.

El método de la pseudoinversa proporciona una solución rápida y directa para el cálculo de los pesos sinápticos:

$$W = \left[(\Phi\Phi^T)^{-1}\Phi\right]^T Y$$

Donde Φ, *K* y *W* son las matrices de respuestas RBF, referencias y pesos, respectivamente, calculadas a partir del conjunto de entrenamiento *N* patrones.

La variable *Y* es una matriz binaria donde el número de columnas está definido en función del número de clases del problema. Suponiendo que existe una función sus parámetros serían: K=Número de clases, Y=Vector. La función retornará una matriz "objetivo" de longitud(Y) x K. Si K=3 y el elemento en el índice 0 del vector Y es 1 entonces la primer fila de la matriz "objetivo" sería {1,0,0}, si el segundo elemento del vector Y es 3, entonces la segunda fila de la matriz "objetivo" sería {0,0,1} y así para todas las filas.

Lo siguiente sería agregarle una fila de 1's a la matriz Φ que generó la función de base radial, conocida como "bias". Ésta fila es necesaria para la implementación del algoritmo.

Ahora se multiplican las matrices donde si K=3, entonces las dimensiones serían:

```
phi[3+1,len(data)] y objetivo[len(data),3]
```

Y la matriz resultante es de dimensiones (4x3), para este caso hipotético. La matriz resultante es **W**, la matriz de pesos.

Teniendo en cuenta que Φ es un resultado de la función RBF pero la matriz de entrada a RBF era un patrón de entrenamiento, ahora se usa nuevamente RBF pero con un patrón de prueba que llamaremos Φ_2.

El resultado del patrón de prueba se le agrega como primer fila una fila de 1's que representa el bias, se aplica transpuesta de Φ_2 y se multiplica para cada fila de Φ_2 con la matriz de pesos.

Esto es posible ya que en el caso hipotético de que K=3 **Φ_2[i]={1,4}** y **W={4,3}.** Esto retornaría un vector de {1,3} con valores escalares y ahora lo que se tiene que ver es el valor mayor de estos 3 valores retornados. Dependiendo el índice del valor mayor es la etiqueta de clase que el "clasificador" (RBFN) calcula. Por ejemplo, si para los patrones de prueba 0 y 1:

- Φ_2[0] * W = {0.131, 2, 4}, entonces la etiqueta de clase correspondiente es 3.
- Φ_2[1] * W = {1,3,1}, entonces la etiqueta de clase correspondiente es 2.

Lo siguiente que se debe hacer es un algoritmo que divida el dataset en 80% instancias de entrenamiento y un 20% instancias de prueba, 5 veces, siempre diferentes rangos, para cada valor de K.

Por ejemplo, para K=3 se ejecutará un algoritmo donde estas separaciones las llamaremos “folds”, para cada “fold” el 20% de prueba y el 80% de entrenamiento serán diferentes instancias del dataset. En un caso hipotético donde el dataset fuera de 100 instancias, cuando fold=1 las primeras 20 serían las instancias de prueba y las 80 restantes las de prueba, segunda iteración cuando fold=2 de la instancia 21 a la 40 serían los elementos de prueba y los restantes elementos serían instancias de entrenamiento y así sucesivamente. Este algoritmo a implementar es genérico y se aplica para cualquier dataset de cualquier cantidad de instancias.

Y adicionalmente para cada fold se calcula el Coeficiente de Relación de Mathews, este mide la genericidad del clasificador, se hace una sumatoria de cada fold y se saca el promedio, este se guarda y finalmente para cada dataset se grafican los resultados.

DESARROLLO:

La implementación de ambos algoritmos fue en el Lenguaje Python, como herramientas de apollo se utilizaron las librerias:

- **Matplotlib.pyplot:** Librería utilizada para graficación de los datos, dando una visualización de la salida de una forma más amigable.
- **re:** Librería utilizada para dividir una cadena con un parámetro multi-delimitador, facilitando la división de columntas por espacios, tabs, enters, comas, etc.
- **Numpy:** Facilitó el manejo de matrices en algunas operaciones.
- **Math:** Librería de python que cuenta con algunas funciones matemáticas, las utilizadas en el proyecto fueron: exponencial - “exp()”
- **Sklearn.metrics – matthews_corrcoef:** Es la implementación de la librería sklearn en python de la función que calcula el *Coeficiente de relación de Matthews*.
- **Statistics:** Es una librería implementada en python que cuenta con funciones de estadística, en el proyecto se implementó para calcular la *desviación estándar* y la *media* de un conjunto de datos en específico.

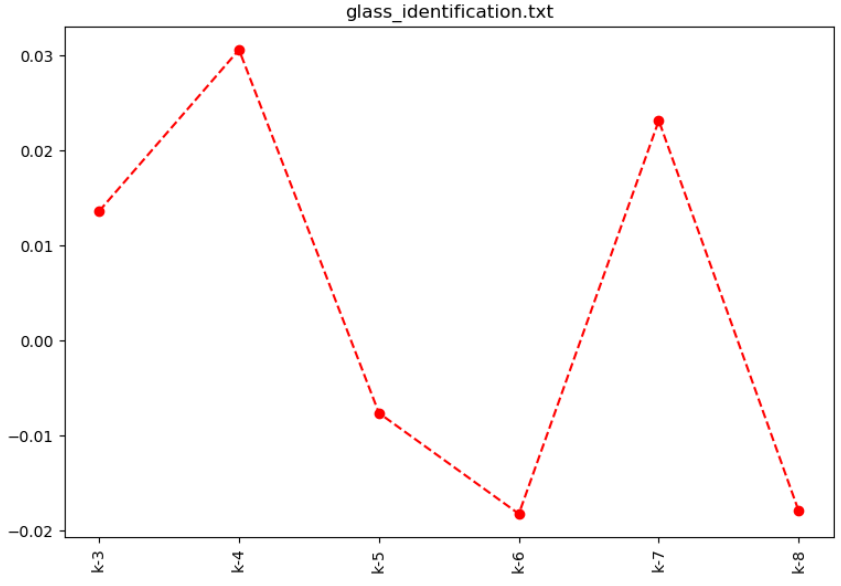
Nota: La única librería/función “no permitida” que se utilizó fue la que calcula el *Coeficiente de relación de Matthews.*

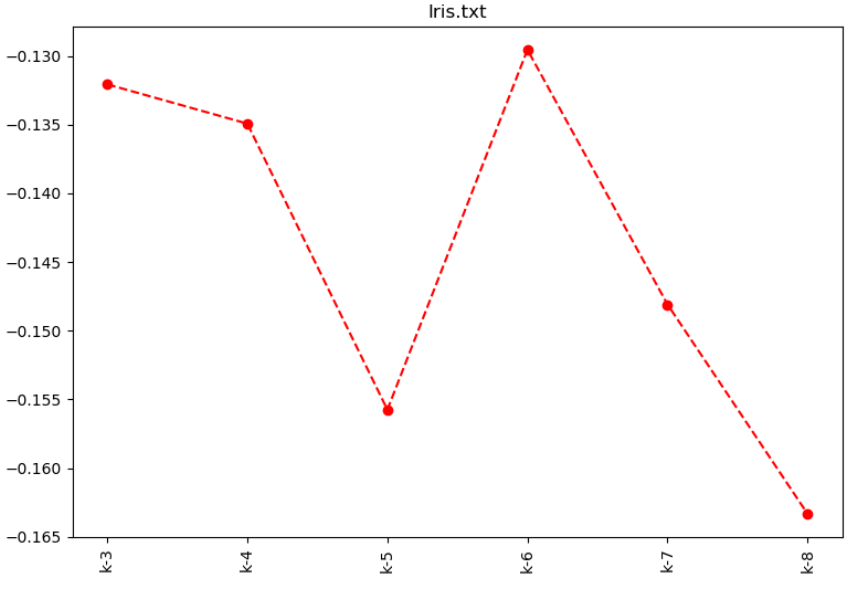
Datasets (problemas) con los que se trabajó:

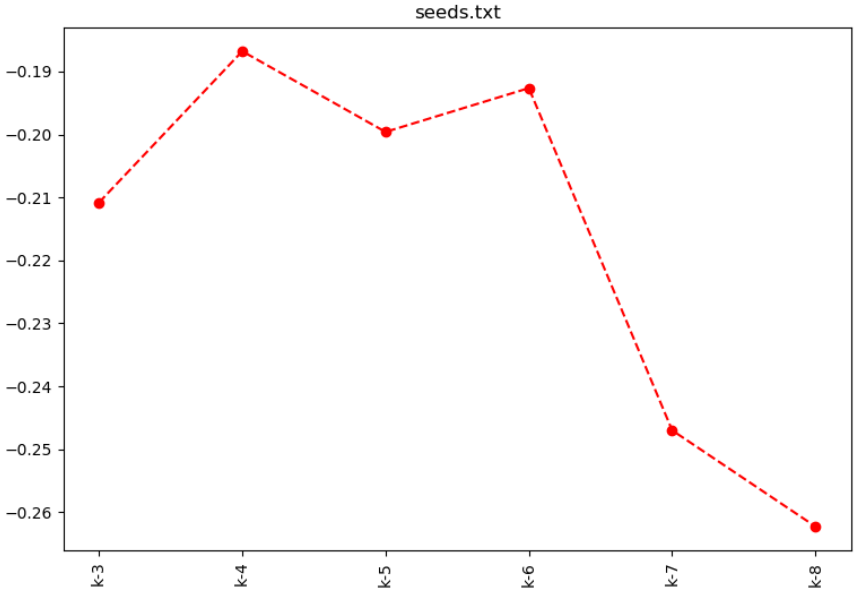
- Glass identification
- Iris
- Seeds

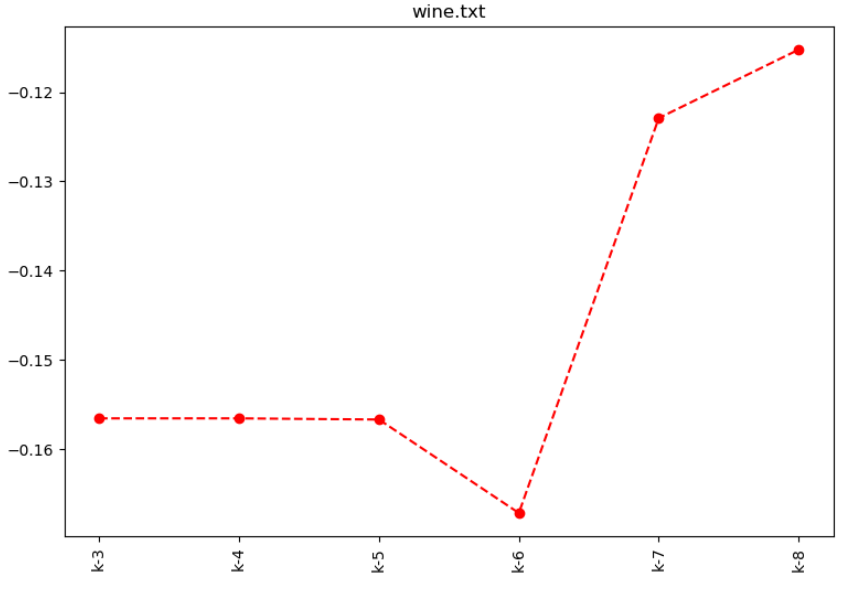
- Wine

RESULTADOS:









Los conocimientos adquiridos para completar el desarrollo de los algoritmos en su mayor parte fueron provenientes del profesor que imparte el curso.

El algoritmo fue ejecutado una sola vez y el resultado de este fueron las imagenes donde se visualiza los valores que obtuvo cuando K={3,4,5,6,7,8}. Cada valor para cada K es el promedio

del Coeficiente de Relación de Matthews, el promedio del CRM retorna un valor entra [-1,1], entre más cercano 0 sea el valor significa que los resultados de las etiquetas que retorna el clasificador es más exacto.

Conclusiones:

1. Se sabe que en ocasiones para los problemas con los que se trabajó se encontrarán "clusters" sin elementos, entonces hay una pequeña excepción en medio de la ejecución del código. **¿Porqué esta excepción? (conclusión del alumno)** Se cree que los problemas no son tan robustos y al exigirle al algoritmo que por ejemplo agrupe en K=20 cuando Iris tiene solo 150 es sobre exigirle y agarra secciones en el plano donde no hay elementos y es ahí donde ocurre esta excepción, sin embargo, no siempre sucede y el algoritmo puede ejecutarse a la perfección en algunas ocaciones.
2. Habiendo partido de la pauta de otras implementaciones del algoritmo con otros dataset, inicialmente se creía que al graficar los datos las funciones tendrían una forma descendente y no ascendente como realmente fue.

CONCLUSIÓNES:

El mejor resultado no es prescisamente cuando K = número de etiquetas de clase.

Conclusiones personales:

Es necesario profundizar en el tema para entenderlo ya que para el alumno son de interés personal. Extender conocimientos matemáticos es fundamental, ya que aunque se han seguido los algoritmos que el profesor ha proporcionado a la clase queda la insertidumbre de “haberlo hecho bien”.