

# Clustering - Algoritmo de Agrupamiento Aglomerativo Jerárquico

Jose Antonio Mejia

Sistemas Inteligentes

17 De Septiembre del 2018

# Contexto

- 1 Contexto
- 2 Tecnico
- 3 Experimentacion
- 4 Conclusiones

- Un Algoritmo de Agrupación Jerárquica son los que producen una secuencia anidada de particiones del conjunto de objetos, es decir, los grupos se organizan de forma jerárquica y cada grupo (cluster) puede verse como la unión de otros grupos (clusters), obteniendo así distintos niveles de jerarquía de grupos.
- Los algoritmos de agrupación jerárquica en realidad se dividen en 2 categorías: de arriba hacia abajo o de abajo hacia arriba. Los algoritmos ascendentes tratan cada punto de datos como un único grupo al principio y luego fusionan (o aglomeran) pares de clústeres hasta que todos los clústeres se hayan fusionado en un solo clúster que contenga todos los puntos de datos.

- La agrupación jerárquica ascendente se denomina, por lo tanto, agrupamiento aglomerativo jerárquico o HAC. Esta jerarquía de conglomerados se representa como un árbol (o dendrograma). La raíz del árbol es el clúster único que reúne todas las muestras, siendo las hojas los grupos con solo una muestra.

- El algoritmo de agrupamiento aglomerativo jerarquico es utilizado en gran medida para ver que tan relacionados estan los datos a los cuales se les aplicara. Gracias a que estos algoritmos fueron dise;ados para crear un dendograma con el resultado final, en este se puede ver que tan relacionados estan los datos en base a sus distacias por poner un ejemplo.

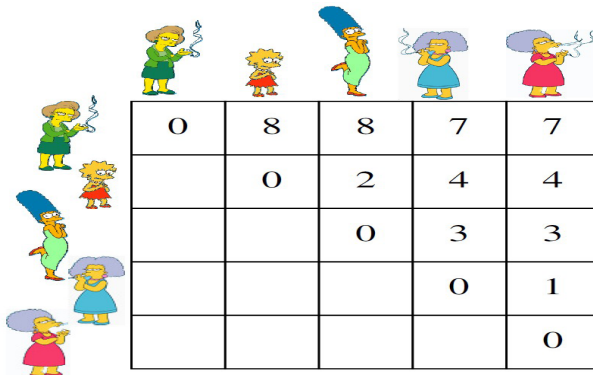
- El primer paso para realizar este algoritmo es generar una matriz de distancias. Para generar esta matriz tendremos que usar la distancia euclidiana que es una formula matematica para determinar la distancia entre dos puntos en un espacio euclideo.






$$d_E(P, Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2} \quad (1)$$

- Una vez generada la matriz se utiliza el criterio de vinculacion conocido como agrupación mínima o de enlace único para determinar que puntos unir.

$$\min \{ d(a, b) : a \in A, b \in B \} \quad (2)$$

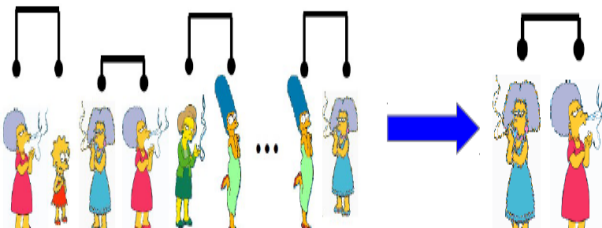
# Ejemplo



				
0	8	8	7	7
	0	2	4	4
		0	3	3
			0	1
				0

- Esta es una matriz de distancia ya generada, ahora procedemos a comparar todos los datos que hay en la matriz hasta encontrar el que tenga la menor distancia.

# Ejemplo



- Una vez realizada la comparacion y encontrado cual es el valor con la menor distacia entre si procedemos a realizar la union.

$$\min d(7, 7) : 7 \quad (3)$$

$$\min d(4, 4) : 4 \quad (4)$$

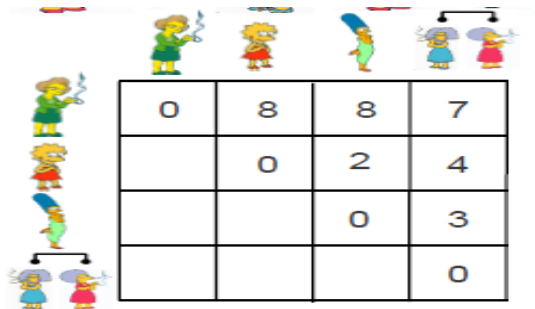
$$\min d(3, 3) : 3 \quad (5)$$









$$\min d(0, 1) : 0 \quad (6)$$



# Ejemplo

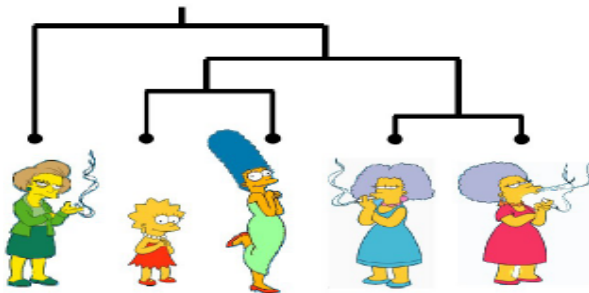
- Después de haber realizado la unión dada como resultado la siguiente matriz:



				
	0	8	8	7
		0	2	4
			0	3
				0

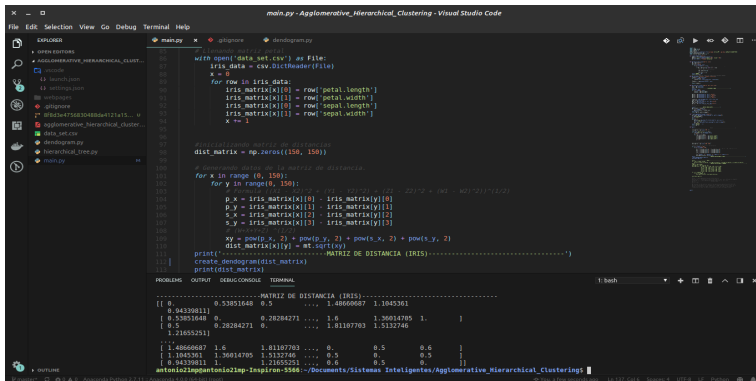
# Ejemplo

- El paso 1 y el paso 2 se repiten hasta que la matriz solo tenga un tamaño de  $2 \times 2$  y luego se genera un dendrograma:



# Experimentacion

- Este algoritmo se aplico al conjunto de datos IRIS dando como resultado los siguientes datos.



The screenshot shows a Visual Studio Code editor window titled "main.py - Agglomerative\_Hierarchical\_Clustering - Visual Studio Code". The editor contains a Python script that reads the Iris dataset from "data\_set.csv", calculates a distance matrix, and generates a dendrogram. The script is as follows:

```
1 # Leer datos de la matriz de distancias
2
3 with open('data_set.csv') as File:
4     iris_data = csv.DictReader(File)
5     x = 0
6     for row in iris_data:
7         iris_matrix[x][0] = row['petal.length']
8         iris_matrix[x][1] = row['petal.width']
9         iris_matrix[x][2] = row['sepal.length']
10        iris_matrix[x][3] = row['sepal.width']
11        x += 1
12
13 # Crear la matriz de distancias
14
15 dist_matrix = np.zeros((150, 150))
16
17 # Calcular la matriz de distancias
18
19 for x in range(0, 150):
20     for y in range(0, 150):
21         p_x = iris_matrix[x][0] - iris_matrix[y][0]
22         p_y = iris_matrix[x][1] - iris_matrix[y][1]
23         s_x = iris_matrix[x][2] - iris_matrix[y][2]
24         s_y = iris_matrix[x][3] - iris_matrix[y][3]
25
26         xy = pow(p_x, 2) + pow(p_y, 2) + pow(s_x, 2) + pow(s_y, 2)
27
28         dist_matrix[x][y] = math.sqrt(xy)
29
30 print('-----MATRIZ DE DISTANCIA (IRIS)-----')
31 create_dendrogram(dist_matrix)
32 print(dist_matrix)
```

The terminal output shows the distance matrix and the dendrogram. The distance matrix is a 150x150 matrix with values ranging from 0.0 to 1.48660687. The dendrogram is a tree diagram showing the hierarchical clustering of the data points.

## Resultados

The screenshot shows a Visual Studio Code window titled "main.py - Agglomerative\_Hierarchical\_Clustering - Visual Studio Code". The Explorer sidebar on the left lists files: .vscode, launch.json, settings.json, webpages, .gitignore, BF8d3e475683048Bda4121a15..., agglomerative\_hierarchical\_cluster..., data\_set.csv, dendrogram.py, hierarchical\_tree.py, and main.py. The main editor area shows the code for main.py, which prints several matrices and dendrograms. The output is as follows:

```

-----MATRIZ RESULTANTE-----
[[ 0.          0.31622777  0.60827625  0.4           0.36055513  1.56204994
   1.45602198  1.91049732]
 [ 0.31622777  0.          2.14709106  0.94339811  1.52970585  2.0808652
   1.7         2.0880613 ]
 [ 0.60827625  2.14709106  0.          1.          0.41231056  2.2090722
   2.99666481  3.53411941]
 [ 0.4         0.94339811  1.          0.          0.31622777  0.31622777
   0.4         0.78102497]
 [ 0.36055513  1.52970585  0.41231056  0.31622777  0.          1.5132746
   2.19317122  2.73130006]
 [ 1.56204994  2.0880652  2.2090722  0.31622777  1.5132746  0.
   1.2083046   1.64012195]
 [ 1.45602198  1.7         2.99666481  0.4         2.19317122  1.2083046
   0.          0.53851648]
 [ 1.91049732  2.0880613  3.53411941  0.78102497  2.73130006  1.64012195
   0.53851648  0.        ]]

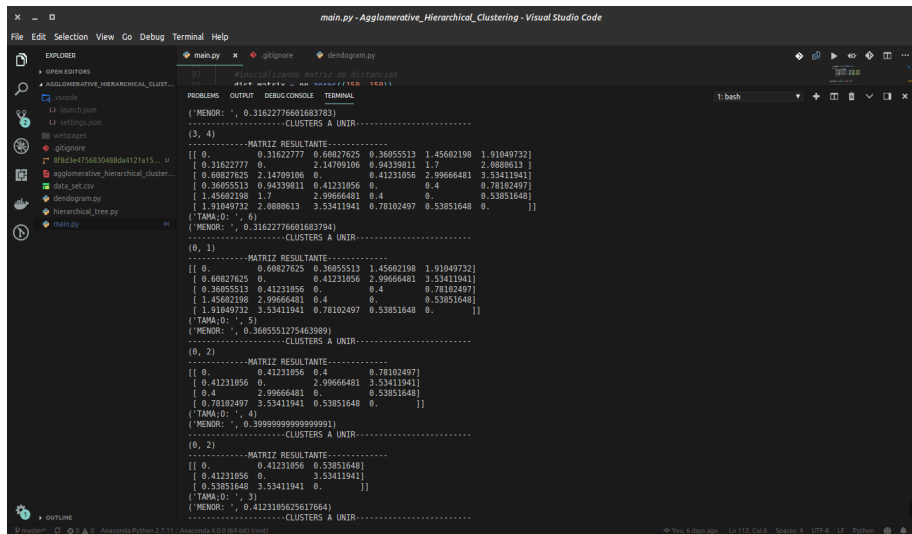
('TAMA:0 ', 0)
('MENOR: ', 0.31622776601683766)
-----CLUSTERS A UNIR-----
(3, 5)

-----MATRIZ RESULTANTE-----
[[ 0.          0.31622777  0.60827625  0.4           0.36055513  1.45602198
   1.91049732]
 [ 0.31622777  0.          2.14709106  0.94339811  1.52970585  1.7
   2.0880613 ]
 [ 0.60827625  2.14709106  0.          1.          0.41231056  2.99666481
   3.53411941]
 [ 0.4         0.94339811  1.          0.          0.31622777  0.4
   0.78102497]
 [ 0.36055513  1.52970585  0.41231056  0.31622777  0.          2.19317122
   2.73130006]
 [ 1.45602198  1.7         2.99666481  0.4         2.19317122  0.
   0.53851648]
 [ 1.91049732  2.0880613  3.53411941  0.78102497  2.73130006  0.53851648
   0.        ]]

('TAMA:0 ', 7)
('MENOR: ', 0.31622776601683783)
-----CLUSTERS A UNIR-----
(3, 4)

```

# Resultados

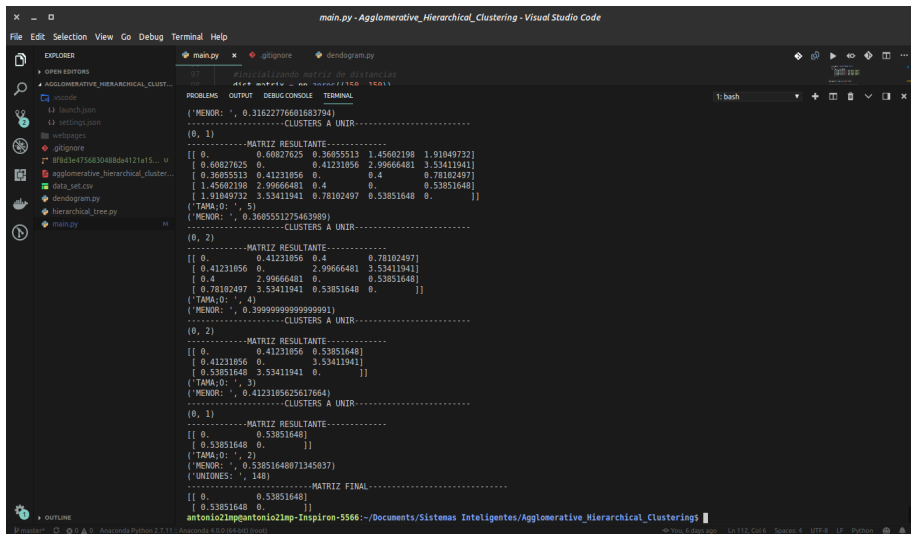


```
main.py - Agglomerative_Hierarchical_Clustering - Visual Studio Code
File Edit Selection View Go Debug Terminal Help

EXPLORER
  OPEN EDITORS
  AGGLOMERATIVE_HIERARCHICAL_CLUSTERING
  .vscode
    launch.json
    settings.json
    webpages
    .gitignore
    .f8fd3e4756830488da4121a15... u
    agglomerative_hierarchical_cluster...
    data_set.csv
    dendrogram.py
    hierarchical_tree.py
    main.py

TERMINAL
  1: bash
  ('MEMOR: ', 0.31622776601683783)
  -----CLUSTERS A UNIR-----
  (3, 4)
  -----MATRIZ RESULTANTE-----
  [[ 0.          0.31622777  0.60827625  0.36055513  1.45602198  1.91049732]
   [ 0.31622777  0.          2.14709106  0.94339811  1.7          2.0880613 ]
   [ 0.60827625  2.14709106  0.          0.41231056  2.99666481  3.53411941]
   [ 0.36055513  0.94339811  0.41231056  0.          0.4          0.78102497]
   [ 1.45602198  1.7          2.99666481  0.4          0.          0.53851648]
   [ 1.91049732  2.0880613  3.53411941  0.78102497  0.53851648  0.          ]]
  ('TAMA:0: ', 6)
  ('MEMOR: ', 0.31622776601683794)
  -----CLUSTERS A UNIR-----
  (0, 1)
  -----MATRIZ RESULTANTE-----
  [[ 0.          0.60827625  0.36055513  1.45602198  1.91049732]
   [ 0.60827625  0.          0.41231056  2.99666481  3.53411941]
   [ 0.36055513  0.41231056  0.          0.4          0.78102497]
   [ 1.45602198  2.99666481  0.4          0.          0.53851648]
   [ 1.91049732  3.53411941  0.78102497  0.53851648  0.          ]]
  ('TAMA:0: ', 5)
  ('MEMOR: ', 0.3605551275463989)
  -----CLUSTERS A UNIR-----
  (0, 2)
  -----MATRIZ RESULTANTE-----
  [[ 0.          0.41231056  0.4          0.78102497]
   [ 0.41231056  0.          2.99666481  3.53411941]
   [ 0.4          2.99666481  0.          0.53851648]
   [ 0.78102497  3.53411941  0.53851648  0.          ]]
  ('TAMA:0: ', 4)
  ('MEMOR: ', 0.39999999999999991)
  -----CLUSTERS A UNIR-----
  (0, 2)
  -----MATRIZ RESULTANTE-----
  [[ 0.          0.41231056  0.53851648]
   [ 0.41231056  0.          3.53411941]
   [ 0.53851648  3.53411941  0.          ]]
  ('TAMA:0: ', 3)
  ('MEMOR: ', 0.4123105625617664)
  -----CLUSTERS A UNIR-----
```

# Resultados



main.py - Agglomerative\_Hierarchical\_Clustering - Visual Studio Code

File Edit Selection View Go Debug Terminal Help

EXPLORER

OPEN EDITORS

AGGLOMERATIVE\_HIERARCHICAL\_CLUSTERING.py

main.py

97 #inicializando matriz de distancias

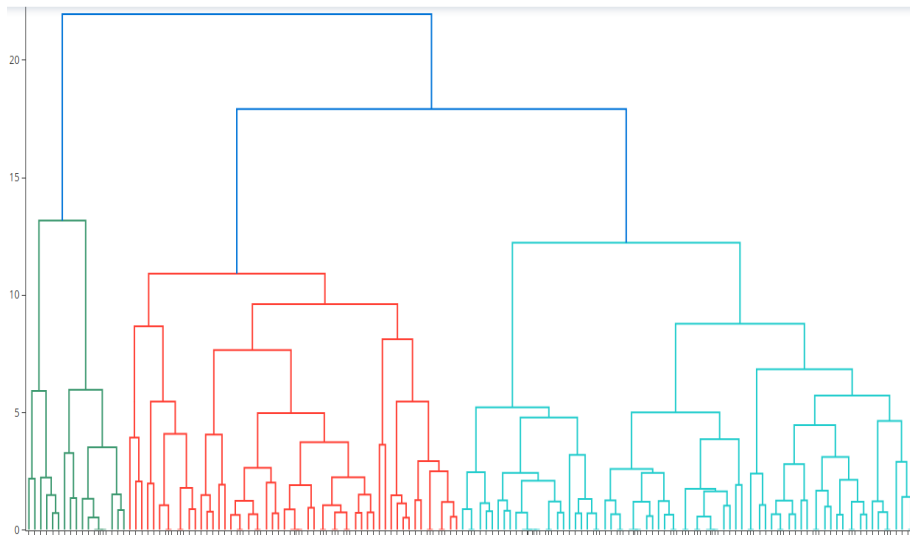
def main():

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: bash

```
(('MENOR: ', 0.31622776601683794)
-----CLUSTERS A UNIR-----
(0, 1)
-----MATRIZ RESULTANTE-----
[[ 0.          0.60827625  0.36855513  1.45602198  1.91049732]
 [ 0.60827625  0.          0.41231056  2.99666481  3.53411941]
 [ 0.36855513  0.41231056  0.          0.4         0.78102497]
 [ 1.45602198  2.99666481  0.4         0.          0.53851648]
 [ 1.91049732  3.53411941  0.78102497  0.53851648  0.          ]]
('TAMA:0: ', 5)
('MENOR: ', 0.3685551275463989)
-----CLUSTERS A UNIR-----
(0, 2)
-----MATRIZ RESULTANTE-----
[[ 0.          0.41231056  0.4         0.78102497]
 [ 0.41231056  0.          2.99666481  3.53411941]
 [ 0.4         2.99666481  0.          0.53851648]
 [ 0.78102497  3.53411941  0.53851648  0.          ]]
('TAMA:0: ', 4)
('MENOR: ', 0.39999999999999991)
-----CLUSTERS A UNIR-----
(0, 2)
-----MATRIZ RESULTANTE-----
[[ 0.          0.41231056  0.53851648]
 [ 0.41231056  0.          3.53411941]
 [ 0.53851648  3.53411941  0.          ]]
('TAMA:0: ', 3)
('MENOR: ', 0.4123105625617664)
-----CLUSTERS A UNIR-----
(0, 1)
-----MATRIZ RESULTANTE-----
[[ 0.          0.53851648]
 [ 0.53851648  0.          ]]
('TAMA:0: ', 2)
('MENOR: ', 0.53851648071345037)
('UNIONES: ', 148)
-----MATRIZ FINAL-----
[[ 0.          0.53851648]
 [ 0.53851648  0.          ]]
antonio2impantonio2imp-Inspiron-5566-~/Documents/Sistemas Inteligentes/Agglomerative_Hierarchical_Clustering
```

# Resultados



- El Algoritmo de Agrupamiento Aglomerativo Jerárquico es excelente al momento de mostrar resultados gracias a la creacion del dendograma pero tiene dos principales desventajas:
- la primera es que tiene un gran acumulacion de errores ya que si la matriz es demasiado grande y al momento de agrupar hay un minimo error este se propaga durante el resto de la construccion del dendograma sin ser posible repararlo.
- La segunda desventaja es que requiere demasiada memoria ya que al tratar con un conjunto de datos mayor este muestra su mal rendimiento.