

Arquitecturas no supervisadas

Sistemas Inteligentes

Christian García Viguera
Pablo Ortigosa Quevedo

Pablo Carmelo Fernández López
Carmen Paz Suárez Araujo



Índice

[Índice](#)

[Introducción](#)

[Funcionamiento de la Red](#)

[Criterio de parada](#)

[Predict](#)

[Parámetros](#)

[Guardado y cargado](#)

[Guardado](#)

[Cargado](#)

[Connected components](#)

[Principal Component Analysis \(PCA\)](#)

[Cientes Ventas por Mayor](#)

[Preprocesado](#)

[Análisis y la categorización](#)

[Vinos](#)

[Preprocesado](#)

[Análisis y la categorización](#)

Introducción

En esta práctica hemos utilizado el código de la Gas Neural Network proporcionado para el análisis de diferentes conjuntos de datos y su clasificación.

A mitad de camino nos dimos cuenta de que malinterpretamos el enunciado de la práctica y entendimos que debíamos hacer dos ejercicios en lugar de solamente uno. Es por ello que hablaremos del dataset de los clientes a pesar de habernos centrado en el de los vinos, para que el tiempo que invertimos en dicha parte no sea en vano.

Funcionamiento de la Red

La GNG agrupa información correlacionada dentro de un espacio dimensional. Lo realiza mediante la cercanía de las muestras.

En la imagen de la derecha encontramos la solución esperada al dataset *Sample Cluster Data 2D*. Con esto, es posible identificar perfiles dentro del dataset.

Hemos intentado entrenar la red con *Sample Cluster Data 2D*, pero no nos ha sido posible, dado el gran tamaño del dataset, que tiene 10 veces más muestras que el dataset *Clientes Ventas por Mayor*, por lo que las épocas son más largas y no podemos analizar las dimensiones de 2 en 2 o de 3 en 3.

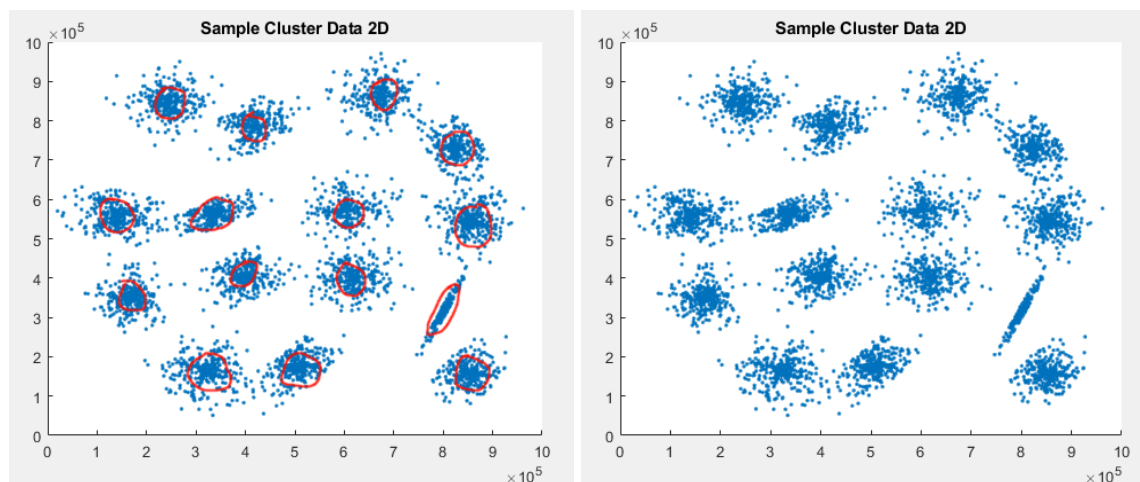


Figura 1: Representación gráfica del conjunto de datos *Sample Cluster Data 2D*.

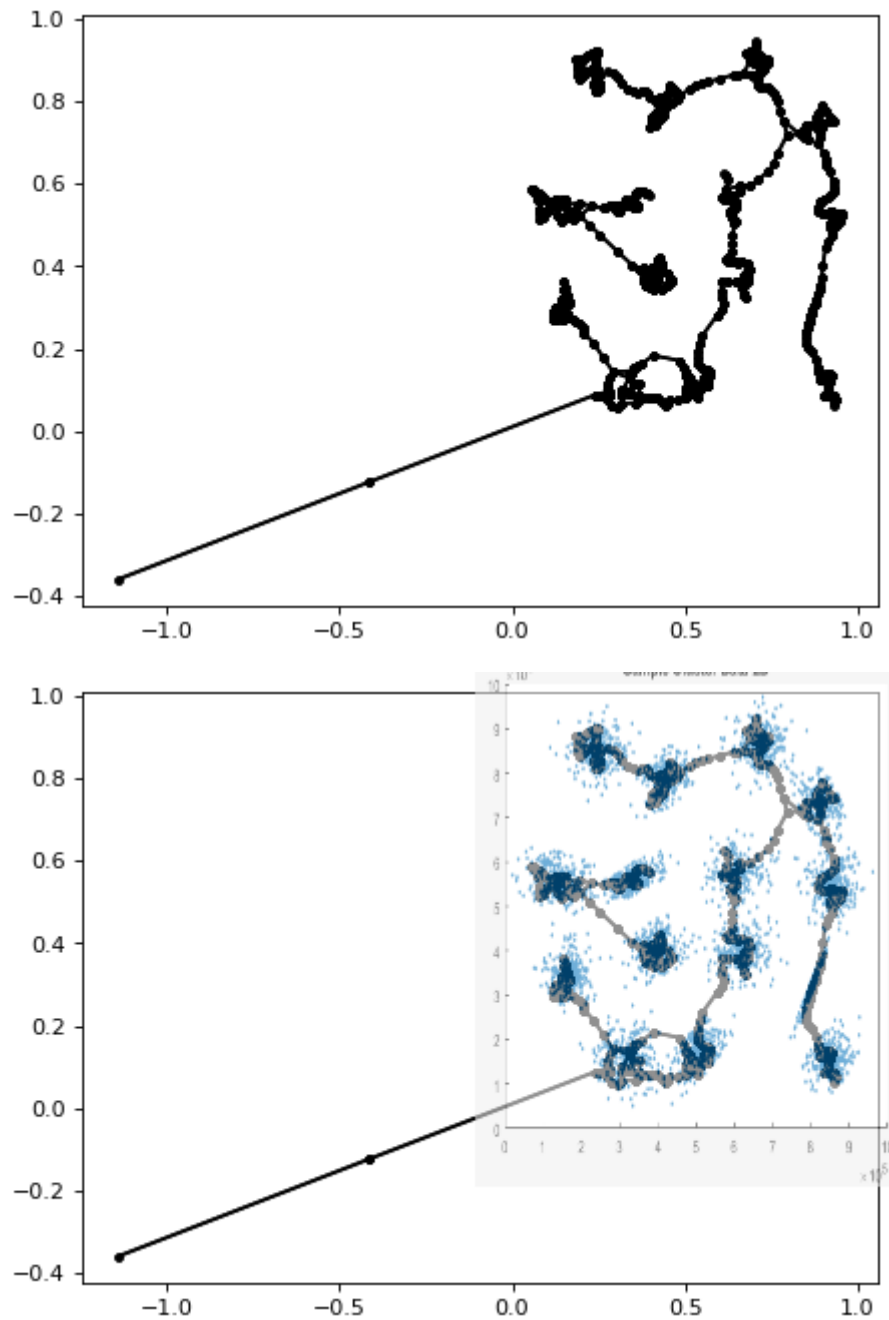


Figura 2: Representación gráfica del conjunto de datos *Sample Cluster Data 2D* superpuesto con el resultado del entrenamiento de la red.

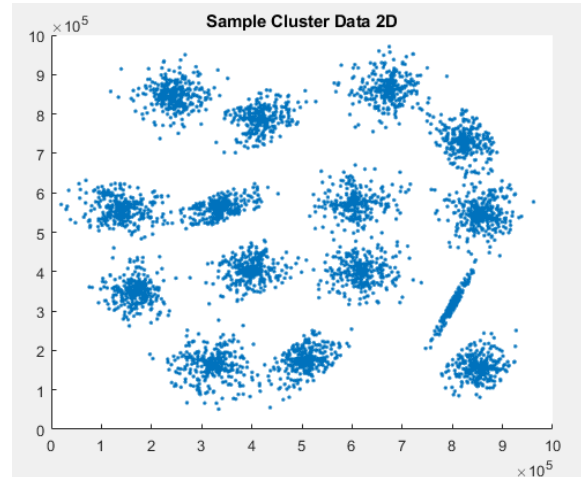
Como podemos ver, la red crea bastantes unidades en las zonas que debería, los clusters, sin embargo hay una unidad que nunca poda y se queda ahí. Esto se debe a la primera iteración de la red pues genera 2 puntos aleatorios para comenzar, estos números no se encuentran dentro del rango de valores del dataset y por lo tanto se genera bastante lejos. Esto nos presenta otro problema, ¿por qué no poda la rama?, la unidad debería aumentar su edad si no se mueve y cuando sea muy vieja ser eliminada, quizás se está moviendo poco a poco. Sea como fuere no está resultando en el comportamiento esperado.

Criterio de parada

Se pide incorporar un criterio de parada según el número de agrupamientos. No nos parece un buen criterio, ya que el número de agrupamientos durante el entrenamiento de la red no indica si este está mejorando o no. Por ejemplo:

En el *dataset* “Sample Cluster Data 2D” hay 15 grupos claramente definidos. Si definimos el criterio de parada en más de 15 grupos ¿qué ocurre si la red ha determinado que existen 16 grupos en la última época? ¿y si determina que existen 20?

En caso de asignar dos grupos a un solo *cluster*, podemos decir que el entrenamiento está progresando bien, dado que el número de agrupamientos está aumentando, pero se cometería un error al parar el entrenamiento asumiendo que los 15 agrupamientos son correctos. Parar la red en este punto es, por tanto, contraproducente, dado que en épocas posteriores podría haber reducido el número de grupos uniendo estos grupos que al principio valoró que eran independientes.



Por ello, nos parece mejor el criterio de parada por épocas, dado que se le da la oportunidad a la red de corregir errores y de llegar al número de grupos deseado aunque se pase del mismo en algunas épocas. Guardando los datos de la red al final de cada época se pueden comparar los resultados *a posteriori* para determinar cuando empezó a decaer el entrenamiento y así obtener la época en la que lo hizo mejor.

Predict

Para asignar un agrupamiento a nuevas observaciones utilizamos *findNearestUnit()* y les asignamos el agrupamiento de la unidad más cercana. Lo cual creemos que es lo más lógico el único caso donde esto podría no funcionar sería en el siguiente:

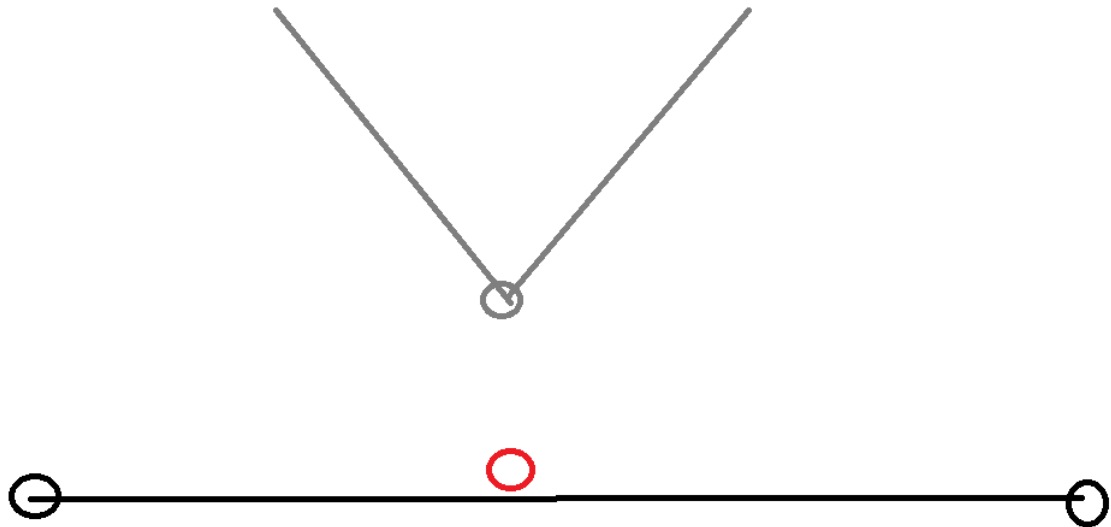


Figura 3: Problema del agrupamiento por cercanía.

Como podemos ver la unidad roja debería pertenecer al grupo negro, que está más cerca, pero la unidad gris es más cercana que las otras unidades, haciendo por tanto que `findNearestUnit` nos devuelva el nodo gris y asumamos incorrectamente que el nodo rojo es parte del grupo gris.

Parámetros

La red tiene diversos parámetros para ajustar su entrenamiento:

- **epsilon_a:** Indica cuanto mover la unidad más cercana a x_i cada vez que es necesario moverla.
- **epsilon_n:** Indica cuanto mover las unidades vecinas a la unidad más cercana a x_i cada vez que es necesario moverlas.
- **a_max:** Edad máxima de una unidad, cada vez que se modifica una unidad se reinicia su edad.
- **eta:** Indica cada cuantas iteraciones crear una nueva unidad x_i .
- **alpha:** Se usa en conjunto con el error, con las unidades que se mueven o creen. Evita que se creen unidades en el mismo lugar todo el rato.
- **delta:** Multiplica al error, así este no crece constantemente. Evita que el error crezca demasiado
- **maxNumberUnits:** El número máximo de unidades que puede crear la red, si llegase a este número, la red pararía, sin embargo hemos eliminado esta condición como comentamos anteriormente.
- **maxNumberCC:** Indica el número de grupos máximos, si llegase a este número, la red pararía, sin embargo hemos eliminado esta condición como comentamos anteriormente.

Guardado y cargado

El proceso de guardado y cargado nos permite guardar los resultados de un entrenamiento y usarlos cuando queramos hacer una predicción con datos nuevos.

En el proceso usamos el formato *json* por simplicidad y por la facilidad para pasar los datos a *Matlab*.

Guardado

Se lleva a cabo en el método *save* en *GrowingNeuralGas*, que se encarga de crear el fichero de la ruta especificada para el guardado, y dentro del cual guarda los datos de *GrowingNeuralGas*.

El guardado de las variables “primitivas” *epsilon_a*, *epsilon_n*, *a_max*, *eta*, *alpha*, *delta*, *maxNumberUnits* y *maxNumberCC* se realiza utilizando un diccionario que se almacena en un fichero *json*.

Por otro lado, *A* y *error* son almacenados en un fichero *npz* utilizando la función *save* de *numpy*.

Por último, *N* es almacenado en un *json* pero utilizando la función *dumps* de la librería *json* de Python

Cargado

El método *load* en *GrowingNeuralGas* realiza la carga de los datos para posteriormente realizar el *predict*.

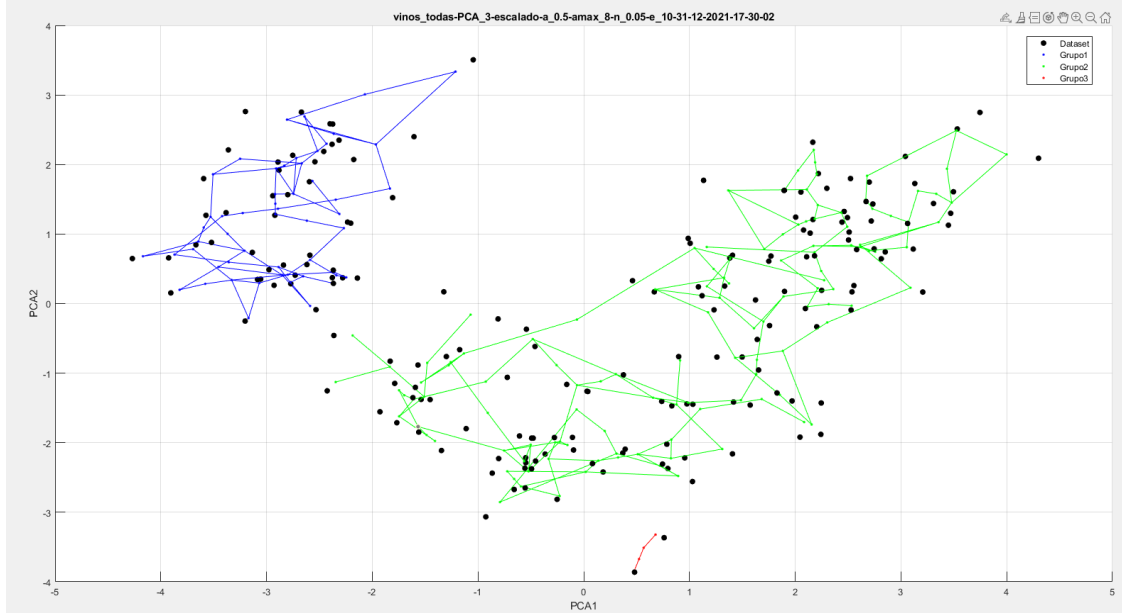
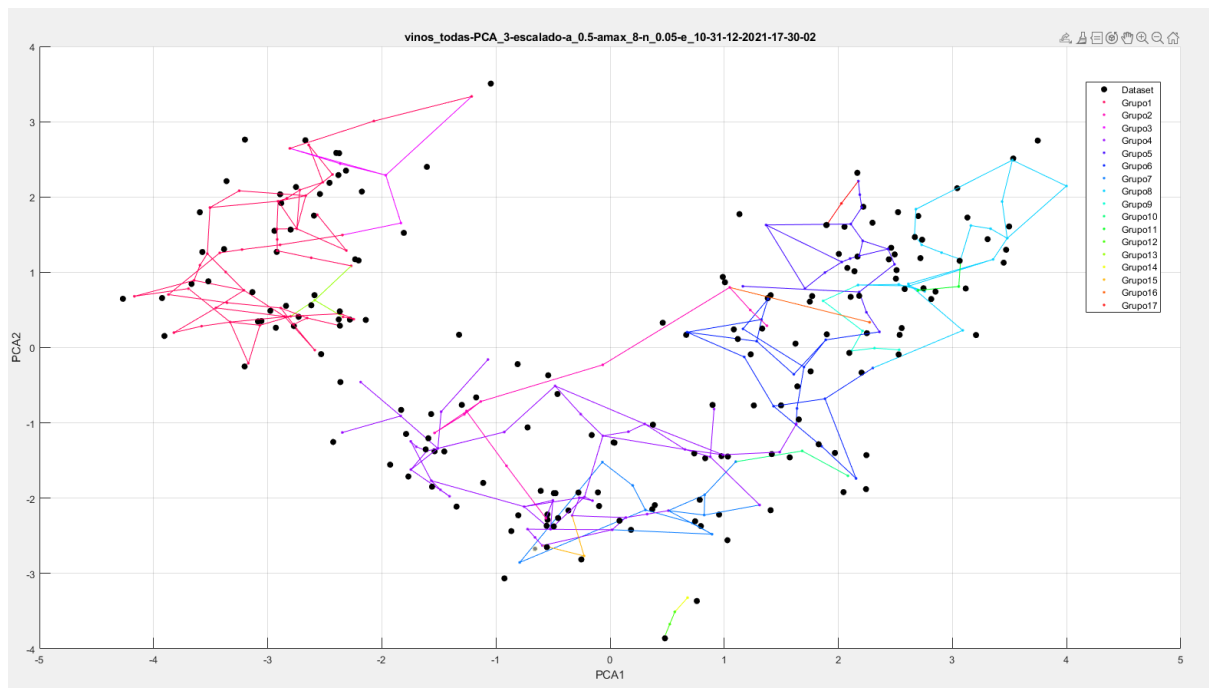
Para las variables *A* y *error* se realiza un cargado de *numpy* y posteriormente se transforman en *Variables* de *Tensorflow*.

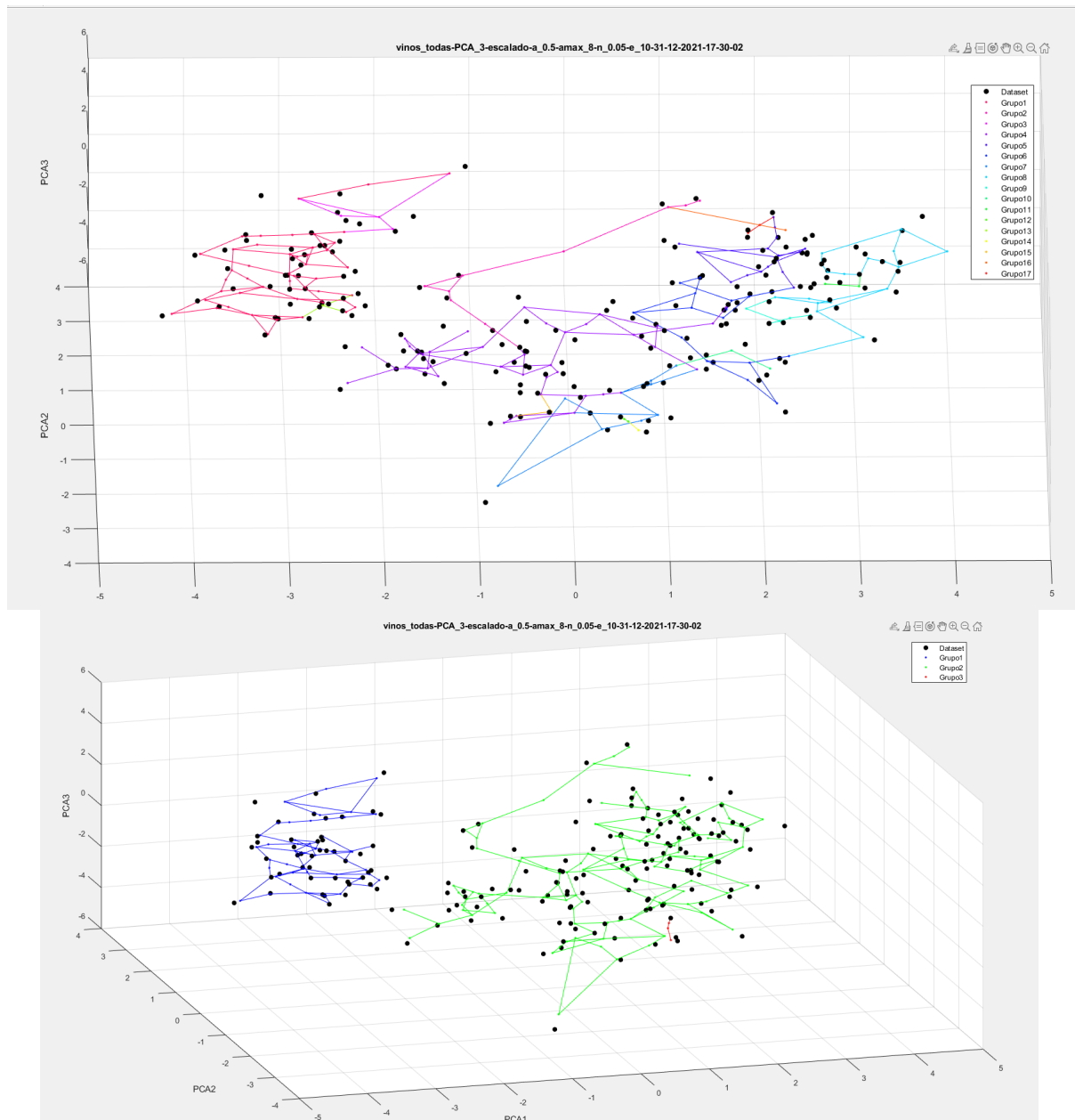
Para *N* se realiza un cargado de la librería *json*, y luego se itera el resultado guardando los elementos en un nuevo objeto *Graph* que se guarda finalmente en la variable *self.N*.

Finalmente, para las variables primitivas, también se hace un cargado de *json* en un diccionario. De este diccionario se buscan los datos por los nombres de sus variables. Por ejemplo: `self.epsilon_n = dic["epsilon_n"]`

Connected components

El método *groups()* el cual usamos para conseguir los grupos es bastante básico, simplemente itera todos los elementos y va actualizando las listas de elementos visitados y a visitar, los que se van a visitar pertenecen al mismo grupo, esta lista va aumentando el tamaño a medida que se iteran las unidades. La lógica comprueba ambas listas y se procede ignorando los elementos ya visitados. Este método lo realizamos nosotros pues el que viene en el código tiene ciertos fallos, mostramos ahora una comparativa de ambos métodos, donde vemos que las conexiones tienen varios grupos incorrectos y unidades que pertenecen a varios grupos. El multicolor es el que venía con el código, el de 3 grupos es el nuestro.





Comparativa: `getConnectedComponents` vs `groups`, los datos proceden de la figura 19
 vinos_todas-PCA_3-escalado-a_0.5-amax_8-n_0.05-e_10-31-12-2021-17-30-02
 Comparativa1.fig y Comparativa2.fig

Principal Component Analysis (PCA)

El PCA es una técnica utilizada para estudiar la correlación entre los datos de un conjunto, la cual nos devuelve los datos en una nueva base donde se indica cuánta información contiene cada variable. Tras esto podemos desechar las variables que poco aportan al dataset con el fin de reducir la dimensionalidad del mismo para poder trabajar y representarlo más fácilmente. Lo hemos realizado en Matlab, donde es más rápido y cómodo dado que la función ya está implementada. Todo quedará más claro cuando se ejemplifique con un dataset.

Cientes Ventas por Mayor

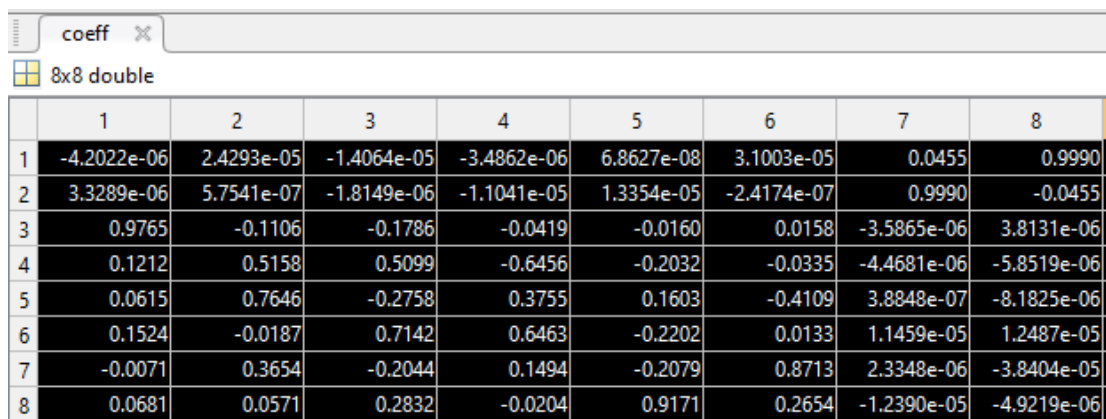
Como ya hemos comentado, para el caso de *Cientes_Ventas_por_Mayor* vamos a aportar lo que hemos realizado a pesar de habernos decantado por el dataset de los vinos.

Preprocesado

Como podemos observar en el resultado de Cientes (**figura 1**), la función *PCA* nos devuelve una matriz de coeficientes respecto a cada variable PCA. Cada columna corresponde con una variable PCA, y están ordenadas descendentemente por varianza. Esta varianza nos ayuda a determinar qué componente principal, es decir, qué combinación de variables originales, aporta mayor información sobre el *dataset*. La variable *explained* devuelta por PCA guarda las varianzas de estas columnas, en el mismo orden, pero en filas, es decir, de arriba a abajo (figura 2).

Las variables del dataset *Cientes_Ventas_por_Mayor.csv*¹ son (por orden): *Channel*, *Region*, *Fresh*, *Milk*, *Grocery*, *Frozen*, *Detergents_Paper* y *Delicassen*. El elemento (3,1) (figura 1), que corresponde con la variable “*Fresh*”, tiene un coeficiente de 0.97 respecto a PCA1, que es la que tiene mayor varianza (figuras 2).

Por tanto, es razonable concluir que la variable que más aporta al dataset es “*Fresh*”, siguiéndole “*Grocery*” y “*Milk*”, y que las que menos aportan son “*Channel*” y “*Region*”.



	1	2	3	4	5	6	7	8
1	-4.2022e-06	2.4293e-05	-1.4064e-05	-3.4862e-06	6.8627e-08	3.1003e-05	0.0455	0.9990
2	3.3289e-06	5.7541e-07	-1.8149e-06	-1.1041e-05	1.3354e-05	-2.4174e-07	0.9990	-0.0455
3	0.9765	-0.1106	-0.1786	-0.0419	-0.0160	0.0158	-3.5865e-06	3.8131e-06
4	0.1212	0.5158	0.5099	-0.6456	-0.2032	-0.0335	-4.4681e-06	-5.8519e-06
5	0.0615	0.7646	-0.2758	0.3755	0.1603	-0.4109	3.8848e-07	-8.1825e-06
6	0.1524	-0.0187	0.7142	0.6463	-0.2202	0.0133	1.1459e-05	1.2487e-05
7	-0.0071	0.3654	-0.2044	0.1494	-0.2079	0.8713	2.3348e-06	-3.8404e-05
8	0.0681	0.0571	0.2832	-0.0204	0.9171	0.2654	-1.2390e-05	-4.9219e-06

Figura 1: Matriz de coeficientes para *Cientes_Ventas_por_Mayor.csv*
Columnas de izquierda a derecha: PCA1, PCA2, ..., PCA8

Como observamos, las variables *Channel* y *Region* no aportan demasiado en ninguna variable PCA, excepto en PCA7 y PCA8, en las cuales vemos que son las únicas que aportan una cantidad significativa, por lo que eliminamos *Channel* y *Region* y volvemos a ejecutar el PCA. Dando resultados más satisfactorios.

¹ Hemos cambiado el nombre de algunos archivos, quitando tildes y espacios.

variable		selection					edit	
explained								
1x6 double								
	1	2	3	4	5	6		
1	45.9614	40.5172	7.0030	4.4023	1.5022	0.6138		

Figura 2: Vector de varianzas para *Cientes_Ventas_por_Mayor.csv*
Columnas de izquierda a derecha: PCA1, PCA2, ..., PCA6.

coeff								
6x6 double								
	1	2	3	4	5	6		
1	0.9765	-0.1106	-0.1786	-0.0419	-0.0160	0.0158		
2	0.1212	0.5158	0.5099	-0.6456	-0.2032	-0.0335		
3	0.0615	0.7646	-0.2758	0.3755	0.1603	-0.4109		
4	0.1524	-0.0187	0.7142	0.6463	-0.2202	0.0133		
5	-0.0071	0.3654	-0.2044	0.1494	-0.2079	0.8713		
6	0.0681	0.0571	0.2832	-0.0204	0.9171	0.2654		

Figura 3: Matriz de coeficientes para *Cientes_Ventas_por_Mayor.csv* sin las variables "Channel" y region. Columnas de izquierda a derecha: PCA1, PCA2, ..., PCA6. **Estas variables PCA son distintas a las anteriores figuras, dado que ahora no contribuye el mismo número de "características" a cada una de ellas (antes 8, ahora 6).**

Análisis y la categorización

Sin los atributos de canal de venta y región. Clientes_Ventas_por_Mayor.csv.

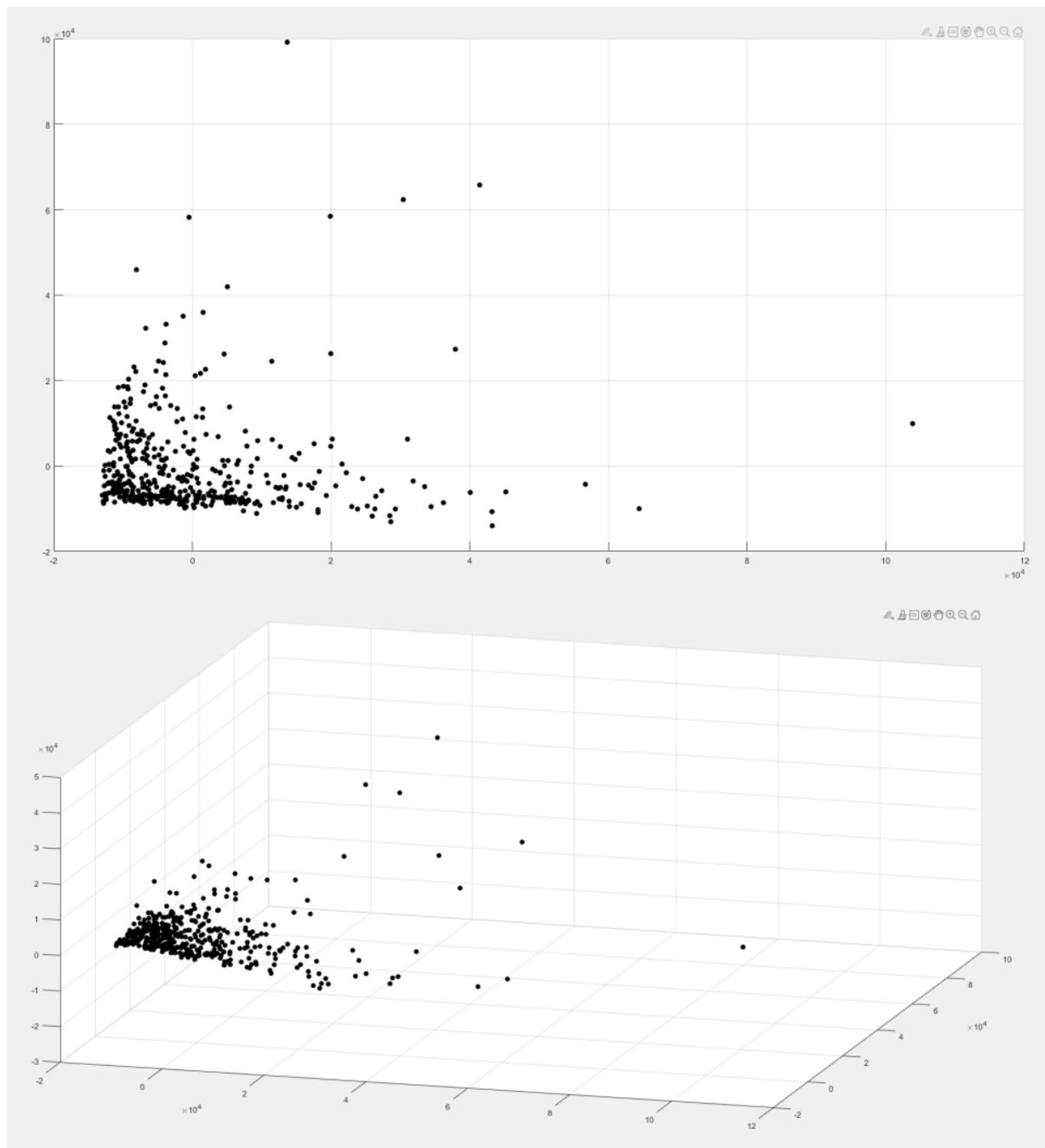
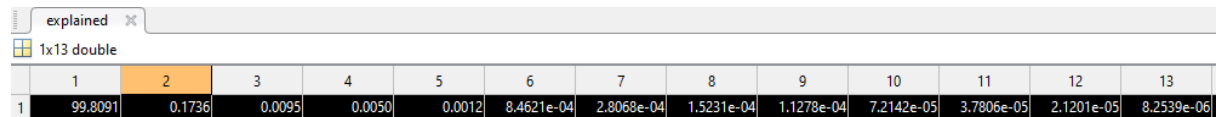


Figura 4: "epsilon_a": 0.5, "epsilon_n": 0.05, "a_max": 8, "eta": 10, "alpha": 1.5, "delta": 0.6, "epoch": 5

Vinos

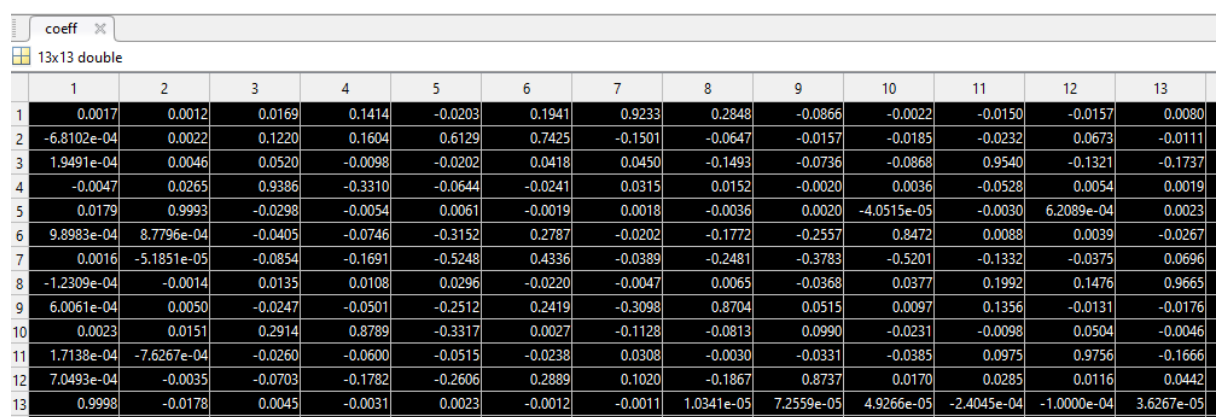
Preprocesado

Esta vez hemos decidido aplicar también un escalado de los datos pues tras el PCA había una representatividad del 99% en tan solo PCA1, lo cual era sospechoso.



	1	2	3	4	5	6	7	8	9	10	11	12	13
1	99.8091	0.1736	0.0095	0.0050	0.0012	8.4621e-04	2.8068e-04	1.5231e-04	1.1278e-04	7.2142e-05	3.7806e-05	2.1201e-05	8.2539e-06

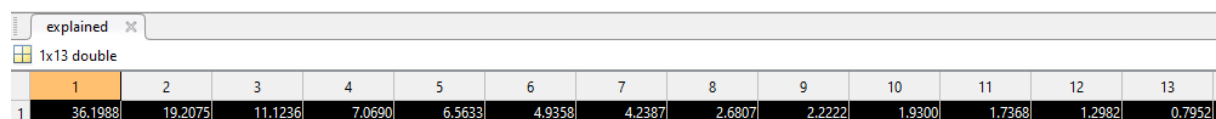
Figura 5: Vector de varianzas para Clasificacion_Vinos.csv (sin escalado) (PCA).



	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0.0017	0.0012	0.0169	0.1414	-0.0203	0.1941	0.9233	0.2848	-0.0866	-0.0022	-0.0150	-0.0157	0.0080
2	-6.8102e-04	0.0022	0.1220	0.1604	0.6129	0.7425	-0.1501	-0.0647	-0.0157	-0.0185	-0.0232	0.0673	-0.0111
3	1.9491e-04	0.0046	0.0520	-0.0098	-0.0202	0.0418	0.0450	-0.1493	-0.0736	-0.0868	0.9540	-0.1321	-0.1737
4	-0.0047	0.0265	0.9386	-0.3310	-0.0644	-0.0241	0.0315	0.0152	-0.0020	0.0036	-0.0528	0.0054	0.0019
5	0.0179	0.9993	-0.0298	-0.0054	0.0061	-0.0019	0.0018	-0.0036	0.0020	-4.0515e-05	-0.0030	6.2089e-04	0.0023
6	9.8983e-04	8.7796e-04	-0.0405	-0.0746	-0.3152	0.2787	-0.0202	-0.1772	-0.2557	0.8472	0.0088	0.0039	-0.0267
7	0.0016	-5.1851e-05	-0.0854	-0.1691	-0.5248	0.4336	-0.0389	-0.2481	-0.3783	-0.5201	-0.1332	-0.0375	0.0696
8	-1.2309e-04	-0.0014	0.0135	0.0108	0.0296	-0.0220	-0.0047	0.0065	-0.0368	0.0377	0.1992	0.1476	0.9665
9	6.0061e-04	0.0050	-0.0247	-0.0501	-0.2512	0.2419	-0.3098	0.8704	0.0515	0.0097	0.1356	-0.0131	-0.0176
10	0.0023	0.0151	0.2914	0.8789	-0.3317	0.0027	-0.1128	-0.0813	0.0990	-0.0231	-0.0098	0.0504	-0.0046
11	1.7138e-04	-7.6267e-04	-0.0260	-0.0600	-0.0515	-0.0238	0.0308	-0.0030	-0.0331	-0.0385	0.0975	0.9756	-0.1666
12	7.0493e-04	-0.0035	-0.0703	-0.1782	-0.2606	0.2889	0.1020	-0.1867	0.8737	0.0170	0.0285	0.0116	0.0442
13	0.9998	-0.0178	0.0045	-0.0031	0.0023	-0.0012	-0.0011	1.0341e-05	7.2559e-05	4.9266e-05	-2.4045e-04	-1.0000e-04	3.6267e-05

Figura 6: Matriz de coeficientes para Clasificacion_Vinos.csv (sin escalado) (PCA).

El método usado es [rescaling de la función normalize de matlab](#). Con esto se logra que al método de PCA le resulte más fácil tratar la información sin deformarla. Esto nos permite analizar la representatividad real de aquellos valores que eran demasiado grandes por naturaleza. Lo cual resulta en que cada PCA sea más relevante de cara al dataset final.



	1	2	3	4	5	6	7	8	9	10	11	12	13
1	36.1988	19.2075	11.1236	7.0690	6.5633	4.9358	4.2387	2.6807	2.2222	1.9300	1.7368	1.2982	0.7952

Figura 7: Vector de varianzas escaladas para Clasificacion_Vinos.csv (PCA).

coeff 13x13 double													
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0.1443	0.4837	-0.2074	-0.0179	-0.2657	0.2135	-0.0564	0.3961	-0.5086	0.2116	-0.2259	-0.2663	0.0150
2	-0.2452	0.2249	0.0890	0.5369	0.0352	0.5368	0.4205	0.0658	0.0753	-0.3091	0.0765	0.1217	0.0260
3	-0.0021	0.3161	0.6262	-0.2142	-0.1430	0.1545	-0.1492	-0.1703	0.3077	-0.0271	-0.4987	-0.0496	-0.1412
4	-0.2393	-0.0106	0.6121	0.0609	0.0661	-0.1008	-0.2870	0.4280	-0.2004	0.0528	0.4793	-0.0557	0.0917
5	0.1420	0.2996	0.1308	-0.3518	0.7270	0.0381	0.3229	-0.1564	-0.2714	0.0679	0.0713	0.0622	0.0568
6	0.3947	0.0650	0.1462	0.1981	-0.1493	-0.0841	-0.0279	-0.4059	-0.2860	-0.3201	0.3043	-0.3039	-0.4639
7	0.4229	-0.0034	0.1507	0.1523	-0.1090	-0.0189	-0.0607	-0.1872	-0.0496	-0.1632	-0.0257	-0.0429	0.8323
8	-0.2985	0.0288	0.1704	-0.2033	-0.5007	-0.2586	0.5954	-0.2333	-0.1955	0.2155	0.1169	0.0424	0.1140
9	0.3134	0.0393	0.1495	0.3991	0.1369	-0.5338	0.3721	0.3682	0.2091	0.1342	-0.2374	-0.0956	-0.1169
10	-0.0886	0.5300	-0.1373	0.0659	-0.0764	-0.4186	-0.2277	-0.0338	-0.0562	-0.2908	0.0318	0.6042	-0.0120
11	0.2967	-0.2792	0.0852	-0.4278	-0.1736	0.1060	0.2321	0.4366	-0.0858	-0.5224	-0.0482	0.2592	-0.0899
12	0.3762	-0.1645	0.1660	0.1841	-0.1012	0.2659	-0.0448	-0.0781	-0.1372	0.5237	0.0464	0.6010	-0.1567
13	0.2868	0.3649	-0.1267	-0.2321	-0.1579	0.1197	0.0768	0.1200	0.5758	0.1621	0.5393	-0.0794	0.0144

Figura 8: Matriz de coeficientes escalados para Clasificacion_Vinos.csv (PCA).

Análisis y la categorización

Diferentes catas realizadas a diferentes tipos de Vino Tinto, con los atributos asociados a la química del Vino. Archivo Clasificación_Vinos.csv

	PCA1	PCA2	PCA3	PCA4	PCA5	PCA6	PCA7	PCA8	PCA9	PCA10	PCA11	PCA12	PCA13
Alcohol	0,144329 3954	0,48365154 78	0,20738262 41	0,01785630 146	0,26566365 29	0,21353864 8	0,0563963 5803	0,3961392 585	0,5086191 163	0,2116047 271	0,2259169 621	0,266286 4508	0,014969 97064
Malic_Acid	0,245187 5803	0,22493093 46	0,08901288 566	0,53689028 35	0,03521362 829	0,53681385 093	0,4205239 436	0,0658267 441	0,0752830 436	0,3090799 436	0,0764855 3925	0,121696 0354	0,025963 74608
Ash	0,002051 061444	0,31606881 4	0,62622390 09	0,21417556 22	0,14302547 38	0,15447466 47	0,1491706 113	0,1702600 209	0,3076944 453	0,0271253 8863	0,4986914 151	0,049622 37326	0,141218 0318
Ash_Alcanity	0,239320 4055	0,01059050 229	0,61208034 99	0,06085941 446	0,06610294 157	0,10082450 51	0,2869691 357	0,4279701 829	0,2004493 139	0,0527994 2387	0,4793137 801	0,055742 86909	0,091682 85458
Magnesium	0,141992 042	0,29963400 32	0,13075693 49	0,35179658 01	0,72704851 48	0,03814393 991	0,3228833 026	0,1563614 329	0,2714025 723	0,0678702 2282	0,0712889 1466	0,062220 10539	0,056774 22188
Total_Phenols	0,394660 8451	0,06503951 182	0,14617896 35	0,19806834 79	0,14931840 93	0,08412229 982	0,0279249 7529	0,4059340 856	0,2860345 201	0,3201313 522	0,3043411 879	0,303882 4511	0,463907 9139
Flavanoids	0,422934 2967	0,00335981 21	0,15068189 99	0,15229479 46	0,10902583 84	0,01892002 309	0,0606852 1421	0,1872453 565	0,0495784 9185	0,1631505 098	0,0256940 9349	0,042898 82872	0,832257 0647
Nonflavanoid_Phenols	0,298533 103	0,02877948 811	0,17036816 24	0,20330101 99	0,50070298 32	0,25859401 1	0,5954472 892	0,2332846 492	0,1955013 195	0,2155350 703	0,1168958 631	0,042352 18869	0,114039 8497
Proanthocyanins	0,313429 4883	0,03930172 229	0,14945430 95	0,39905653 29	0,13685982 12	0,53379538 85	0,3721393 457	0,3682267 503	0,2091448 677	0,1341839 653	0,2373625 02712	0,095553 0712	0,116917 0672
Color_Intensity	0,088616 70472	0,52999567 21	0,13730621 25	0,06592568 164	0,07643677 955	0,41864414 15	0,2277121 402	0,0337969 1689	0,0562175 2277	0,2907751 84	0,0318387 9825	0,604221 6257	0,011992 80247
Hue	0,296714 5636	0,27923514 79	0,08522192 251	0,42777140 94	0,17361452 21	0,10598274 25	0,2320756 414	0,4366236 249	0,0858283 9469	0,5223988 87	0,0482120 0957	0,259213 9953	0,089888 84454
OD280	0,376167 4107	0,16449619 28	0,16600458 81	0,18412074 28	0,10116098 89	0,26585107 45	0,0447637 0074	0,0781078 9231	0,1372269 019	0,5237058 683	0,0464232 9747	0,600958 7206	0,156718 1347
Proline	0,286752 2269	0,36490283 18	0,12674591 73	0,23207086 04	0,15786879 99	0,11972557 2	0,0768045 046	0,1200226 69	0,5757861 075	0,1621159 976	0,5392698 277	0,079401 62173	0,014447 3367
Max	0,422934 2967	0,52999567 21	0,62622390 09	0,53689028 35	0,72704851 48	0,53681385 892	0,5954472 249	0,4366236 075	0,5757861 683	0,5237058 277	0,5392698 6257	0,604221 6257	0,832257 0647

Tabla 1: Matriz de coeficientes de Clasificacion_Vinos.csv, con los valores de cada variable PCA marcados en rojo.

Como se puede observar, la composición de PCA1 y PCA2 están muy repartidas. En concreto, están formadas mayoritariamente por 7 y 6 variables respectivamente, al contrario

que PCA3 que está formada mayoritariamente por 2. Esto significa que casi todo el dataset está representado en tan solo las dos primeras columnas.

En lo que respecta al resto de columnas, la gran mayoría está representada en buena parte por 3 o más variables. Por lo que todo está bastante repartido. Sin embargo tras la visualización (se puede observar en más detalle en cualquier archivo 3D de los resultados de la red) parece que encontramos 3 clusters que creemos que son el resultado.

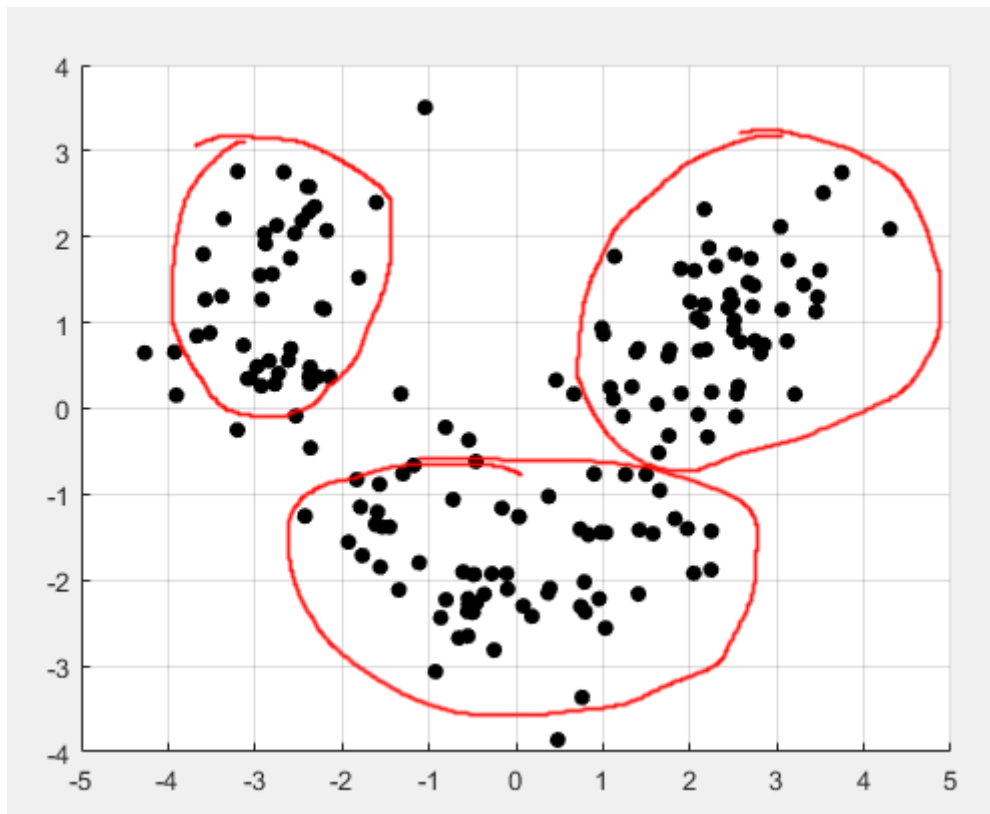


Figura 9: *Clusters visibles*



Figura 10: "epsilon_a": 0.2, "epsilon_n": 0.05, "a_max": 8, "eta": 8, "alpha": 0.1, "delta": 0.1, "epoch": 5
 vinos_todas-PCA_3-escalado-a_0.2-amax_8-n_0.05-e_8-30-12-2021-18-43-56
Entrenada con PCA1, PCA2 y PCA3 (3D)

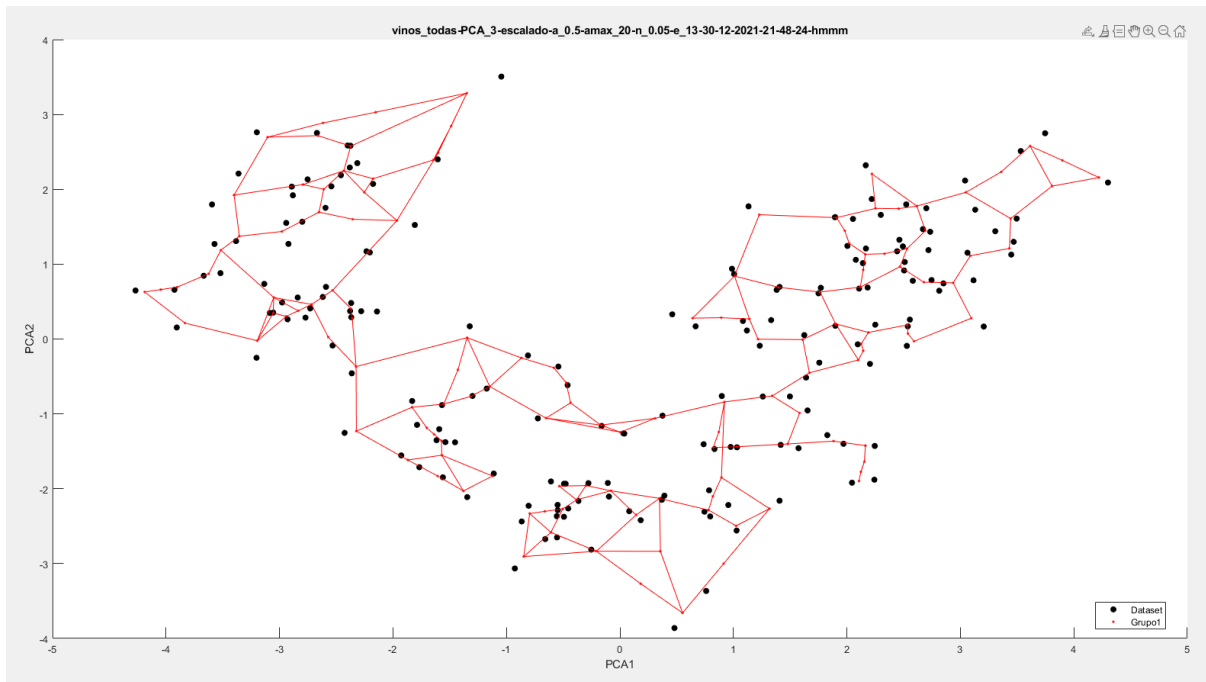


Figura 11: "epsilon_a": 0.5, "epsilon_n": 0.05, "a_max": 20, "eta": 13, "alpha": 1.1, "delta": 0.6, "epoch": 10
 vinos_todas-PCA_3-escalado-a_0.5-amax_20-n_0.05-e_13-30-12-2021-21-48-24-hmmm
 Entrenada con PCA1 y PCA2 (2D)

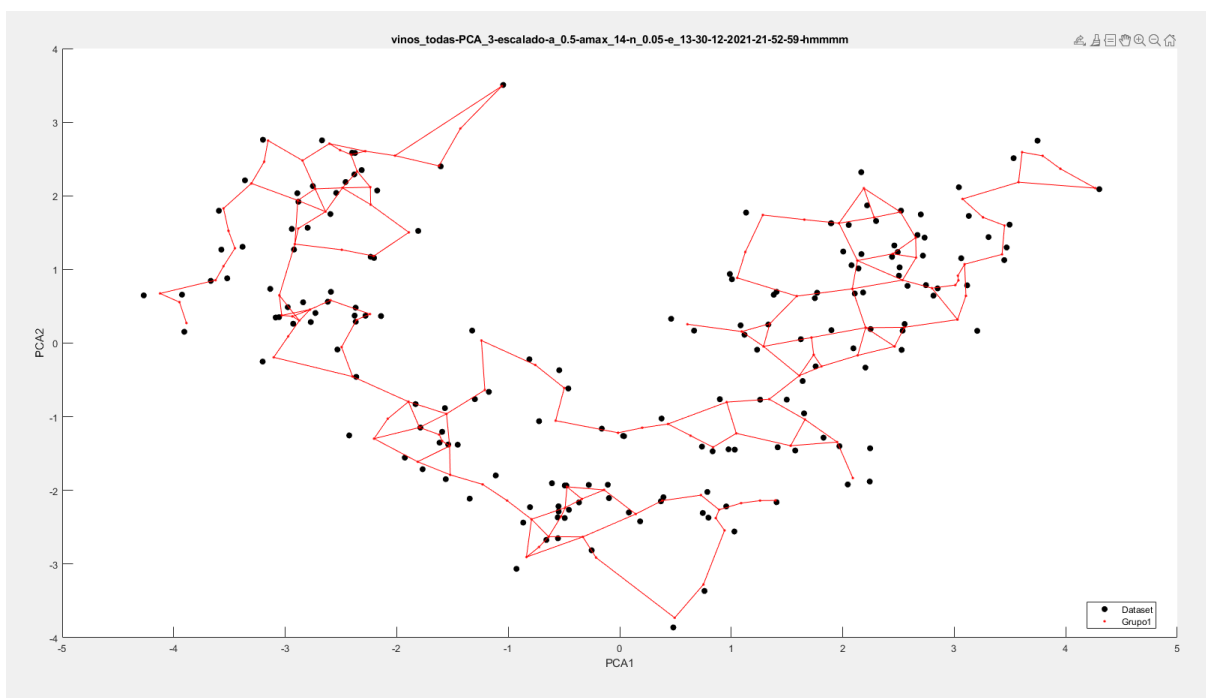


Figura 12: "epsilon_a": 0.5, "epsilon_n": 0.05, "a_max": 14, "eta": 13, "alpha": 1.1, "delta": 0.6, "epoch": 10
 vinos_todas-PCA_3-escalado-a_0.5-amax_14-n_0.05-e_13-30-12-2021-21-52-59-hmmm
 Entrenada con PCA1 y PCA2 (2D)

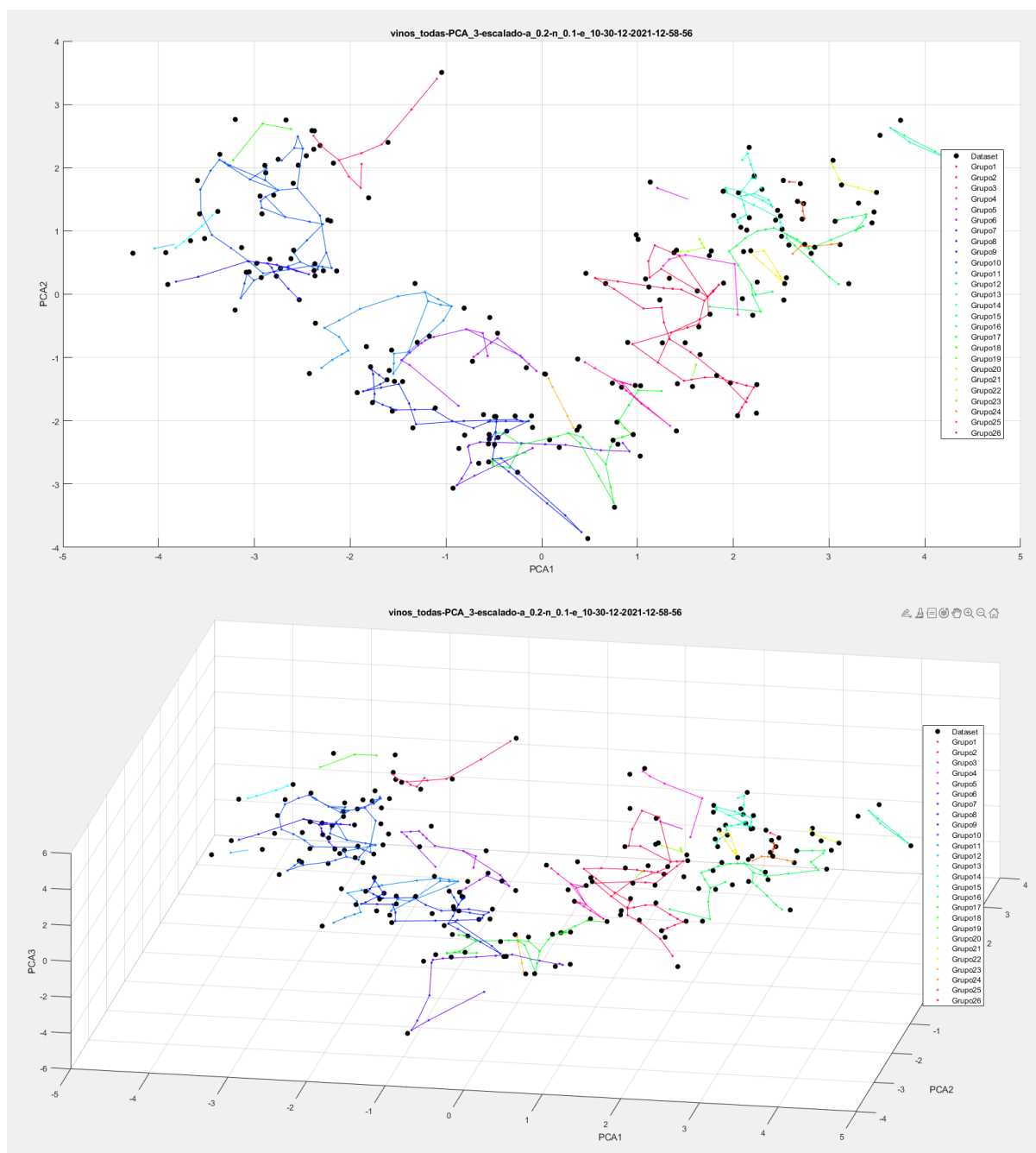


Figura 13: "epsilon_a": 0.2, "epsilon_n": 0.1, "a_max": 10, "eta": 10, "alpha": 0.1, "delta": 0.1, "epoch": 20
 vinos_todas-PCA_3-escalado-a_0.2-n_0.1-e_10-30-12-2021-12-58-56
 Entrenada con PCA1, PCA2 y PCA3 (3D)

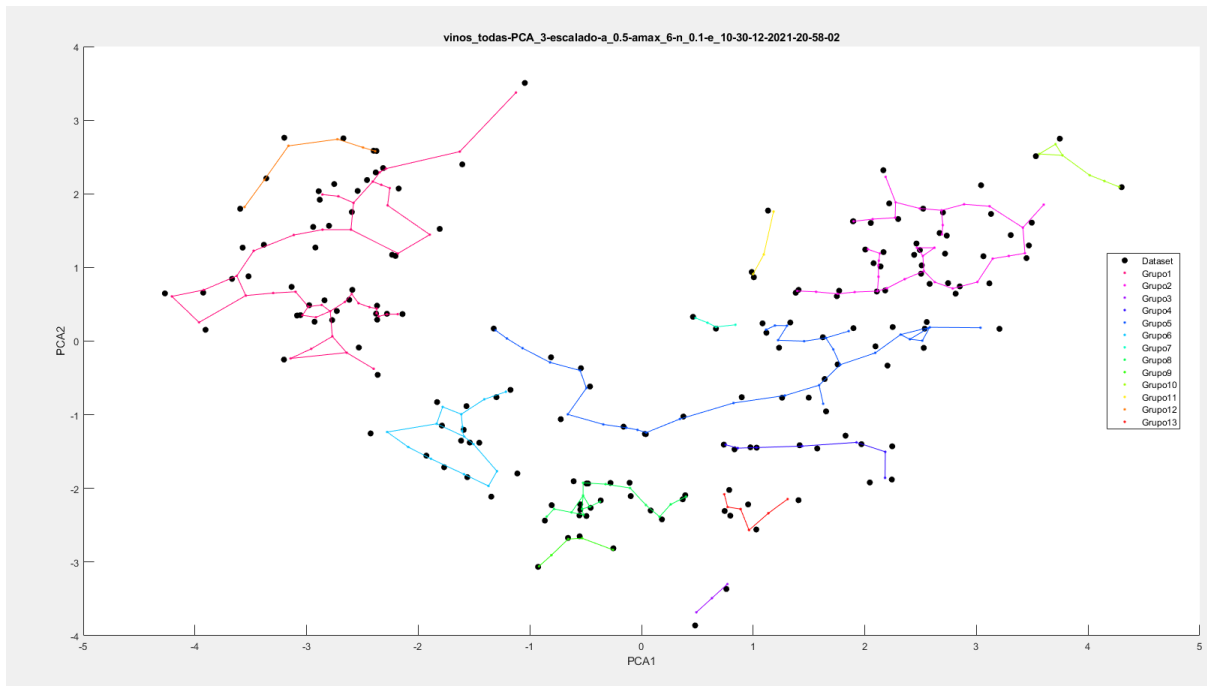


Figura 14: "epsilon_a": 0.5, "epsilon_n": 0.1, "a_max": 6, "eta": 10, "alpha": 0.5, "delta": 0.1, "epoch": 10
 vinos_todas-PCA_3-escalado-a_0.5-amax_6-n_0.1-e_10-30-12-2021-20-58-02
 Entrenada con PCA1 y PCA2 (2D)

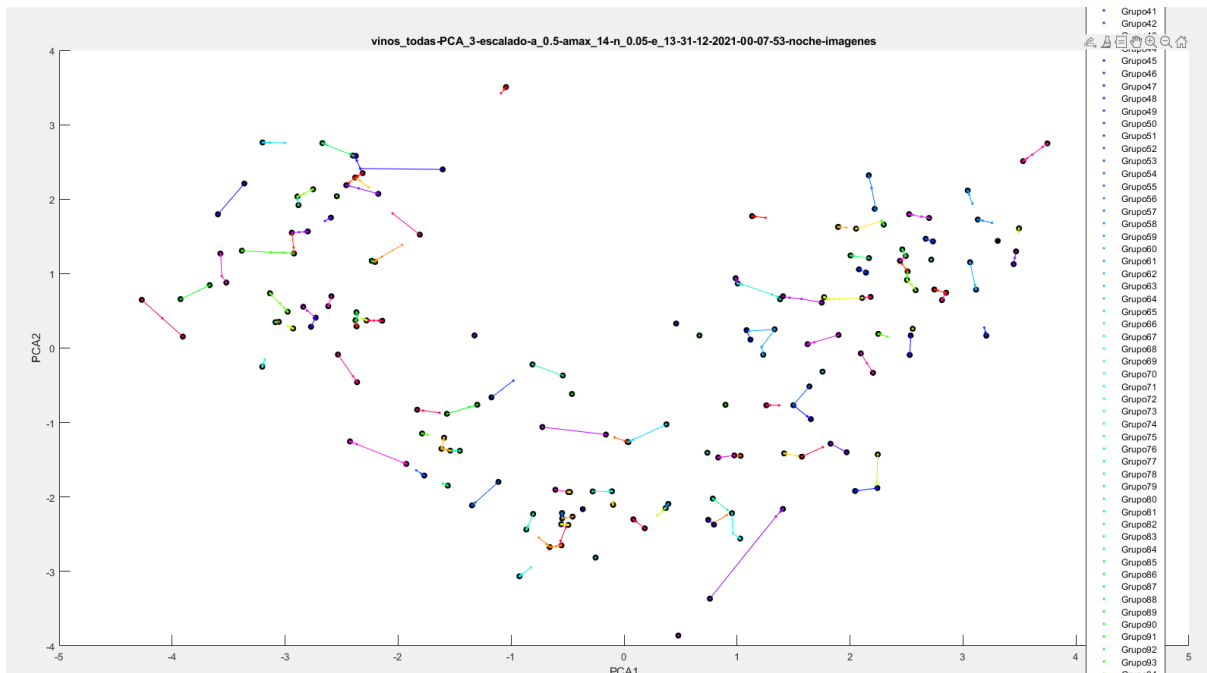


Figura 15: "epsilon_a": 0.5, "epsilon_n": 0.05, "a_max": 14, "eta": 13, "alpha": 1.1, "delta": 0.6, "epoch": 160
 vinos_todas-PCA_3-escalado-a_0.5-amax_14-n_0.05-e_13-31-12-2021-00-07-53-noche-imagenes
 Entrenada con PCA1 y PCA2 (2D)

En total realizamos más de 20 pruebas, pero sólo nos parecieron interesantes las que mostramos.

- Empecemos por la peor de todas, la **figura 15**. Esta figura surgió porque queríamos conocer los límites de la red y ver dónde ocurría el overfitting, probamos 80 epochs y seguía dando resultados que si bien no eran buenos, no se parecían a esta imagen, así que decidimos probar con 160 epochs. Claramente vemos que la red hace overfitting y hace los grupos lo más pequeños posibles.
- Continuemos con la figura **10**, podemos ver un buen comportamiento en la red, pero sólo considera que haya un grupo, lo cual debe estar mal, pero empezamos a ver como la red aprende que la conexión entre el grupo inferior y el izquierdo es muy ligera, sin embargo no poda esta rama, el grupo derecho sin embargo está más cerca y por tanto sus conexiones son mayores.
- La siguiente figura es la **11**, aquí vemos que la red intenta meter todos los datos dentro del grupo, probablemente le sea más fácil al ser los datos 2D y no 3D.
- Tras esto tenemos **12** que es en esencia lo mismo que **12**, pero aquí la red se atreve a cortar más gracias al parámetro de la edad que es menor.
- Finalmente en **13** y **14** la red por fin separa los grupos, tras haber disminuido todos los parámetros. El problema que se presenta a continuación es que crea más grupos de los que creemos que debe, sin embargo, al menos en la figura **13** intentan mantener la forma, en la **14** hay algunos grupos pequeños que parecen correctos y luego restos de “fideos”. Sería algo plausible eliminar los fideos manualmente, ya que claramente una línea de puntos no es un grupo, sino conexiones que se quedan sueltas tras podar. Lo que nos dejaría con aproximadamente 4 grupos en vez de los 3 que habíamos planteado. También está la diferencia de que **14** representa un plano y **13** un espacio.

La conclusión que sacamos es que los valores que buscamos están entre los de **13** y **14** por ello probamos más entrenamientos con valores intermedios, a ver qué resultados obtenemos.

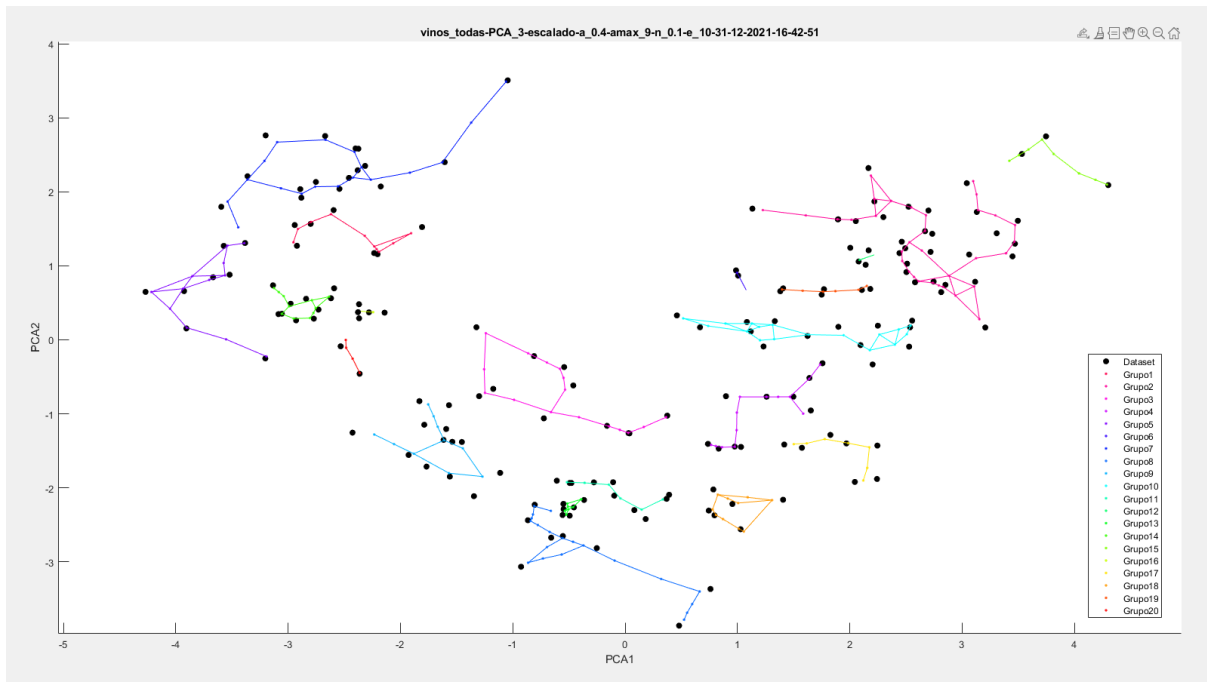


Figura 16: "epsilon_a": 0.4, "epsilon_n": 0.1, "a_max": 9, "eta": 10, "alpha": 0.4, "delta": 0.1, "epoch": 10
 vinos_todas-PCA_3-escalado-a_0.4-amax_9-n_0.1-e_10-31-12-2021-16-42-51

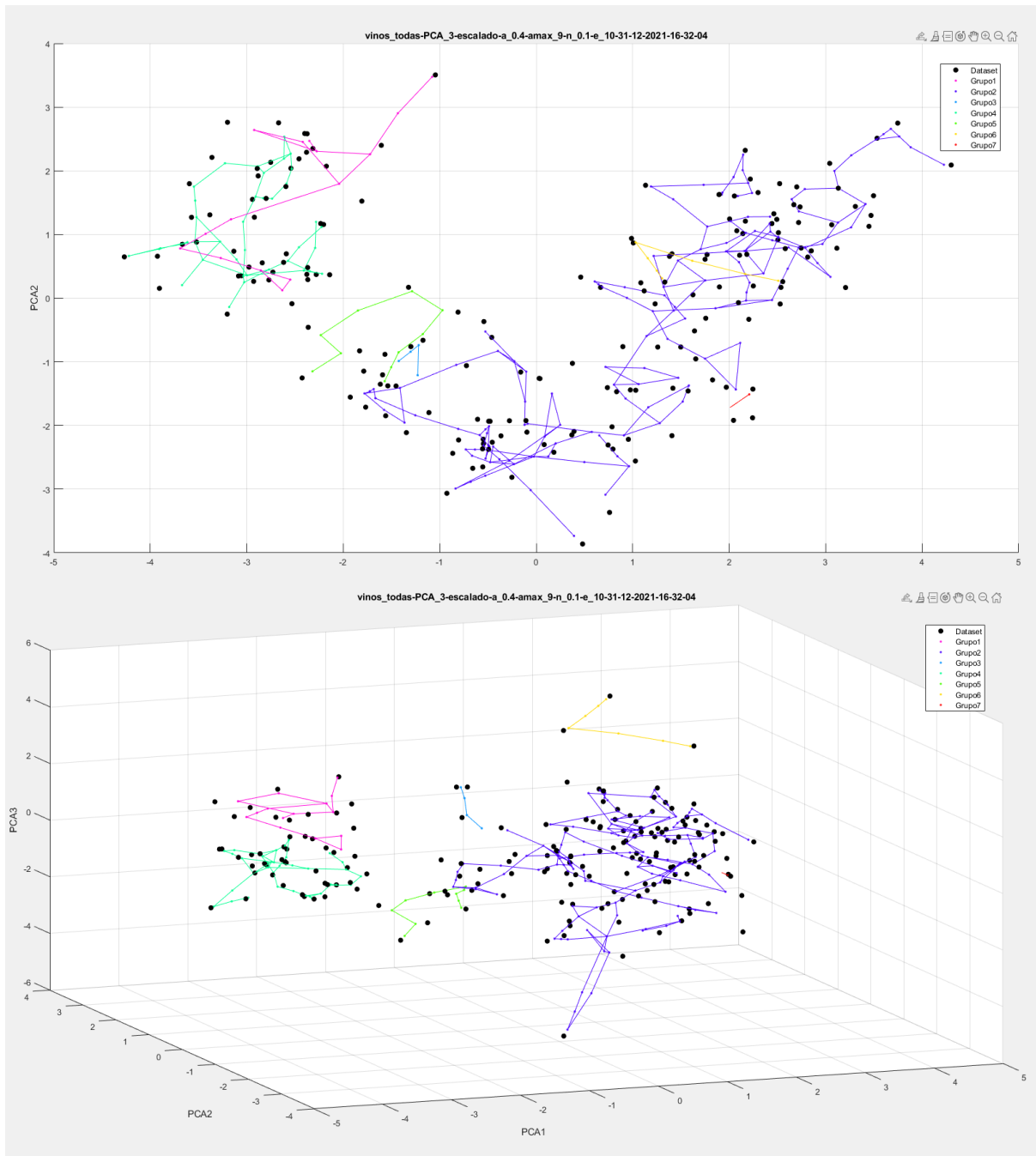


Figura 17: "epsilon_a": 0.4, "epsilon_n": 0.1, "a_max": 9, "eta": 10, "alpha": 0.4, "delta": 0.1, "epoch": 10
 vinos_todas-PCA_3-escalado-a_0.4-amax_9-n_0.1-e_10-31-12-2021-16-32-04

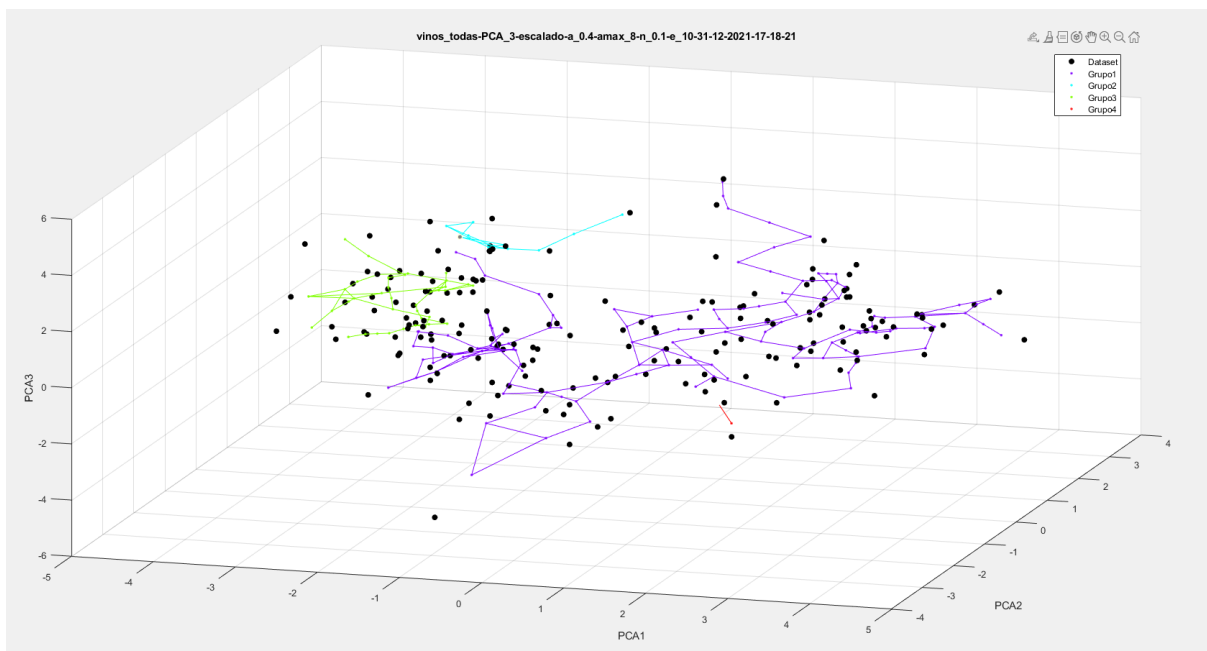
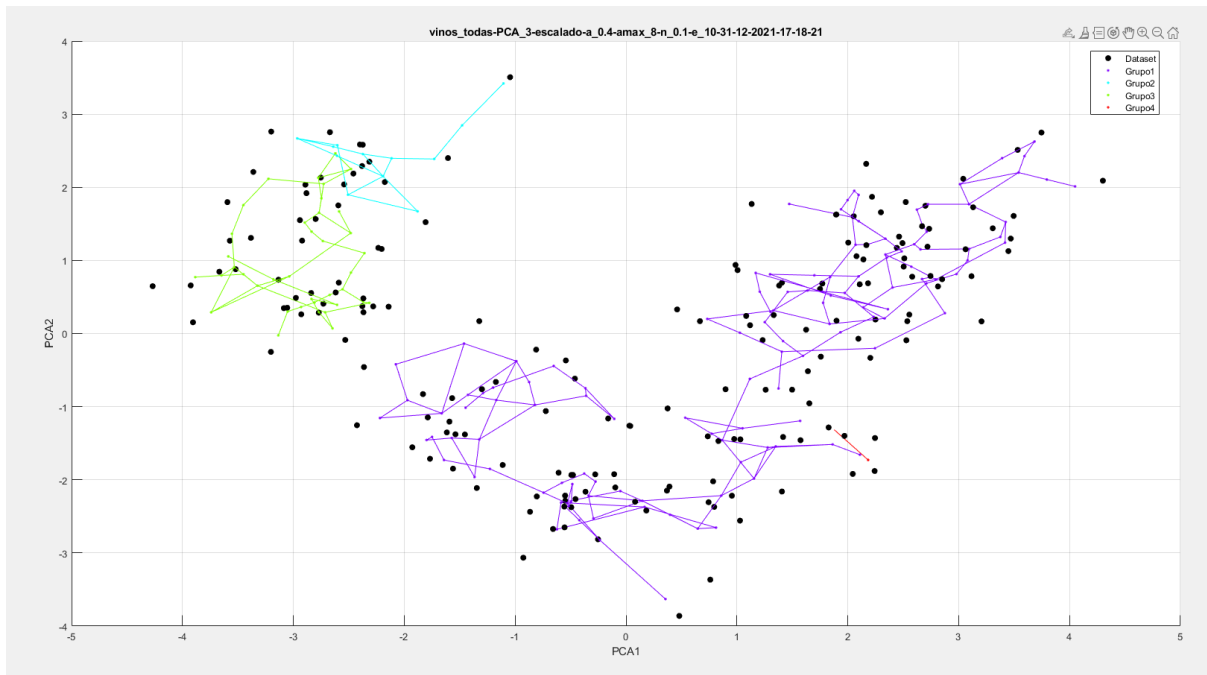


Figura 18: "epsilon_a": 0.4, "epsilon_n": 0.1, "a_max": 8, "eta": 10, "alpha": 0.4, "delta": 0.1, "epoch": 10
 vinos_todas-PCA_3-escalado-a_0.4-amax_8-n_0.1-e_10-31-12-2021-17-18-21



Figura 19: "epsilon_a": 0.5, "epsilon_n": 0.05, "a_max": 8, "eta": 10, "alpha": 0.4, "delta": 0.1, "epoch": 10
 vinos_todas-PCA_3-escalado-a_0.5-amax_8-n_0.05-e_10-31-12-2021-17-30-02

Para acabar finalmente, subiendo y bajando valores hemos llegado a algo que parece tener cierta forma, sin embargo, no se desconectan los grupos que habíamos propuesto. Puede ser que nos hayamos equivocado, sin embargo, creemos que esto se debe al tamaño del dataset junto a cierta insistencia por parte de la red como decíamos en la **figura 2** al principio del informe.

Como ejemplo del efecto de tener un dataset muy pequeño vamos a usar el paper que vimos en clase, donde se ejemplificaba la red con una forma claramente definida (por lo que no hay puntos orbitando el cluster), donde dicha forma viene dada por una ecuación, y que por tanto contendría una cantidad “infinita” de puntos en caso de que sean necesarios.

En el mismo paper la figura devuelta por la red comienza a parecerse a la original a partir de las 5000 entradas. Nosotros estábamos limitados a 10 épocas, ya que subir este número en nuestro caso no mejoraba el resultado sino que lo empeoraba. Si multiplicamos nuestras muestras por las épocas nos quedamos cortos con 1780 iteraciones, por lo que la única forma de conseguir más iteraciones sería aumentando las muestras.

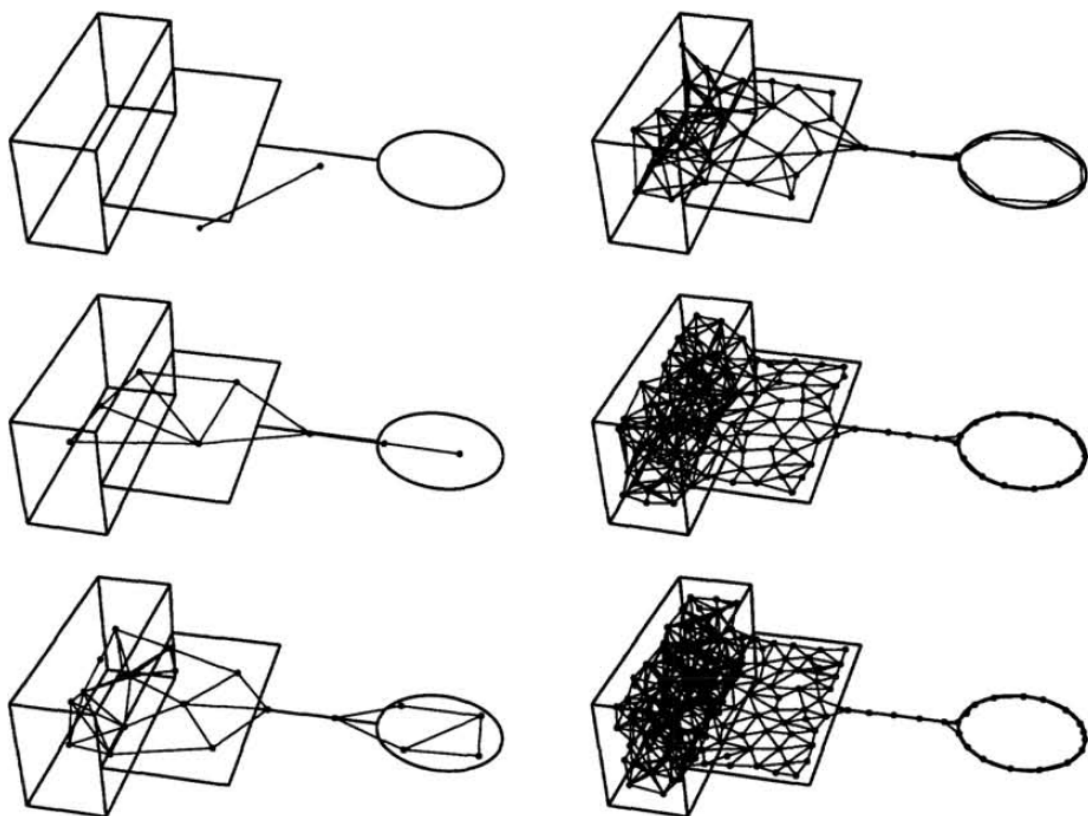


Figura 20: Paper PDF, página 7, encabezado 630, figure 2

“growing neural gas”
 (uses “competitive Hebbian learning”)

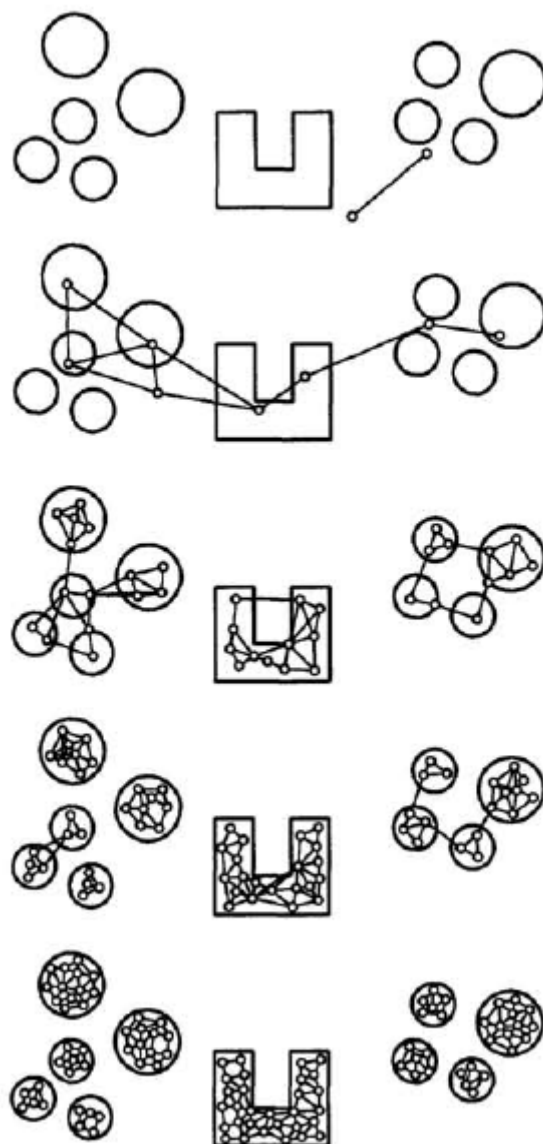


Figura 21: Paper PDF, página 8, encabezado 631, figure 3