Universidad Francisco de Vitoria
UFV Madrid

Asignatura:

Inteligencia Artificial II

Título del documento:

Laboratorio 01

Preparado por:

Nombre	Gonzalo Monedero Nicolás Salgado Iñigo Vázquez Marcos Molina	21/02/2022

1 Índice

2 Introducción

En la siguiente práctica aplicaremos los conocimientos teóricos adquiridos en clase relativos a los mapas autoorganizativos de Kohonen a un entorno práctico. De esta manera, buscamos poder aprender realmente cómo funcionan estos mapas y poder imaginar aplicaciones suyas del día a día. La clasificación automatizada que realizan estos mapas puede resultar muy útil a la hora del estudio de datos y la toma de conclusiones.

No solo eso, con esta práctica también buscamos el aprender a tomar decisiones de diseño cruciales para estos mapas, como el lado del mapa, el período o el coeficiente de aprendizaje. Estas variables cambian totalmente el mapa y como de funcional y preciso puede llegar a ser. Para saber cómo de preciso es, emplearemos conocimientos teóricos como el del cálculo del error topológico y el error cuantificable.

$$\bar{d} = \frac{1}{N}\sum_{i=1}^{N}\|x_i - bmu\| \qquad \bar{t} = \frac{1}{N}\sum_{i=1}^{N} t(x_i)$$

Error cuantificable **Error topológico**

El laboratorio 1 este compuesto por dos prácticas que analizaremos por partes e independientemente. Esta memoria es la perteneciente al grupo 1 de IA II de 3º Ingeniería Informática B, compuesto por Gonzalo Monedero, Nicolás Salgado, Iñigo Vázquez y Marcos Molina.

3 Práctica 1

3.1 Introducción

En la práctica 1 se nos pide elaborar un mapa autoorganizativo de colores (SOM – Self Organising Map). En la foto adjuntada se muestra cómo funciona y cuál es la salida de esta red.

Tras realizar este mapa autoorganizativo de colores, se nos presentan 3 cuestiones que resolveremos en los siguientes apartados:

3.2 Cuestiones

3.2.1 Cuestión 1

¿Cuáles son los valores de Lado Mapa, Periodo y Eta más adecuados? Realiza gráficas como las vistas en teoría para justificar tu selección de los valores concretos de estos parámetros y explica el motivo de tu elección.

En esta sección, nosotros hemos querido ir punto por punto sabiendo cual es el valor más adecuado. Es mejor ir valor por valor puesto que resulta mucho más difícil saber cuáles son los tres valores más adecuados combinándolos entre sí, ya que la salida final es una combinación de los tres. Es por ello por lo que primero sabremos cual es el lado del mapa más adecuado, luego el Eta y por último el periodo.

Para saber si un valor es adecuado o no, miraremos como influye el cambio de ese valor respecto a: N.º de clases generadas por la red, el error cuantificable y el error topológico. Queremos en este caso, maximizar el N.º de clases, y minimizar los errores, con el menor gasto de recursos posible.

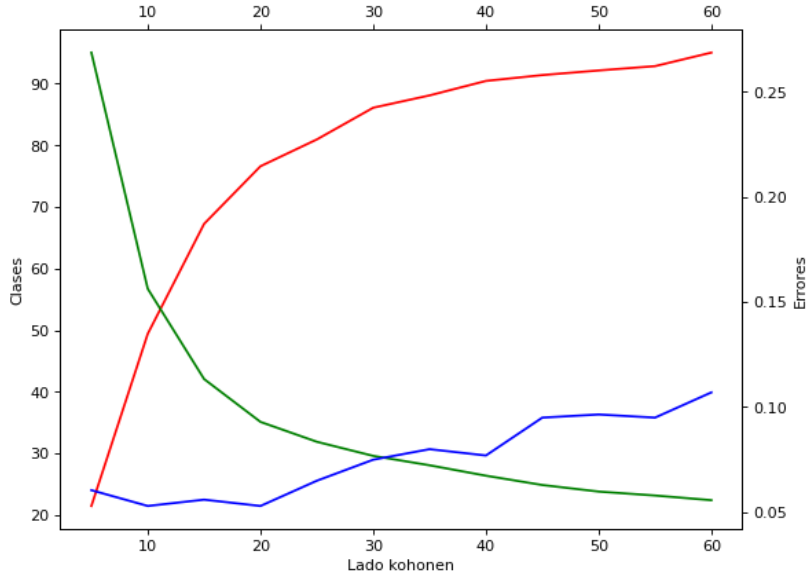
Para realizar este estudio hemos creado un automatizado de ejecución que nos informe del resultado de estas variables según el valor a ejecutar, $AutomatizadorColores.ipynb$

Para realizar las siguientes gráficas empleamos parte de código de: (StackOverflow, s.f.)

3.2.1.1 Lado mapa adecuado

Para el SOM, queremos optimizar el lado de Kohonen para que con un menor lado haya el máximo número de clases. El error cuantificable debe de ser bajo, ya que este son las medias de las distancias. Por tanto, a menor media quiere decir que las BMUs se aproximan más al valor de entrada, funcionando correctamente así el SOM. Por otro lado, el topológico debe de ser bajo también ya que este nos dice si la siguiente BMU más baja de cada neurona es adyacente a la ganadora. Esto nos verificara que el mapa es homogéneo y que se ha realizado una buena clasificación.

Como bien hemos comentado, primero estudiaremos el lado del mapa de Kohonen variando el mapa de lados de 3 hasta 60. Para realizar la gráfica que comentaremos, hemos realizado un Dataset de valores de ejecución para cada lado del mapa. Cada lado se ejecuta 20 veces y se hace la media de los valores sacando así un set de 3 valores por cada lado probado. Hay que matizar que estos lados se prueban con el mismo periodo (1000) y el mismo Eta (0.2).Una vez tenemos el Dataset con los valores, pintamos dichos valores para ver cómo influye el lado de Kohonen en las variables comentadas:

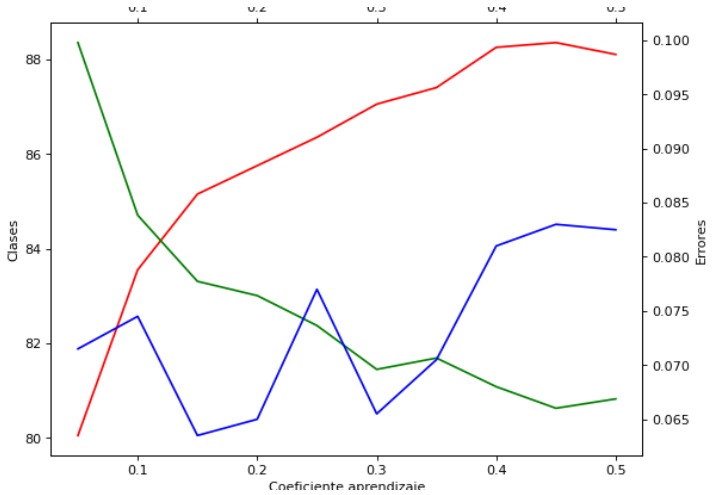


La línea roja representa como influye el lado de Kohonen en el N.º de clases. La línea verde en el error cuantificable y la línea azul en el error topológico

Tanto para el error de cuantificación como el número de clases las tendencias son asintóticas es por ello por lo que deberemos fijarnos más en el error topológico que como podemos observar a mayor número de lados mayor es. Como hemos dicho antes realmente aumentar el número de lados en una gran cantidad no nos interesa ya que no ganamos casi clases ni reducimos el error de cuantificación debido a la tendencia asintótica que tiene las dos líneas poligonales es por ello por lo que concluimos que 30 es una buena elección por su proximidad a las dos asíntotas del número de clases y el error de cuantificación, sin sacrificar un notable aumento en el error topológico.

3.2.1.2 Eta adecuada

Una vez sabemos cuál es el lado de Kohonen adecuado (30), realizamos la misma metodología que antes. Generamos un Dataset con valores entre 0.1-0.7 de Eta para ver cuál es el más adecuado. Cada valor lo repetimos 5 veces y sacamos la media de cada Eta para poder tener una representación mejor. Al igual que con el lado del mapa, pintamos los resultados para ver cuál es el coeficiente de aprendizaje más adecuado:

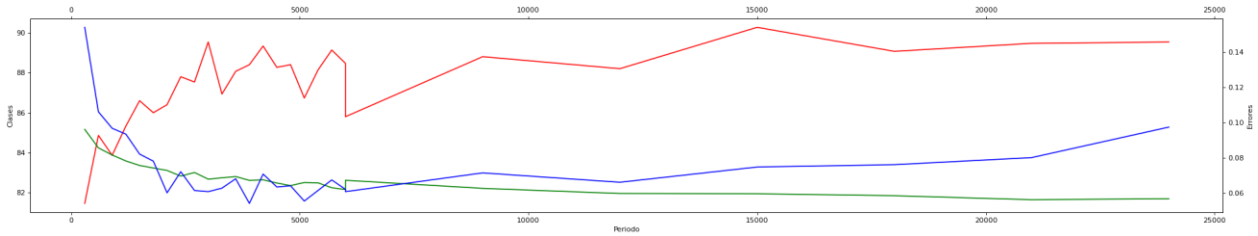


Al igual que en anterior gráfico, la línea roja representa como influye el Eta en el nº de clases. La línea verde en el error cuantificable y la línea azul en el error topológico.

Podemos ver que el único punto donde podemos maximizar el número de clases generado y minimizar ambos errores es el punto donde el Eta es 0.3. Pese a que haya puntos donde por ejemplo el error cuantificable es mucho más bajo (punto 0.5) vemos como el topológico se dispara. Queremos en nuestro caso tener los dos errores más bajos posibles y es por ello por lo que el 0.3 minimiza los dos.

3.2.1.3 Periodo adecuado

Por último, queremos saber el periodo adecuado. Seguiremos la misma metodología que en los anteriores apartados. Con estos datos, pintamos e interpretamos la gráfica para saber cuál es el periodo más adecuado:



Observando la gráfica podemos ver como los datos visualmente los mejores valores respecto a los errores (puesto que dentro de lo que cabe el número de clases no varía tanto) son aquellos entornos a las 5000 iteraciones. Es por ello por lo que el periodo adecuado es 5000 para nuestro SOM. Cogemos este número para redondear, ya que como hemos mencionado los mejores outputs están en base a este número, y es que al ser salidas que varían, debemos de coger un número medio, ya que la diferencia que puede suponer de 5000 a 5250 iteraciones es mínima.

3.2.1.4 Conclusión

Como hemos podido ver en los anteriores apartados gracias a las gráficas, llegamos a la conclusión de que el correcto funcionamiento para este SOM es:

- Lado del mapa: 30
- Eta: 0.3
- Periodo: 5000

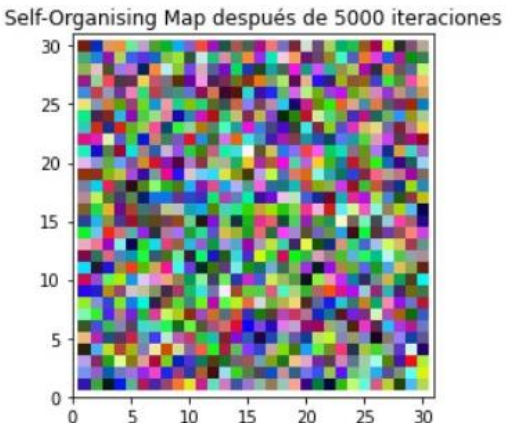
3.2.2 Cuestión 2

Para la mejor clasificación que hayas obtenido del Dataset de entrenamiento, adjunta la información generada en la ejecución y explica detalladamente que estamos viendo en cada uno de ellos.

Ya sabiendo el lado del mapa adecuado, la eta y el periodo, entre manos el SOM y realizamos la clasificación. Hay que matizar que los gráficos y valores que se mostraran a continuación son la salida de una ejecución. Al generar valores aleatorios para cada ejecución estas varían y no son siempre las mismas. Hemos cogido una ejecución aleatoria para realizar este apartado.

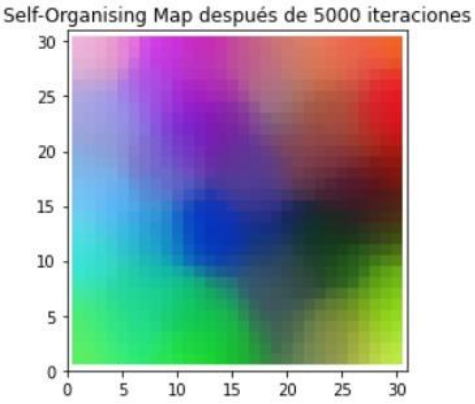
3.2.2.1 Gráfico RGB de pesos iniciales

En este caso, rellenamos nuestra matriz de pesos con valores aleatorios. Al pintarlos nos queda un mapa de colores desordenados.



3.2.2.2 Gráfico RGB de pesos entrenados

Estos colores, tras realizar el entrenamiento de la matriz, vemos como esta se ordena por espectros de colores. Matizar también como es visible el vecindario, viendo a grandes rasgos bolas de un mismo color por tonalidad.



3.2.2.3 Número de clases y errores

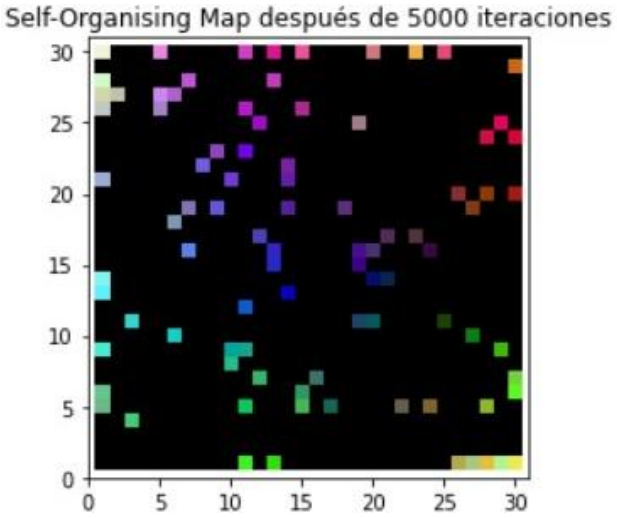
Tras realizar la clasificación de 100 colores aleatorios, realizamos los cálculos pertinentes para contar el número de clases y los respectivos errores.

```
Numero de Clases:  89
Error De Cuantificacion:  0.06492551190079071
Error Topologico:  0.04
```

3.2.2.4 Mapas

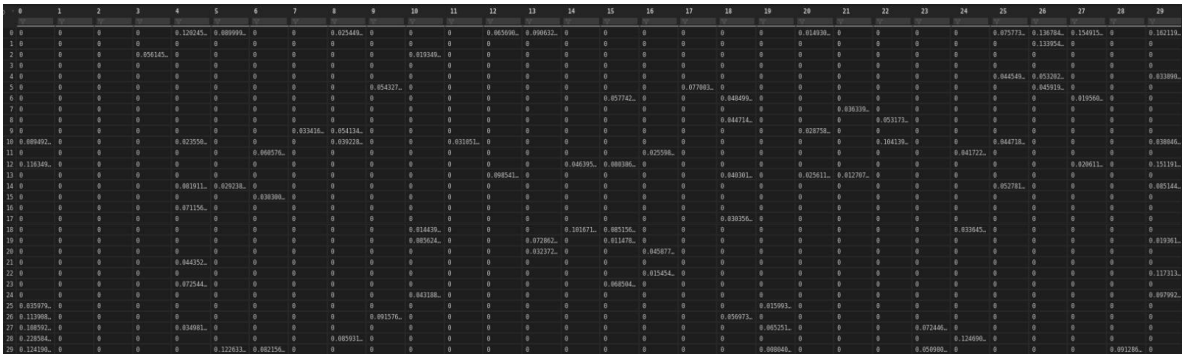
3.2.2.4.1 Mapa de clasificación

En el mapa de clasificación podemos ver el último patrón clasificado. Matizar como esta matriz coincide a grandes rasgos con la de pesos entrenada puesto que aquellas clasificadas estarán en posiciones parecidas al color de la matriz de pesos entrenada.



3.2.2.4.2 Mapa de distancias

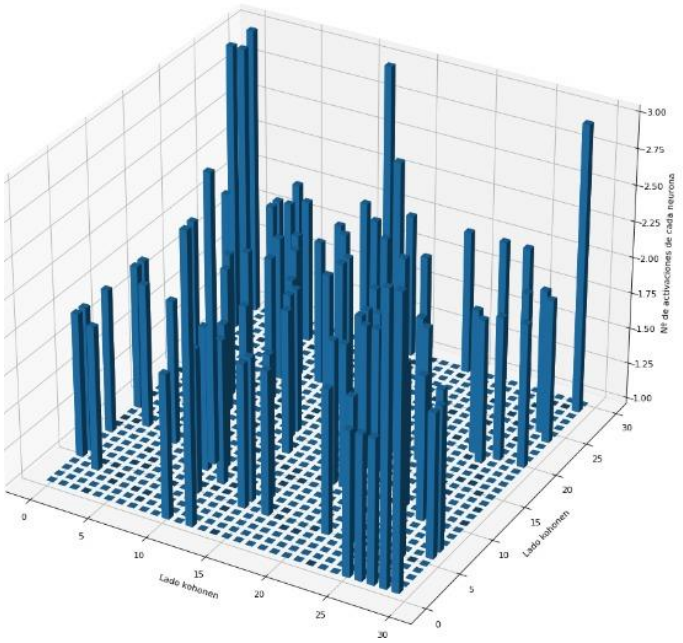
En el mapa de distancias observamos la distancia media del peso de una neurona activada con los patrones clasificados. Observamos pues como hay muchas neuronas con valores 0, ya que nunca se han activado. Posteriormente veremos el mapa de activación y comprobamos que las neuronas con valores en esta tabla son las activas en el histograma 3D del mapa de activación.



3.2.2.4.3 Mapa de activación

Aquí tenemos una representación 3D del mapa de neuronas activadas y de cuantas veces se ha activado cada neurona. Si las contamos, tendríamos las 90 clases mencionadas, es decir, 90 neuronas que se han activado al menos, una vez.

Para realizar este histograma nos hemos apoyado de los siguientes recursos: (Matplotlib , s.f.) (Programmerclick, s.f.)



3.2.3 Cuestión 3

Sin modificar el entrenamiento, clasifica el Dataset de prueba y adjunta los mismos resultados que en el apartado 2.3 para el siguiente Dataset:

[255,255,255][255,0,0][0,255,0][0,0,255][255,255,0][255,0,255][0,255,255][0,0,0]

Anteriormente la clasificación la realizábamos mediante un set de colores aleatorio:

```
colores_clasficiar = np.random.randint(valor_min, valor_max, (valores_color, 100))
colores_clasficiar = colores_clasficiar / 255
colores_clasficiar = np.transpose(colores_clasficiar)
```

Este set lo cambiamos pues por lo mencionado:

```
colores_clasficiar = [[255,255,255],[255,0,0],[0,255,0],[0,0,255],[255,255,0],[255,0,255],[0,255,255],[0,0,0]]
colores_clasficiar = np.squeeze(np.asarray(colores_clasficiar))
colores_clasficiar = colores_clasficiar / 255
```

Una vez tenemos este nuevo set de datos para clasificar, procedemos a mostrar la misma información que mostramos para la cuestión anterior:

3.2.3.1 Número de clases y errores

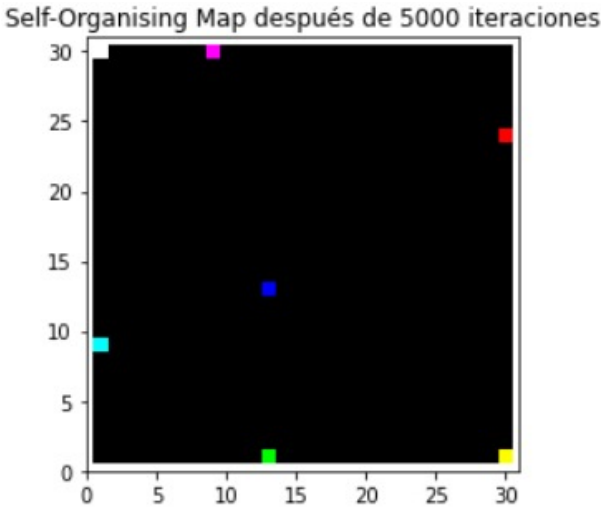
Vemos, como todos los colores que se nos adjuntan nuestro SOM los clasifica correctamente, viendo 8 clases y teniendo 8 entradas, por tanto, una clase para cada entrada. Respecto al error de cuantificación vemos como es más alto que en el entrenamiento. Esto tiene sentido, puesto que son colores muy dispares y con valores muy específicos, haciendo difícil que la distancia media sea baja. Por último, recalcar como el error topológico en este caso es nulo. El significado de esto es que primero de todo al ser pocas clases es probable que la 2nd BMU sea adyacente por probabilidad, pero no solo eso, si no que al ser valores dispares tener un error topológico bajo nos indica que nuestro mapa es muy homogéneo.

```
Numero de Clases:  8
Error De Cuantificacion:  0.1504745613331261
Error Topologico:  0.0
```

3.2.3.2 Mapas

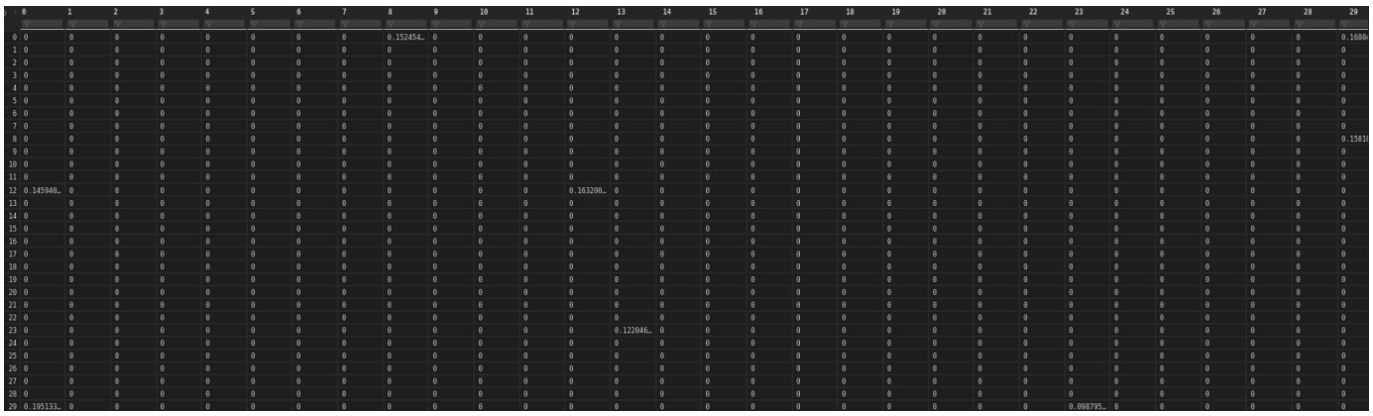
3.2.3.2.1 Mapa de clasificación

En el mapa de clasificación vemos como los 8 colores (el negro no se ve por el fondo) están correctamente representados, viendo como dentro de lo que cabe los colores parecidos tienen relativamente cerca (azules entre sí, verde cerca del amarillo y azul, rojo alejado de los azules…)



3.2.3.2.2 Mapa de distancias

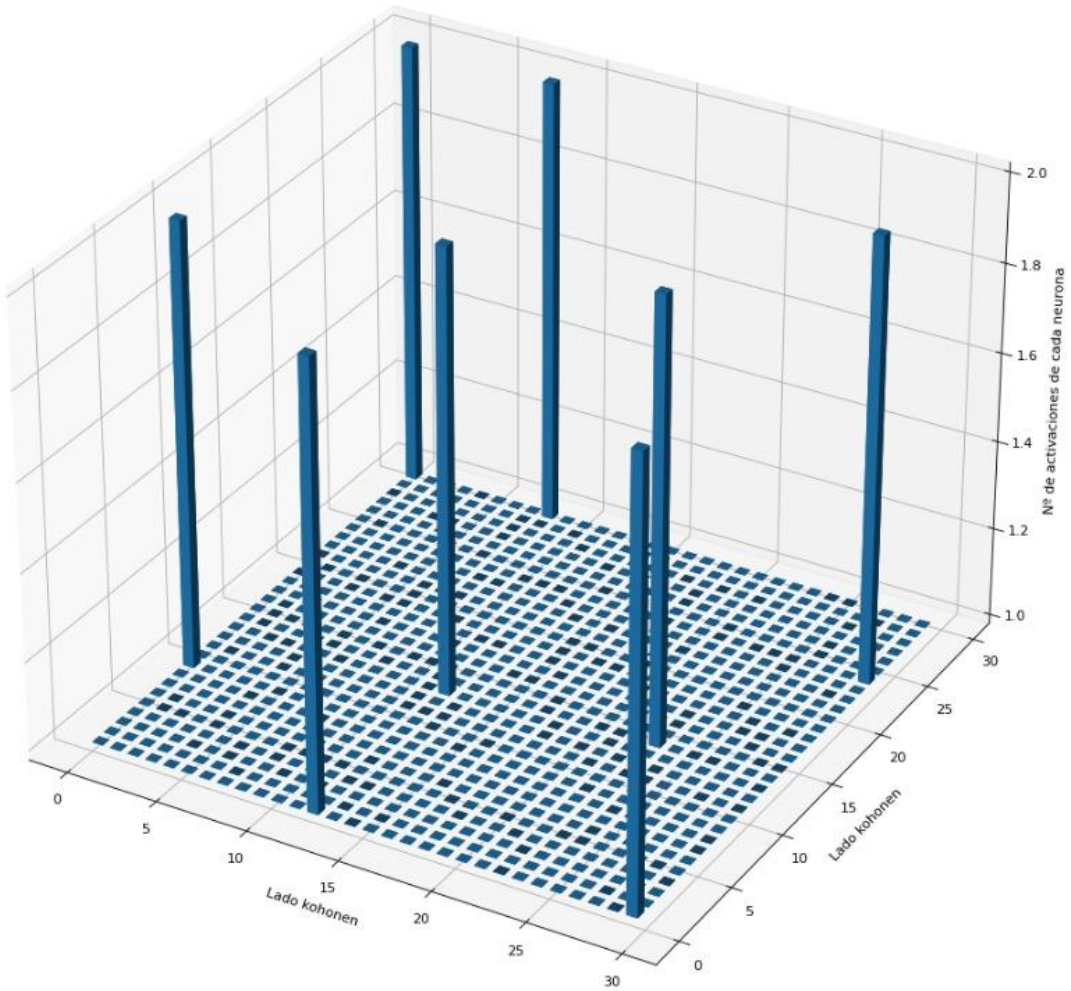
El mapa de distancias tiene como hemos visto distancias parecidas entre todo, alrededor de la media vista en el error de cuantificación (0.15). Esto nos indica que tenemos bien entrenado el mapa, no teniendo valores dispares.



3.2.3.2.3 Mapa de activación

Por último, vemos el mapa de activación. Cada neurona se activa solamente una vez, por cada color que le pasamos. Este mapa coincide totalmente con el mapa de clasificación, viendo como coinciden ambas matrices.

Para realizar este histograma nos hemos apoyado de los siguientes recursos: (Matplotlib , s.f.) (Programmerclick, s.f.)



4 Práctica 2

4.1 Introducción

En esta práctica el objetivo es el mismo, realizar un SOM, pero en este caso para Pokémon. Para realizar este SOM, empleamos como plantilla el SOM de colores, cambiando escasas variables para adaptarlo a las entradas de Pokémon.

El mapa autoorganizativo entonces tiene la misma metodología y el mismo funcionamiento, pero como es obvio, al cambiar las entradas cambian los valores. Es por ello por lo que realizaremos todas las cuestiones que hicimos para la práctica 1 pero aplicadas a este nuevo mapa autoorganizativo.

4.2 Cuestiones

4.2.1 Cuestión 1

¿Cuáles son los valores de Lado Mapa, Periodo y Eta más adecuados? Realiza gráficas como las vistas en teoría para justificar tu selección de los valores concretos de estos parámetros y explica el motivo de tu elección. Ten en cuenta que puede que no sea fácil separar totalmente todas las clases de Pokémon. (recuerda que algunos son de dos tipos).

Al igual que en la práctica 1, realizaremos las mismas gráficas y pruebas para saber los valores adecuados. Primero sacaremos el lado del mapa, luego el Eta y por último el periodo.

En este caso, también hemos realizado un Dataset con los valores de salida para cada cambio en las variables mencionadas. Este Dataset se genera mediante el notebook llamado $AutomatizadorPokemon.ipynb$, al igual que en la práctica anterior.

Matizar, por último, un dato importante que en el Dataset de entrenamiento de Pokémon existen 70 combinaciones únicas de tipos de Pokémon (teniendo en cuenta que hay combinaciones de dos tipos), por tanto, acercarnos a ese valor es una buena señal.

Para realizar las siguientes gráficas empleamos parte de código de: (StackOverflow, s.f.)

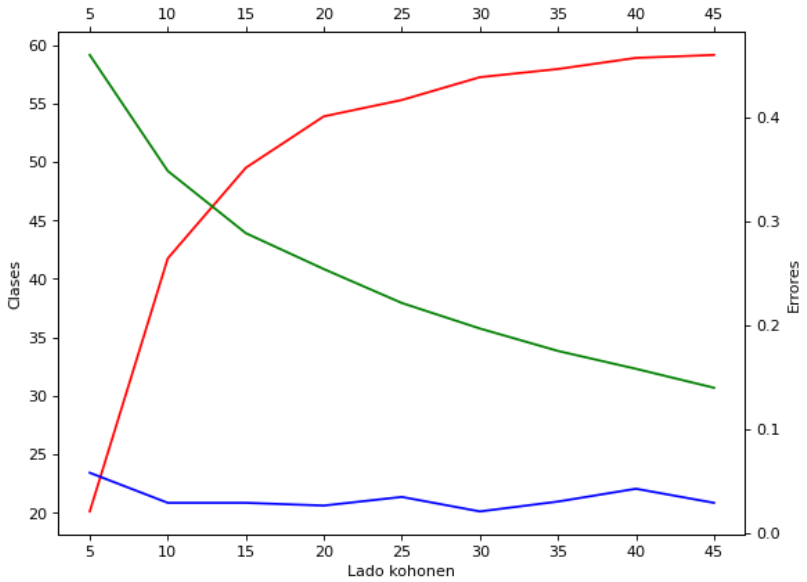
4.2.1.1 Lado mapa adecuado

Al igual que en el mapa de colores, queremos optimizar el lado de Kohonen para que, con menor lado, el mayor número de clases posible (lo más cercano a 70 clases, puesto que hay 70 tipos únicos de Pokémon). Los errores, además, deben mantenerse bajos.

Estudiaremos pues el lado de Kohonen con lados de mapa de 5 hasta 45, usando para todo un estándar de periodo 1000 y Eta de 0.2. Cada lado lo ejecutamos 20 veces y realizamos la media. De esta manera, obtenemos precisión altísima de los mejores valores. Una vez tenemos estos valores, realizamos el Dataset siguiente:

	0	1	2	3
0	5	20.8	0.456353…	0.029959…
1	10	42.8	0.346055…	0.051012…
2	15	48.6	0.291765…	0.030769…
3	20	53.8	0.252327…	0.030769…
4	25	54.2	0.220551…	0.058299…
5	30	56.6	0.197568…	0.017813…
6	35	59	0.172434…	0.027530…
7	40	59.4	0.158339…	0.044534…
8	45	59.2	0.139078…	0.037246…

Con los datos entonces, pintamos los resultados para ver cómo influye el lado de Kohonen en estas variables:



La línea roja representa como influye el lado de Kohonen en el Nº de clases. La línea verde en el error cuantificable y la línea azul en el error topológico.

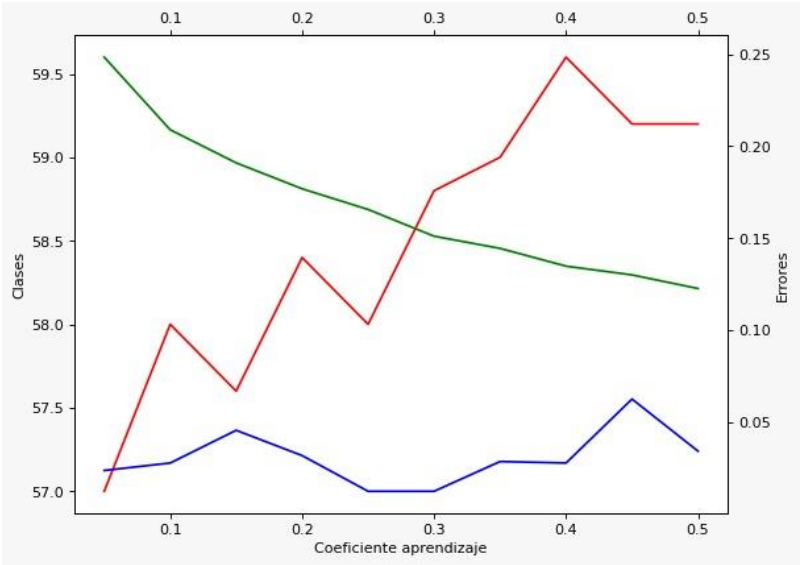
Podemos ver de manera curiosa como la gráfica tiene gran parecido con la gráfica de colores, teniendo la misma tendencia todas las variables. Y es que al igual que el anterior, el mejor lado de Kohonen es el de 30. Esto es ya que es el único que es capaz de maximizar el número de clases sin ver el error aumentado. Mencionar como 45 también podría ser una posibilidad, teniendo mejores estadísticas. Pero añadir 15 (45) al lado de Kohonen supone 1125 neuronas extra, lo que incrementaría de manera exponencial la ejecución por simplemente una mejora mínima del resultado. Queremos maximizar las clases y minimizar los errores con el menor lado posible y esto solo se da con 30.

4.2.1.2 Eta adecuada

En este caso, ya sabemos cuál es el lado de Kohonen más adecuado (30). Siguiendo la misma metodología, recogemos datos en este caso de Etas 20 veces, haciendo la media de cada uno, variando desde 0.05 hasta 0.5, creando así el presente Dataset:

	0	1	2	3
0	0.05	57	0.248463…	0.023481…
1	0.1	58	0.208933…	0.027530…
2	0.15	57.6	0.190918…	0.045344…
3	0.2	58.4	0.176809…	0.031578…
4	0.25	58	0.165532…	0.012145…
5	0.3	58.8	0.150991…	0.012145…
6	0.35	59	0.144354…	0.028340…
7	0.4	59.6	0.134691…	0.027530…
8	0.45	59.2	0.129921…	0.062348…
9	0.5	59.2	0.122513…	0.034008…

Sabiendo estos datos, procedemos entonces a encontrar cual sería el mejor coeficiente de aprendizaje para nuestro mapa autoorganizativo de Pokémon.



Al igual que en anterior gráfico, la línea roja representa como influye el Eta en el nº de clases. La línea verde en el error cuantificable y la línea azul en el error topológico.

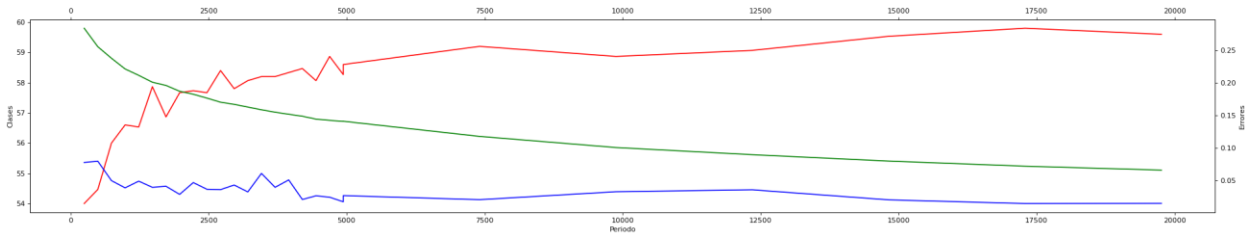
Podemos apreciar que el error topológico no es apreciablemente afectado por el coeficiente de aprendizaje es por ello que nos vamos a concentrar en los picos causados por el número de clases, tomando con bastante importancia el valor 0.4 donde podemos observar el máximo absoluto, es verdad que el error de cuantificación disminuye a mayor sea el coeficiente pero como podemos observar una tendencia asintótica vamos a conformarnos con el error de cuantificación que nos proporciona el 0.4 de coeficiente de aprendizaje ya que nos quedamos de este modo con el máximo absoluto del número de clases.

4.2.1.3 Periodo adecuado

En este punto, ya sabemos cuál es el lado de Kohonen más adecuado (30) y el mejor coeficiente de aprendizaje (0.4). Es por ello por lo que basándonos en estos valores procedemos a realizar pruebas incrementando el periodo desde 247-19760. En este caso la precisión es de 15 valores, puesto que 20 suponía un exceso de recursos, recogiendo los datos:

	0	1	2
0	247	0.283418…	0.078002…
1	494	0.255108…	0.079892…
2	741	0.237462…	0.050202…
3	988	0.221111…	0.039136…
4	1235	0.211303…	0.049392…
5	1482	0.200785…	0.039946…
6	1729	0.195876…	0.041565…
7	1976	0.186959…	0.029149…
8	2223	0.182394…	0.047233…
9	2470	0.176488…	0.036977…
10	2717	0.1701922	0.036437…
11	2964	0.166915…	0.043454…
12	3211	0.162775…	0.032928…
13	3458	0.158626…	0.061268…
14	3705	0.154905…	0.039946…
15	3952	0.151770…	0.051282…
16	4199	0.148736…	0.021322…
17	4446	0.144335…	0.026990…
18	4693	0.142415…	0.024831…
19	4940	0.140929…	0.017813…
20	4940	0.141063…	0.027260…
21	7410	0.117873…	0.021052…
22	9880	0.100833…	0.033198…
23	12350	0.089888…	0.036167…
24	14820	0.080100…	0.020782…
25	17290	0.072285…	0.015114…
26	19760	0.066279…	0.015384…

Como hemos hecho anteriormente, procedemos a pintar y a sacar el período más adecuado:



Por último, vemos como el periodo más adecuado está en 7500, donde podemos tener el punto más bajo del error topológico y el máximo número de clases. Podríamos intentar usar 15000 iteraciones, pero supondrían 75000 iteraciones extras para una variación mínima de los datos.

4.2.1.4 Conclusión

Como hemos podido ver en los anteriores apartados gracias a las gráficas, llegamos a la conclusión de que el correcto funcionamiento para este SOM es:

- Lado del mapa: 30
- Eta: 0.4
- Periodo: 7500

4.2.2 Cuestión 2

Para la mejor clasificación que hayas obtenido del Dataset de entrenamiento, adjunta el número de clases, mapa de clasificación, mapa de activaciones (histograma 3D), mapa de distancias del Dataset, el error de cuantificación y el error topológico. Explica detalladamente los resultados. Se valorará MUY positivamente que añadas gráficas o representaciones visuales intuitivas.

4.2.2.1 Número de clases y errores

En este apartado podemos ver como la red esta correctamente entrenada. En el set de entrenamiento se le adjuntan 70 tipos únicos de Pokémon (incluyendo combinaciones). Nuestra red reconoce 59 de media, con un error de cuantificación de 0.11 y un 0 de error topológico. Por un lado, el número de clases nos indica que reconoce la mayoría de los tipos y que hay tipos que agrupa por su parecido. El error de cuantificación nos indica que la distancia media es baja, teniendo los clústeres cerca. Por último, el error topológico nos dice que nuestro mapa es perfectamente homogéneo.

```
Numero de Clases:  59
Error De Cuantificacion:  0.11307170197714114
Error Topologico:  0.0
```

4.2.2.2 Mapas

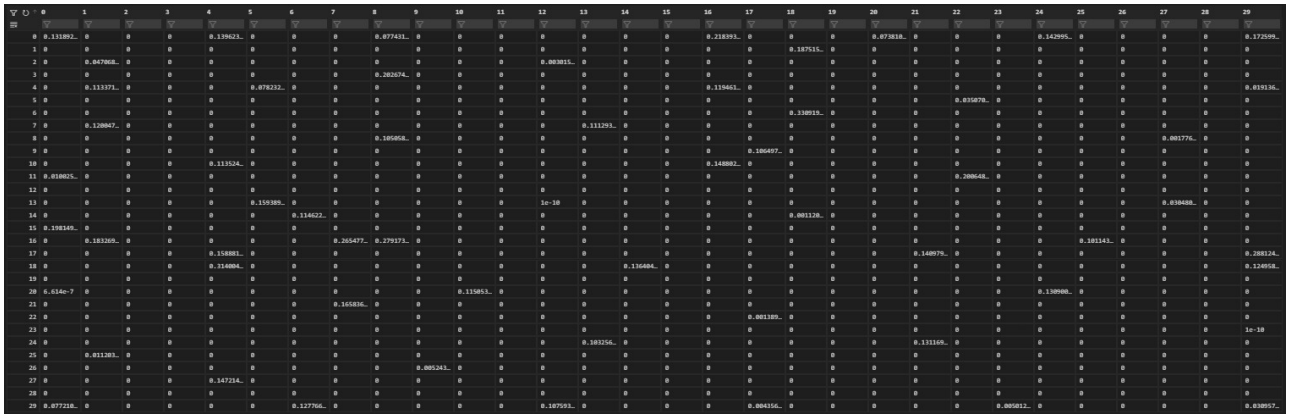
4.2.2.2.1 Mapa de clasificación

A continuación, vemos dos dimensiones del mapa de clasificación. Matizamos que solo vemos dos puesto que en el mapa de clasificación tenemos una matriz de 3 dimensiones $(30\ x\ 30\ x\ 18)$. En el mapa de clasificación de colores podíamos pintarlos puesto que la tercera dimensión era RGB, pero en este caso no tenemos una forma viable de representar esta última dimensión formada por 18 valores.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0.125	0.125	0	0.25	0.25	0.25	0.25	0.25	0	0.25	0.25	0.25	0.25	0.5	0.25	0.25	0.5	0.25
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0.0625	0.125	0	0.5	0.25	0.0625	0.25	0.25	0.25	0.125	0	0.5	0.25	0.5	0.25	0.5	0.5	0.25
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0.0625	0.25	0.25	0.5	0.125	0.0625	0.25	0.25	0.25	0.0625	0	0.5	0.25	0.125	0.5	0.5	0.25	0.25
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0.0625	0.25	0.25	0.5	0.125	0.125	0.125	0.25	0.25	0.0625	0	0.25	0.25	0.25	0.25	1	0.125	0.5
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0.125	0.25	0.25	0.5	0.25	0.0625	0.5	0.5	0.25	0.0625	0	0.5	0.25	0.25	0.25	1	0.25	0.25
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0.5	0.25	0.25	0.125	0.25	0.125	1	1	0.25	0.0625	0.0625	0.5	0.25	0.5	0.25	0.5	0.25	0.125
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	1	0.5	0.25	0.125	0.25	0.125	0.5	0.5	0.5	0.125	0.125	0.5	0.25	0.5	0.125	0.25	0.25	0.125

4.2.2.2.2 Mapa de distancias

En cuanto al mapa de distancias, observamos la distancia media de cada neurona respecto de los patrones activados por esta neurona. Vemos muchas neuronas que no han sido activadas, es decir, tienen distancia media de 0, y las neuronas que sí han sido activadas están repartidas de forma algo homogéneas. Las distancias son muy diversas, pero se sitúan entre los valores 0.3 y 0, por lo que entendemos que al ser valores bastante pequeños entendemos que la red ha conseguido crear clústeres de Pokémon lo suficientemente precisos y que ninguna neurona se ha quedado demasiado lejos de los Pokémon que se tenían que entrenar. Por lo tanto, entendemos que la red está bien entrenada y se ha adaptado bien al conjunto de datos de entrenamiento.

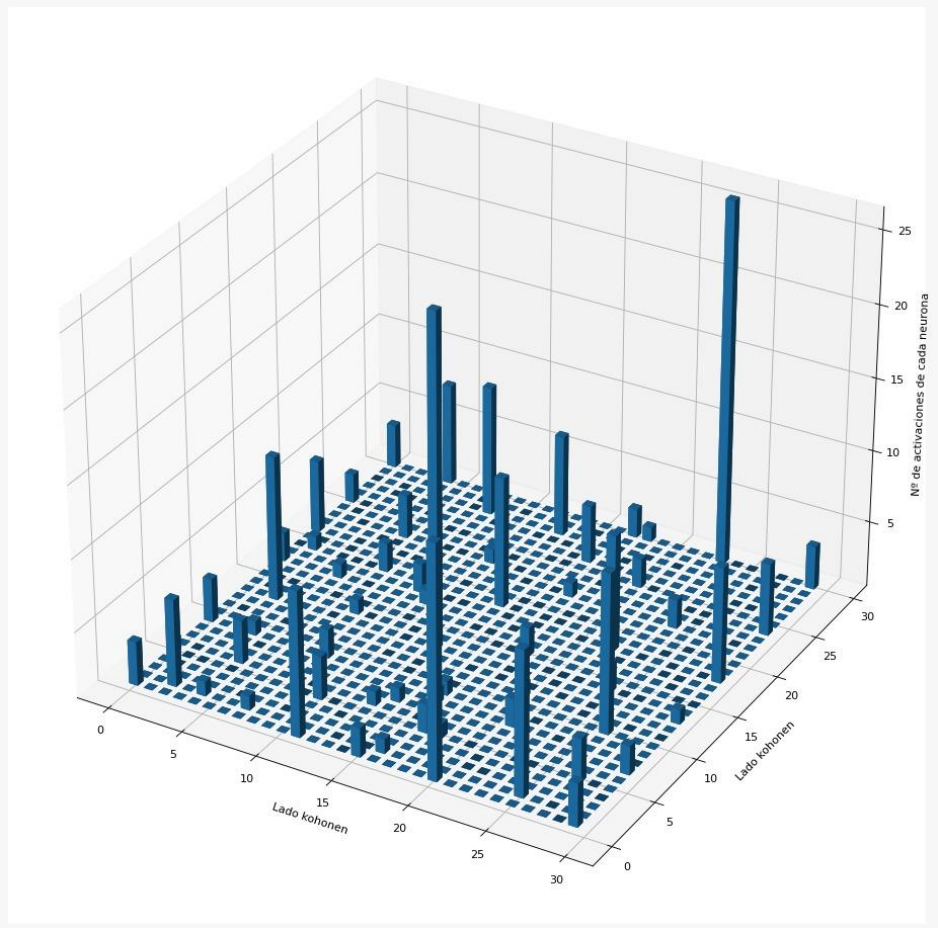


4.2.2.2.3 Mapa de activación

En el mapa de activación podemos ver el número de neuronas que se han activado y a su vez la cantidad de veces activadas. Esto es debido a que es un mapa en 3D, en el que tenemos lado x lado x N.º de activaciones. Es así como hemos visto la distribución de clases y en función de que se ha clasificado así.

Matizar como dos clases se repiten muchísimo más de la media. Eso nos indica que hay dos tipos de únicos de Pokémon que son bastante comunes en el Dataset de entrenamiento.

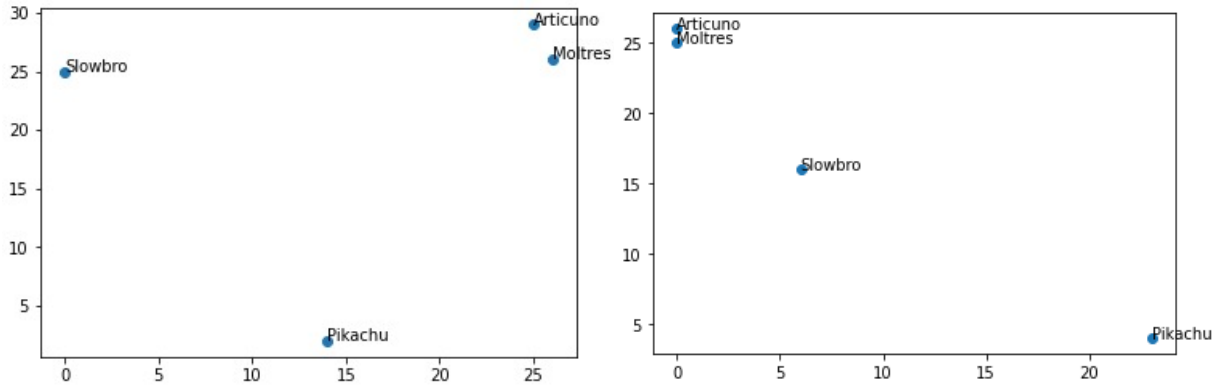
Para realizar este histograma nos hemos apoyado de los siguientes recursos: (Matplotlib , s.f.) (Programmerclick, s.f.)



4.2.3 Cuestión 3

Sin modificar el entrenamiento, clasifica el Dataset de pokemon_classify.csv (haz al menos dos ejecuciones) y responde a las siguientes preguntas: ¿En cuál de ellos se clasifica a Pikachu? ¿Por qué caen en el mismo grupo los Pokémon Articuno y Moltres siendo antagonistas en cuanto al tipo (hielo y fuego)? ¿Qué significa que el clúster donde se ha clasificado a Slowbro?

Para realizar esta cuestión, como bien se nos pide en el enunciado debemos de clasificar los cuatro Pokémon del CSV de clasificación. Es por ello por lo que con la red ya entrenada ejecutamos para ver la salida. A continuación, adjuntamos dos ejemplos de ejecución donde podemos ver las neuronas donde se clasifica cada Pokémon en un pequeño histograma:



4.2.3.1 Clasificación Pikachu

En ambas ejecuciones podemos ver como Pikachu está clasificado en sitios relativamente opuestos a los demás. Esto nos quiere decir que Pikachu tiene una combinación de tipos antagonista comparado con los otros Pokémon. Lo cual tiene sentido porque tanto el tipo agua de Slowbro como el tipo volador de Moltres y Articuno son débiles frente a ataques eléctricos.

4.2.3.2 Mismo grupo Articuno y Moltres

Es verdad que Moltres y Articuno tienen tipos antagonistas pero el motivo por el cual los vemos en el mismo grupo es porque comparten el tipo Volador, el cual aparece más veces en el Dataset de entrenamiento que tipo Fuego o Hielo. Esto hace que tengamos muchos más datos para crear clúster de tipo Vuelo, haciendo un clúster más grande donde se aúnen diferentes combinaciones. De esta manera el tipo Vuelo es más influyente a la hora de clasificar.

4.2.3.3 Clúster de Slowbro

El clúster donde se encuentra Slowbro es aquel donde los Pokémon de agua y psíquico se encuentren. Es adecuado decir también que aquellos Pokémon de agua simples o de psíquico simples se encuentren por los alrededores de este clúster, puesto que Slowbro es una combinación de ambos tipos. Además, por donde se encuentra, entendemos que los Pokémon de este clúster serán débiles frente a eléctrico, estando como vemos, muy alejado de Pikachu.

5 Conclusión

A modo de conclusión haremos un breve repaso de los puntos importantes que hemos tratado en esta práctica y los conocimientos que esta nos ha aportado.

En esta práctica hemos podido enfrentarnos a problemas relativamente reales de clasificación mediante mapas autoorganizativos. Hemos podido ver de una forma fácil e intuitiva como se clasifican los colores y los Pokémon, no por valores arbitrarios si no por reconocimiento de patrones de estos, en base a las entradas. El mapa de colores era una forma intuitiva de ver como mediante gradiente podíamos ver clúster de colores, y como su vecindario representaba visualmente el clúster. Por otro lado, la práctica de Pokémon nos ha enseñado como interpretar más allá de lo visual los datos, centrándonos y apoyándonos mucho en el sentido de los errores, clases y de donde nos ubica los datos de clasificación.

Por último, nos gustaría hacer referencia a algunos problemas que nos hemos enfrentado y como los hemos resuelto. El primero al que nos enfrentamos fue el correcto entendimiento de cuanto influía la implementación del vecindario en la autoorganización del mapa, siendo un cambio absoluto en la salida y en la calidad del mapa al implementar este concepto. Otro problema notorio que tuvimos es el poder entender y graficar cuales son los valores adecuados del mapa de Kohonen y como el cambio de los valores supone un gran cambio para la calidad del entrenamiento.

Concluir pues, lo mucho que hemos aprendido y disfrutando todos los integrantes con esta práctica, aprendiendo conceptos interesantes y pudiéndolos aplicar, siendo tremendamente satisfactorio además, el ver como correctamente funcionaban nuestros mapas.

6 Bibliografía

Matplotlib . (s.f.). Obtenido de https://matplotlib.org/stable/gallery/mplot3d/hist3d.html
Programmerclick. (s.f.). Obtenido de https://programmerclick.com/article/13171781661/
StackOverflow. (s.f.). Obtenido de https://stackoverflow.com/questions/9103166/multiple-axis-in-matplotlib-with-different-scales