Asignatura:

Inteligencia Artificial II

Título del documento:

Self-Organizing Maps

Práctica 1 - Laboratorios 1 y 2

Presentado por:

David Anastasio Enrique Alba

Luis Eduardo de Santiago Martínez

Cristina Espejo Fernández

Mario Martínez Vitutia

Carmen Moreno Udaondo

Francisco Javier Zaballa Gutiérrez

Tabla de contenidos

Tabla de figuras

Resumen

En este documento describimos la implementación y comprobación de Mapas Autoorganizativos (SOM) en dos laboratorios distintos. En el primero, enfocado en la clasificación de colores, abordamos la generación de 100 colores aleatorios en formato RGB, con los cuales entrenamos un SOM capaz de agrupar los colores similares en zonas contiguas de la red, demostrando la eficiencia y eficacia de esta técnica para la visualización y organización topológica de datos. El segundo laboratorio sigue esta metodología en un caso más complejo, la clasificación de Pokémons, donde empleamos 18 atributos (against∗)(against∗) relacionados con los daños que reciben los distintos tipos de Pokémon, permitiendonos agruparlos según patrones de resistencia y debilidad.

Los resultados de ambos laboratorios nos han enseñado que el SOM, configurado con los hiperparámetros adecuados (tamaño del mapa, learning rate, número de iteraciones y radio de vecindario), logra tanto una baja distancia media de cuantificación como preservar la topología. En el caso de los colores, obtenemos una distribución muy precisa de los colores en el mapa. Para los Pokémons, incluso con un mayor número de dimensiones (18 columnas), el SOM conseguimos identificar patrones de similitud basados en sus debilidades/resistencias, reflejando la adaptabilidad de la red a contextos con más atributos.

1 Introducción

Los mapas autoorganizativos o ***Self-Organizing Maps*** (SOM) son un tipo de red neuronal no supervisada que proyecta datos de alta dimensionalidad en un mapa bidimensional, preservando la relación de similitud entre los patrones [1], [2], [3]. Representan una herramienta muy útil en el ámbito de la Inteligencia Artificial y el Data Mining. Esto se debe a que permiten descubrir estructuras y agrupaciones de los datos sin un conocimiento previo de etiquetas y/o categorías.

Dentro de las redes neuronales, el algoritmo de Kohonen para los SOM destaca por su enfoque competitivo, donde las neuronas de la red compiten por representar mejor un patrón de entrada. La neurona ganadora, denominada **BMU (*Best Maching Unit*),** se ajusta de forma más intensa, y sus vecinas también se ven afectadas, creando así una topología de similitudes en el mapa [1], [4]. A lo largo del entrenamiento, tanto el radio del vecindario como el *learning rate* van decreciendo, lo que nos permite que al principio del entrenamiento se hagan ajustes grandes y globales y al final se hagan ajustes pequeños y locales.

En esta práctica, se busca que implementemos y comprobemos el funcionamiento de un SOM a través de dos laboratorios:

- **Laboratorio 1:** Un SOM de colores, donde generamos 100 colores aleatorios y observamos cómo se agrupan de forma coherente según sus componentes (RGB).

- **Laboratorio 2:** Un SOM de Pokemons, donde trabajamos con características de diferentes Pokemons (por ejemplo, sus valores de ataque, defensa, tipo, etc.) para detectar agrupamientos y similitudes.

1.1 Contexto general de los SOM

Los SOM se basan en la idea de **autoorganización**, es decir, los propios datos “enseñan” a la red la forma en que deben agruparse [1]. Cada neurona del mapa tiene un vector de pesos que, tras repetidas presentaciones de patrones, converge hacia regiones específicas del espacio de entrada, formándose grupos de patrones similares.

La implementación de estos mapas suele ser ligera en computación si se compara con otros métodos, ya que, a pesar de requerir varias iteraciones, cada actualización (BMU + vecindario) es un cálculo simple y local. Además, se pueden emplear técnicas de optimización para mejorar la convergencia [4].

En el contexto de la práctica, hemos creado unos notebooks en Python que, siguiendo las directrices del enunciado, permiten:

- Configurar los parámetros principales (tamaño del mapa, número de iteraciones, *learning rate* y vencindario).

- Llevar a cabo las funciones de búsqueda de la BMU, actualización de los pesos y cálculo de métricas.

- Visualizar los resultados mediante gráficos de activación, distancias y clasificación.

1.2 Objetivo de la práctica

El objetivo de esta práctica se centraba en diseñar una herramienta de entramiento SOM y validarla en dos casos de uso concretos:

1. **Clasificación de colores:** Laboratorio 1.
    - Generar un conjunto de datos aleatorios (valores RGB entre 0 y 255).
    - Entrenar un SOM para que sea capaz de agrupar esos colores según su similitud.
    - Probar el mapa con un conjunto de colores de referencia (rojo, verde, azul, etc.) y analizar la organización obtenida.
2. **Clasificación de Pokemons:** Laboratorio 2.
    - Utilizar el dataset proporcionado con diferentes atributos (tipo, ataque, defensa, etc.).
    - Aplicar el SOM para detectar potenciales agrupaciones naturales entre los diferentes Pokemons.
    - Comparar los resultados con nuestras expectativas.

De esta forma, se buscaba que entendiésemos los fundamentos de los SOMs y que pusiésemos en práctica todo el ciclo de desarrollo: desde la implementación de las funciones neuronales hasta la obtención de métricas e interpretación de los mapas.

1.3 Estructura del documento

A partir de esta introducción, hemos organizado la memoria de la siguiente forma:

1. **Sección 2 – Laboratorio 1:** Describimos los detalles de la configuración inicial de los parámetros, el proceso de entrenamiento y el resultado de la clasificación de un conjunto de colores generados aleatoriamente. Además, mostramos las visualizaciones (mapas de pesos, mapa de activaciones, etc.) para analizar el desempeño del SOM.
2. **Sección 3 – Laboratorio 2:** Explicamos como hemos aplicado los mismos métodos, pero con sus respectivas modificaciones a un dataset de Pokemons.
3. **Conclusiones:** Reflexiones finales sobre la práctica y los aprendizajes que hemos tenido.
4. **Bibliografía:** Fuentes que hemos utilizada sobre los diferentes conceptos y métodos.

2 Laboratorio 1 – Clasificación de colores

En este primer laboratorio, hemos implementado y validado un SOM con datos de colores generados aleatoriamente en formato RGB. El objetivo era observar cómo, tras el proceso de entrenamiento, el SOM es capaz de agrupar colores similares en regiones contiguas de la red.

2.1 Configuración inicial

Como punto de partida, hemos creado un vector de datos de colores aleatorias haciendo uso de `np.random.randint(0, valor_maximo + 1, (num_entradas, num_datos))`, donde:

- `Num_entradas`: Corresponde a los canales R, G y B (3).
- `Valor_maximo`: Es el valor máximo que puede tomar cada canal.
- `Num_datos`: Corresponde al número de colores (RGB) a generar.

Estos datos se almacenan en el vector `datos` que tiene la forma $(3, 100)$, de tal forma que cada columna representa un color. Posteriormente, para inicializar la red del SOM, lo hacemos de la siguiente forma:

1. **Creamos la matriz de pesos:** Definimos un lado para el mapa de forma que el SOM sea una red 2D de dimensiones $LadoMapa \times LadoMapa$ (`Lado_mapa` $= 100$).
2. **Inicialización aleatoria de los pesos:** Cada neurona de la red cuenta con un vector de pesos $[w_R, w_G, w_B]$, al que se le asignan inicialmente valores aleatorios entre 0 y 255.
3. **Definición de los hiperparámetros:** Establecemos un número de iteraciones totales (`periodo` $= 1000$), un *learning rate* inicial (`learning_rate` $= 0.05$) y un radio de vecindario inicial (`vecindario` $=$ `lado_mapa` $/ 2$).

De esta forma, la red queda preparada para ir recibiendo los patrones/vectores y autoorganizar las neuronas.

2.1.1 Parámetros óptimos

Para encontrar los parámetros óptimos del SOM (tamaño del mapa, número de iteraciones y learning rate), hemos implementado una función (`evaluar_parametros_som`). El método realiza los siguientes pasos:

1. **Recorre** sistemáticamente los rangos de parámetros que habíamos predefinidos:
    a. `Lado_mapa` $= [25, 50, 100]$.
    b. `Periodo` $= [1000, 2000, 3000, 4000, 5000]$.
    c. `Learning_rate` $= [0.01, 0.02, 0.03, 0.04, 0.05]$.

2. Para cada combinación (`lado_mapa`, `periodo`, `learning_rate`):

    a. **Inicializamos** el SOM con pesos aleatorios y un vecindario.

    b. **Entrenamos** el SOM con el vector de datos, llamando a la función `entrenar_datos`, que implementa el bucle de entrenamiento (cálculo de la BMU, actualización de pesos, etc.).

    c. Al finalizar el entramiento, calculamos dos métricas:

        i. **Error de cuantificación:** Calculamos la distancia euclídea media entre cada patrón y la neurona BMU que lo clasifica.

        ii. **Error topológico:** Mide el porcentaje de patrones para los cuales la segunda BMU no es adyacente a la primera en la red.

3. **Normalizamos** ambos errores para darles el mismo peso y, finalmente, seleccionamos la combinación de parámetros cuyo sumatorio de errores normalizados sea mínimo.

Escogimos un rango de learning rate más reducido (en torno a 0.05) porque, con valores más altos (por ejemplo, superiores a 0.10), se podían observar en el SOM final círculos o efectos de "anillos" en la distribución de los colores, lo que reflejaba un comportamiento inestable o incompleto.

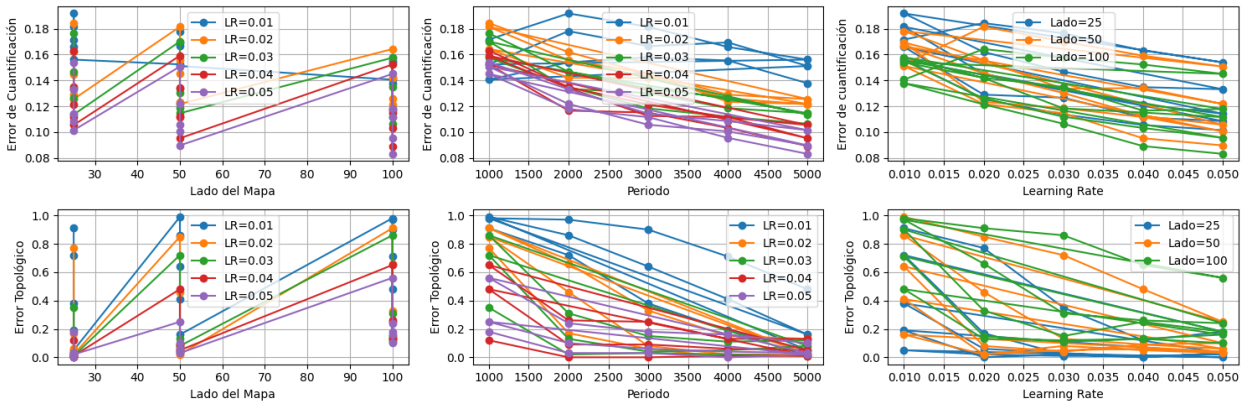

Ilustración 1. Resultado comparativa de parámetros (colores).

De esta forma evaluamos todas las combinaciones de parámetros posibles dentro de los rangos establecidos. El resultado que obtuvimos que proporcionaba la menor combinación de errores de cuantificación (0.0897) y topológico (0.0300) fue el siguiente:

- `Lado_mapa` $= 50$.
- `Periodo` $= 5000$.
- `Learning_rate` $= 0.05$.

El lado del mapa de 50 nos ofrece el espacio suficiente para distribuir ordenadamente los 100 colores, 5000 iteraciones permiten un ajuste detallado a lo largo del tiempo y un learning rate inicial de 0.05 no es tan grande como para generar inestabilidad ni tan pequeño como para ralentizar de forma notable la convergencia.

2.2 Entrenamiento de la red

El entrenamiento consiste en un bucle de 5000 iteraciones en el que se presenta, en cada paso, un color tomado aleatoriamente del conjunto de 100 colores. El SOM localiza su neurona ganadora (BMU) y actualiza tanto la BMU como las neuronas en su radio de vecindario, con un learning rate que decrece a lo largo de las iteraciones.

2.2.1 Entrenamiento del SOM

El proceso principal del entrenamiento en cada iteración es el siguiente:

1. **Visualización inicial:** Mediante el método de `pintar_mapa`, mostramos gráficamente la distribución inicial de los pesos en el espacio RGB. Esto nos permite tener una idea de la disposición inicial (aleatoria) de los que colores que "representa" cada neurona antes de iniciar el entrenamiento.

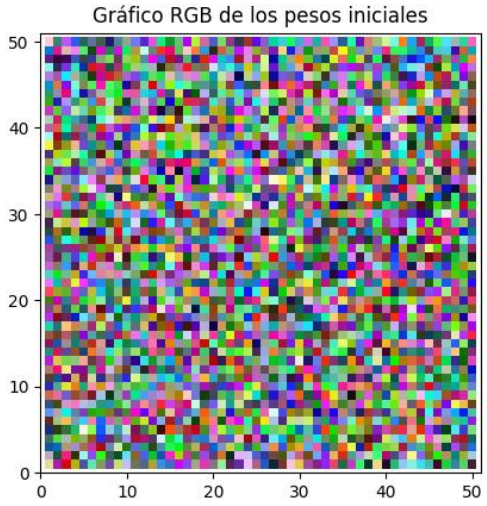

Ilustración 2. Gráfico RGB de los pesos iniciales.

2. **Bucle de entrenamiento (épocas):**

   a. De los 100 colores disponibles, escogemos uno al azar y lo pasamos a ser de forma $(1, 3)$ para ajustarse al formato de los pesos.

   b. Con `calcular_bmu`, obtenemos la neurona cuyo vector de pesos minimiza la distancia euclídea con el patrón.

   c. Actualizamos los parámetros:

      i. Reducimos gradualmente el `lr_actual` desde el valor inicial (`learning_rate`) hasta uno próximo a 0 a lo largo de las `periodo` iteraciones.

      ii. Disminuimos el radio de vecindario (`vec_actual`) hasta un mínimo que generalmente es 1.

d. Para cada neurona cuya distancia a la BMU sea inferior o igual al radio `vec_actual`, calculamos un factor de amortiguación para corregir el peso.

e. Cada 100 iteraciones, volvemos a pintar la matriz de pesos con el estado para observar cómo progresa el entrenamiento.

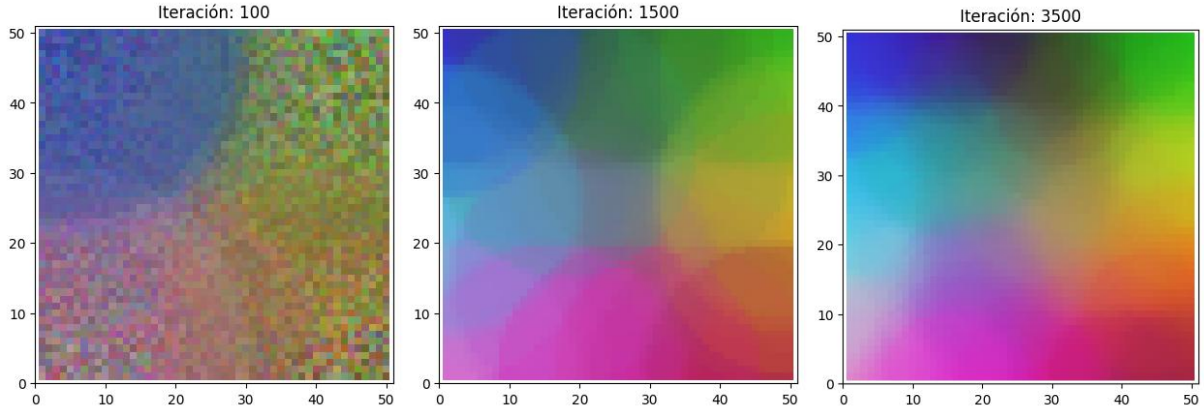

Ilustración 3. Gráfico RGB de las iteraciones 100, 1500 y 3500.

3. **Resultado final:** Tras completar todas las iteraciones, la matriz de pesos se devuelve como resultado de la función, quedando ajustada a la distribución de colores del dataset.

Gracias a este procedimiento, el SOM termina autoorganizando sus neuronas de modo que cada región de la red representa grupos de colores similares en el espacio RGB.

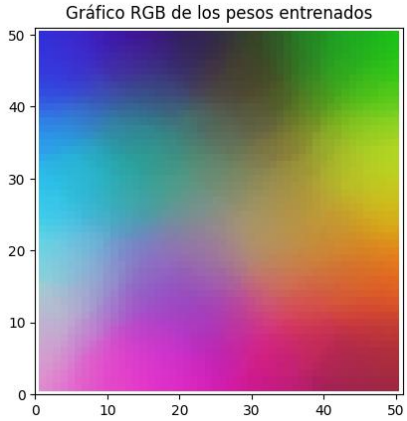

Ilustración 4. Gráfico RGB de los pesos entrenados.

2.2.2 Clasificación del SOM

Una vez que hemos entrenado, utilizamos el SOM para clasificar patrones (colores) con la función `clasificar_som`, la cual recibe los siguientes parámetros.

- `Datos`: El conjunto de datos a clasificar.
- `Matriz_pesos`: La matriz de pesos entrenada en el apartado anterior.
- `Calcular_BMU`: La función para calcular la BMU.

El proceso de clasificación consiste en las siguientes fases:

1. **Inicialización de mapas:**
   a. El mapa de clasificación almacenará el último color (vector RGB) clasificado por cada neurona. Tiene unas dimensiones de $(lado\_mapa \times lado\_mapa \times num\_entradas)$.

   b. El mapa de activaciones nos muestra cuántos patrones se han asignado a cada neurona $(lado\_mapa \times lado\_mapa)$..

   c. El mapa de distancias calcula la distancia media entre los patrones clasificados por cada neurona y sus pesos. Tiene unas dimensiones de $(lado\_mapa \times lado\_mapa)$.

2. **Asignación de cada patrón:**

   a. Para cada color en datos, llamamos a `calcular_bmu`, localizando la neurona ganadora.

   b. Incrementamos la activación de esa neurona y actualizamos la distancia promedio (con la fórmula que tiene en cuenta la distancia euclídea y el número total de patrones ya asignados a esa neurona).

   c. El mapa de clasificación se sobrescribe (haciendo la media del color actual con los almacenados) o lo conserva si es el primer patrón que asigna.

3. **Cálculo de errores:**

   a. Error de cuantificación: La media de las distancias patrón-BMU de todos los datos.

   b. Error topológico: Se calcula para cada patrón la segunda BMU y se verifica si es adyacente a la primera (en la red). Cuantos más patrones tenga la BMU y segunda BMU en neuronas contiguas, menor es el error topológico.

4. **Salida:**

   a. `Mapa_clasificación:` Los colores finales por neurona.

   b. `Mapa_activaciones:` El número de colores asignados.

   c. `Mapa_distancias:` Distancia promedio por neurona.

   d. `Num_clases:` Cuántas neuronas fueron activadas.

   e. `Error_cuantificación y error_topologico.`

A continuación, vamos a analizar los resultados que hemos obtenido en la clasificación utilizando los parámetros óptimos.

1. **Mapa de clasificación:** En la figura podemos ver un mapa de 50×50 neuronas donde cada casilla representa el último (o promedio de los) patrón(es) que esa neurona ha clasificado. Las casillas oscuras (negras) indican neuronas que no han activado ningún patrón o tienen un número de activación nulo.

    a. **Distribución de colores:** Podemos observar cómo diferentes áreas de la red se especializan en una gama concreta, apareciendo tonos azules en la zona superior-izquierda, verdes en el borde superior-derecho, y magentas/rosas o rojos hacia la parte inferior del mapa, entre otras transiciones.

    b. **Grado de granularidad:** A pesar de que la mayoría de las neuronas están activadas por uno o pocos patrones, se ve una transición ordenada de colores, indicando que el SOM ha aprendido la topología del espacio RGB.

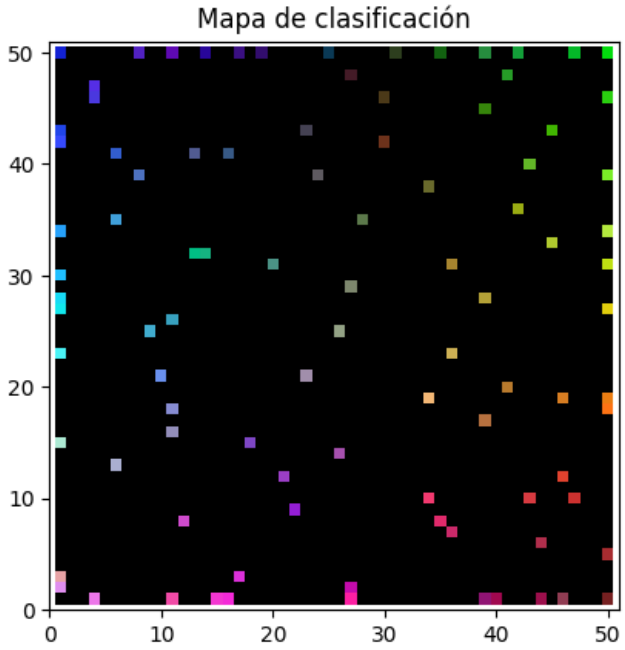

Ilustración 5. Mapa de clasificación.

2. **Mapa de activaciones:** En el histograma 3D mostramos la cantidad de patrones que cada neurona ha clasificado.

    a. **Altura de las barras:** La mayoría de las barras se sitúan en torno a 1 o 2, lo que significa que la gran mayoría de neuronas activas reconocen solo uno o dos colores de los datos. Esto nos sugiere una muy buena especialización de cada neurona en un rango específico de color.

    b. **Amplia cobertura:** Como hay un total de 100 patrones y se han activado 94 neuronas diferentes, casi cada color tiene su "neurona exclusiva" o la comporte con uno/dos patrones muy similares. Esto nos demuestra que es un mapa extenso con espacio suficiente para distinguir tonalidades cercanas.

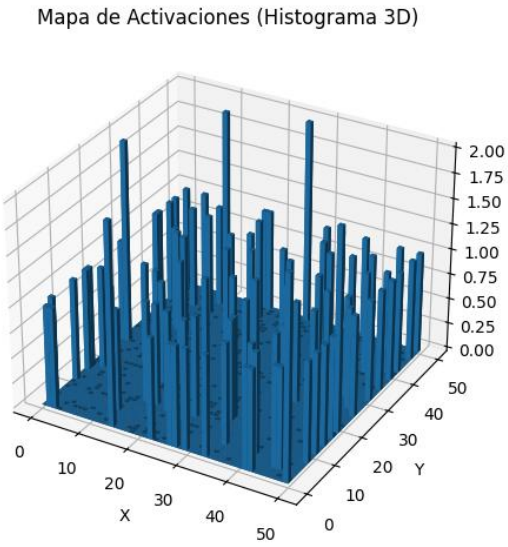

Ilustración 6. Mapa de activaciones.

3. **Mapa de distancias del dataset:** Este gráfico muestra la distancia media (en el espacio RGB) de los patrones asignados a cada neurona respecto a los pesos de esta.

    a. **Valores mayormente bajos:** La escala (amarillos y blancos) indica que la mayoría de las neuronas presentan distancias menores a 0.02, lo que nos confirma un ajusta muy preciso entre cada patrón y su neurona ganadora (BMU).

    b. **Algunas neuronas con distancias algo mayores:** Sugiere que los colores asignados a esas neuronas tienen una variación más notable respecto al peos de la neurona, o bien agrupan patrones menos frecuentes.

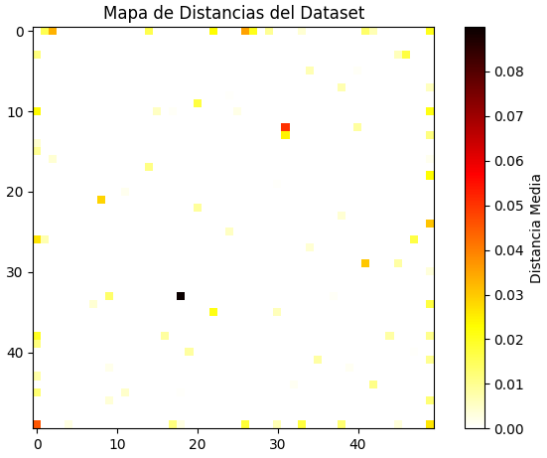

Ilustración 7. Mapa de distancias del dataset.

4. **Errores:**

    a. **Error de cuantificación (0.011755):** Este valor, muy bajo, nos sugiere que la distancia media entre los patrones de color y su BMU es pequeñísima, lo que nos demuestra que el SOM los representa fielmente.

b. **Error topológico (0.020000):** Sólo un 2% de los patrones tienen una segunda BMU que no es adyacente a la primera en la red, lo cual significa que el mapa preserva la estructura de similitud entre colores de muy buena forma.

2.3 Resultados y mapas de las pruebas

Para validar el SOM entrenado con los mejores parámetros, hemos clasificado los siguientes colores de prueba:

```
colores_prueba = np.array([
    [255, 255, 255],  # Blanco
    [255, 0, 0],      # Rojo
    [0, 255, 0],      # Verde
    [0, 0, 255],      # Azul
    [255, 255, 0],    # Amarillo
    [255, 0, 255],    # Magenta
    [0, 255, 255],    # Cian
    [0, 0, 0]         # Negro
])
```

Ilustración 8. Colores de prueba.

El proceso de prueba lo hemos implementado de tal forma que recorre cada patrón (color), determina su neurona ganadora (BMU) y añade la información en tres mapas. Además, calculamos el error de cuantificación y el error topológico para estos 8 colores. Los resultados obtenidos son los siguientes:

1. **Mapa de clasificación:** En este mapa, únicamente se pitan las neuronas que han clasificado alguno de los 8 colores de prueba. Cada cuadrado coloreado representa el último color clasificado en esa neurona.

    a. **Ubicación dispersa:** Podemos observar cómo cada uno de los 8 colores básicos se ha asignado a una neurona distinta, quedando en diferentes esquinas o bordes del mapa.

    b. **Número de clases:** Como cada color se corresponde con su propia neurona activada, el mapa refleja 8 clases (una por cada color de entrada). Es decir, cada color de prueba ocupa una neurona diferente en la red.

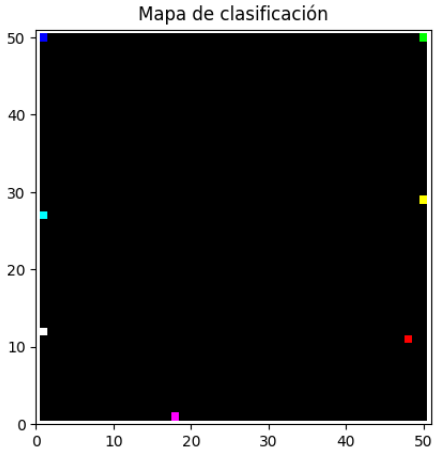

Ilustración 9. Mapa de clasificación de la prueba.

2. **Mapa de activaciones:** Este gráfico nos muestra cuantos patrones (de los 8 de prueba) ha clasificado cada neurona.

    a. **Columnas altas en 8 posiciones:** Podemos comprobar que hay ocho “picos” con altura 1, indicando que cada una de esas neuronas ha reconocido un único color.

    b. **Cero en las demás:** El resto de las neuronas no se activan, por lo que no clasificaron ninguno de los 8 patrones de prueba, tal como se vemos en la figura.

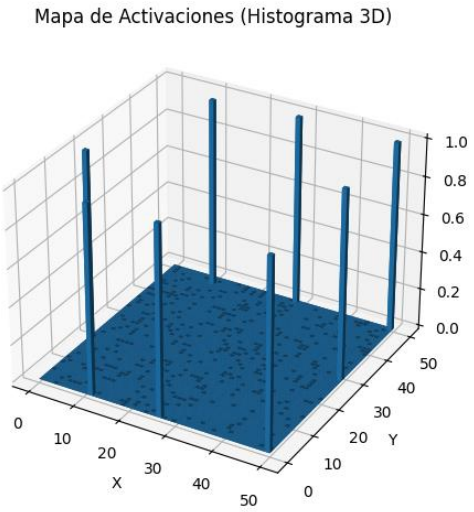

Ilustración 10. Mapa de activaciones de la prueba.

3. **Mapa de distancias del dataset:** Este mapa representa la distancia media entre el color clasificado y los pesos de la neurona BMU para cada posición.

    a. **Valor alto:** El valor más alto en el mapa (negro/rojo) nos muestra distancias cercanas o superiores a 0.1-0.15.

    b. **Algunos puntos rojos/naranjas:** Estos puntos corresponden a neuronas que asignaron colores a una distancia mayor, lo que sugiere que esos colores de prueba no tienen una neurona entrenada exactamente en ese mismo punto del espacio RGB.

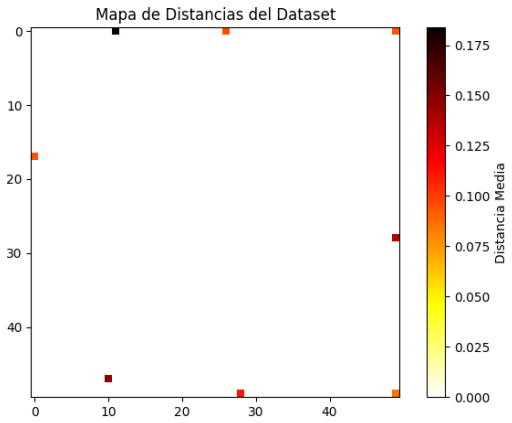

Ilustración 11. Mapa de distancias del dataset de la prueba.

4. **Errores:**

    a. **Error de cuantificación (0.118587):** Este valor nos indica la distancia euclídea media (al cuadrado) entre los 8 colores y la neurona que los ha clasificado. Comparado con los valores típicos de 0 a 255 en RGB, un error aproximado de 0.12 nos sugiere que el SOM no siempre tiene un peso que coincida exactamente con los colores de prueba, aunque sigue siendo razonablemente cercano.

    b. **Error topológico (0.125000):** Aproximadamente el 12.5% de los colores tiene su segunda BMU en una neurona que no es adyacente a la BMU principal. Esto implica que, para la mayoría de los colores de prueba, el mapa conserva la topología. Sin embargo, uno de los colores tiene una segunda BMU más lejana, indicando una ligera pérdida de continuidad topológica.

En conjunto, estos resultados nos confirman la capacidad de nuestro SOM para clasificar patrones que formaron parte de su entrenamiento, con un error de cuantificación y un error topológico razonables.

3 Laboratorio 2 – Clasificación de Pokémon

En este segundo laboratorio, hemos utilizado como base la red SOM que hemos desarrollado en el Laboratorio 1, añadiéndole ciertas modificaciones para adaptarla al número de atributos en el dataset de Pokemons. El objetivo era generalizar la red anterior y aplicarla en un contexto con más dimensiones, abordando patrones con un mayor número de características.

3.1 Configuración inicial

Para empezar, cargamos la información del archivo *pokemon_train.csv* en un Datarame de pandas, de la siguiente forma: `datos_original = pd.read_csv(pokemon_train.csv)`.

El dataset contiene, para cada Pokemon, diferentes columnas que indican el raño recibido de los diferentes tipos existentes (en total, 18 columnas). Si lo hablamos en cuanto al contexto de la red SOM:

- `Num_entradas`: Corresponde a las columnas que indican cuanto daño reciben los Pokemons (18 en total).
- `Valor_maximo`: Es el valor máximo que puede tomar cada atributo/canal, con el fin de normalizar los datos.
- `Num_datos`: Corresponde al número total de filas/Pokemons cargados desde el CSV:

En cuanto a la inicialización de la matriz de pesos y de los hiperparámetros, seguimos el mismo procedimiento antes descrito en el Laboratorio 1, asignando pesos iniciales de forma aleatoria y definiendo los parámetros de aprendizaje (learning rate, periodo, etc.). De esta forma, conseguimos que la red quede lista para recibir los datos de Pokemons y actualizar sus neuronas en función de este nuevo espacio de características.

3.1.1 Parámetros óptimos

Al igual que en el laboratorio anterior, hemos decidido implementar una búsqueda automatiza sobre los distintos valores posibles para los principales parámetros. Los rangos de valores que hemos escogido en este caso son:

a. `Lado_mapa` $= [25, 50, 100]$.
b. `Periodo` $= [1000, 2000, 3000, 4000, 5000]$.
c. `Learning_rate` $= [0.01, 0.05, 0.10, 0.50]$.

Siguiendo el mismo proceso que en el laboratorio 1, entrenamos la red con cada combinación, buscando aquellos errores más bajos, que impliquen que nuestra red tiene la calidad necesaria y suficiente. En cada combinación, entrenamos la red y calculamos:

- **Error de cuantificación:** Media de las distancias euclídeas entre cada Pokémon y su neurona BMU.
- **Error topológico:** Porcentaje de patrones para los que la segunda BMU no es adyacente a la primera, evaluando si la red preserva la topología.

La diferencia principal que hemos encontrado respecto al laboratorio anterior es el número de dimensiones (18 en vez de 3), lo que incrementa el tiempo de entramiento en la búsqueda de los mejores parámetros.

Una vez se ha entrenado la red con todas las combinaciones del rango, graficamos de modo que se pueda observar cómo ha mejorado/empeorado el modelo a medida que los parámetros cambiaban de valores.

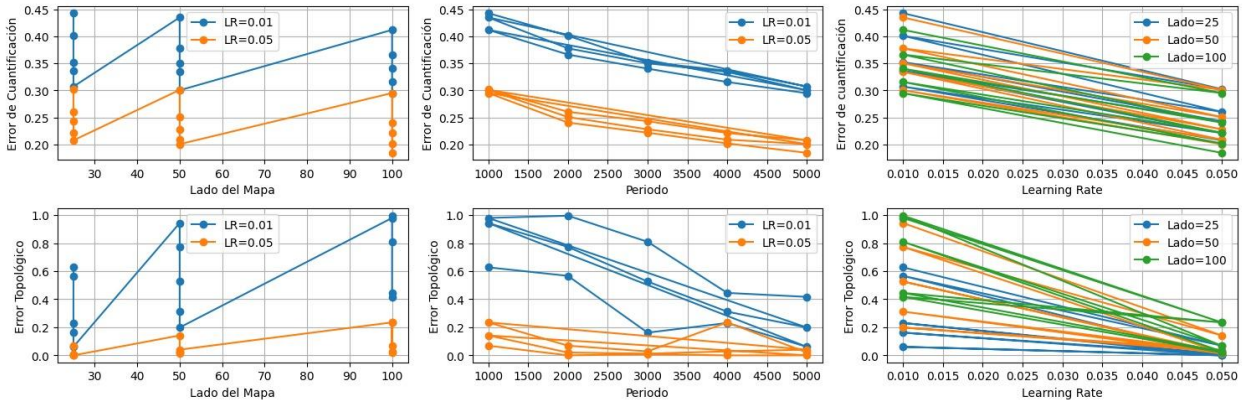

Ilustración 12. Resultado comparativa de parámetros (Pokemons, 1er rango).

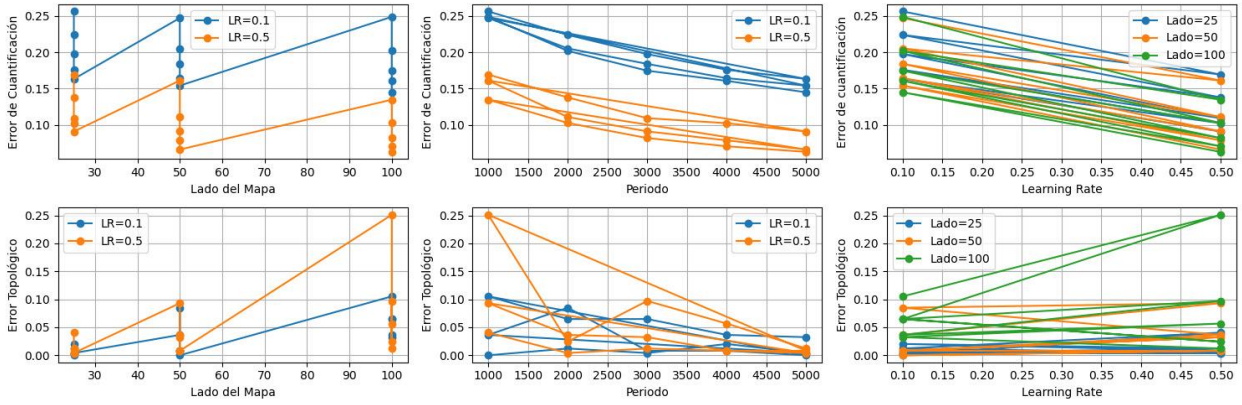

Ilustración 13. Resultado comparativa de parámetros (Pokemons, 2ndo rango).

Además, obtenemos el mínimo error, lo cual nos proporciona el set de valores que consiguen entrenar la red de forma más eficiente, en este caso con estos errores y estos valores:

- `Lado_mapa` $= 100$.
- `Periodo` $= 5000$.
- `Learning_rate` $= 0.5$.
- `Error topológico` $= 0.0628$.
- `Error cuantificación` $= 0.0121$.

3.2 Entrenamiento de la red

En este segundo laboratorio, el SOM trabaja únicamente con los atributos numéricos relacionados con la efectividad de tipos (daños recibidos) de cada Pokémon, sin tener en cuenta directamente su tipo. De esta forma, la red se organiza en función de los 18 atributos

(`against_*`) y genera grupos de Pokemons con características afines, basadas en las resistencias y debilidades entre tipos.

Hemos configurado el entrenamiento para que se realice a lo largo de 5000 iteraciones. En cada iteración, se le presenta un patrón de entrada (los 18 valores `against_*` de un Pokemon), se localiza su neurona ganadora (BMU) y se actualizan los pesos de la BMU y sus neuronas vecinas, ajustando tanto la tasa de aprendizaje como el radio del vecindario progresivamente hasta terminar el período indicado.

3.2.1 Entrenamiento del SOM

El entrenamiento, similar al del laboratorio 1 pero adaptado al mayor número de dimensiones, sigue los siguientes pasos:

1. **Selección aleatoria del Pokemon:**
    a. De los 274 Pokemons presentes en el dataset, elegimos uno al azar: `indice_random = np.random.randint(0, len(datos))`
    b. El vector de entrada contiene sus 18 valores `against_*.`
2. **Cálculo de la BMU:**
    a. A través de la función `calcular_bmu,` obtenemos la neurona ganadora midiendo la distancia euclídea entre el patrón y cada neurona del SOM.
    b. La BMU es la neurona cuyos pesos están más próximos al patrón de entrada.
3. **Actualización de los parámetros:**
    a. Reducimos gradualmente el `lr_actual` desde el valor inicial (`learning_rate`) hasta uno próximo a 0 a lo largo de las `periodo` iteraciones.
    b. Disminuimos progresivamente hasta un valor mínimo (habitualmente 1) el radio del vecindario (`vec_actual`), acotando la influencia de la BMU sobre neuronas lejas a medida que avanza el entrenamiento.
4. **Función de amortiguación (decay):** El factor de amortiguación atenúa la actualización de los pesos según la distancia 2D de cada neurona a la BMU. Cuanto mayor sea la distancia a la BMU, menor será el decay aplicado.
5. **Monitorización:** Dado que cada neurona tiene 18 dimensiones de pesos, resulta difícil visualizar el mapa de forma directa. Por ello, cada cierto número de iteraciones (p. ej., cada 100), se registran y muestran los errores (cuantificación y topológico), para confirmar si el modelo está mejorando o si se ha estabilizado.

```
1. EQ: 0.06405533272425384, ET: 0.0931174089068826
2. EQ: 0.06311432936949325, ET: 0.0931174089068826
3. EQ: 0.0628675211840323, ET: 0.08906882591093117.
```

Ilustración 14. Errores de cuantificación y topológico entrenamiento.

Este proceso permite que el SOM autoorganice los Pokemons en base a la similitud/parecido de sus atributos de efectividad. Al finalizar, las neuronas de la red se especializan en diferentes zonas que representan grupos de Pokémon con patrones `against_*` similares.

3.2.2 Clasificación del SOM

Una vez completado el entrenamiento, la clasificación de los Pokemons se realiza mediante la función `clasificar_som`, que recibe los siguientes parámetros:

- `Datos`: El datase ton los valores y el tipo de cada Pokémon.
- `Matriz_pesos`: La matriz de pesos entrenada.
- `Calcular_BMU`: La función para calcular la BMU.

Los pasos específicos de la clasificación son:

1. **Inicialización de mapas:**
   a. Para el mapa de clasificación creamos una matriz $(lad\_mapa \times lad\_mapa)$ con cadenas vacías (‘ ‘), ya que en este laboratorio guardaremos el tipo del Pokemon (combinando `tipo1` y `tipo2`) en lugar de valores RGB.

   b. El mapa de activaciones es un array $(filas, columnas)$ con ceros par contar cuántos Pokemons son asignados a cada neurona.

   c. El mapa de distancias es una matriz $(filas, columnas)$, que también inicializamos a 0, donde se acumula la distancia media entre los patrones y los pesos de cada neurona.

2. **Asignación de cada Pokemon:**

   a. Para Pokemon en el dataset, llamamos a `calcular_bmu`, para encontrar la neurona ganadora, y concatenamos su tipo (`tipo1` + `tipo2`) como texto dentro del mapa de clasificación (si la neurona se activa por primera vez, se asigna directamente; de lo contrario, se sobrescribe/concatena).

   b. El mapa de activaciones se incrementa en la posición de la BMU, reflejando el número de Pokemons asignados a esa neurona.

   c. El mapa de distancias se actualiza de forma incremental, calculando la distancia euclídea entre el Pokemon y los pesos de la BMU y ajustando la media según el número total de patrones en esa neurona.

3. **Cálculo de errores:**

   a. Error de cuantificación: La media de las distancias patrón-BMU en todo el dataset.

   b. Error topológico: Par cada Pokemon, se buscamos una segunda BMU y verificamos si es adyacente a la primera. Si la segunda BMU no se encuentra entre las vecinas inmediatas, se contabiliza el error topológico.

4. **Salida:**

a. `Mapa_clasificación:` Muestra los tipos asignados a cada neurona, lo que nos permite identificar regiones de Pokémon con atributos `against_*` parecidos y, en consecuencia, tipos con patrones de resistencia semejantes.

b. `Mapa_activaciones:` Nos indica cuántos Pokémon han caído en cada neurona.

c. `Mapa_distancias:` Expone la distancia media en cada neurona y es útil para determinar zonas con buen ajuste y áreas con mayor dispersión.

d. `Num_clases:` Cuántas neuronas resultan activadas.

e. `Error_cuantificación y error_topologico.`

En definitiva, esta fase de clasificación confirma cómo la organización que se ha creado durante el entrenamiento agrupa Pokemons con similares debilidades y resistencias, sin necesitar la etiqueta de tipo como dato de entrada, sino deduciendo la afinidad a través de las 18 dimensiones `against_*.`

A continuación, vamos a analizar los resultados que hemos obtenido en la clasificación utilizando los parámetros óptimos.

1. **Mapa de clasificación:** En la figura podemos ver un mapa de 100 x 100 neuronas donde cada casilla representa el último (o promedio de los) patrón(es) en forma de círculo de color que esa neurona ha clasificado. Ese color representa el tipo(s) del patrón de entrada que activó a esa neurona en cuestión. Para transformar los datos de tipo string (`tipo_completo`) a colores, se ha usado un diccionario (`tipo_color_dict`) y una función (`mezclar_colores_segun_tipo`) que mezcla los colores de aquellos Pokémon que cuentan con 2 tipos.

    a. Del mapa de clasificación podemos ver cómo se forman diferentes grupos en base a los colores asignados a los tipos de cada Pokémon. Vemos, por ejemplo, como los Pokémon de tipo “water/…” tienen tonos azulados asignados, y se encuentran todos en la misma zona, en la centroderecha. Vemos como el verde en la parte superior derecha (“grass/…”), el rojizo a la centroizquierda (“fire/…”), y el morado en la parte superior derecha (“dragón/…). Existen muchos más grupos, pero son menos numerosos, por lo que no generan un cluster tan visible como los mencionados

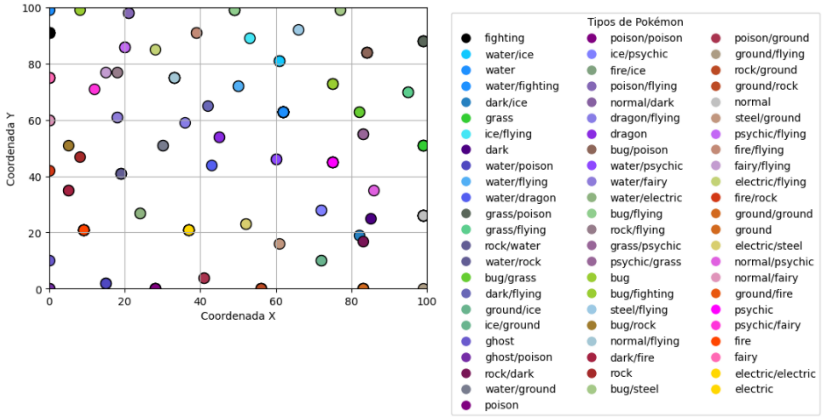

Ilustración 15. Mapa de clasificación.

2. **Mapa de activaciones:** En este histograma 3D, el eje X y el eje Y representan las coordenadas de la red 2D del SOM, mientras que la altura de cada barra indica cuántos Pokemons se han asignado a la neurona correspondiente.

    a. **Distribución general:** Podemos ver que la gran mayoría de las neuronas tienen alturas muy pequeñas (entre 0 y 2), lo que significa que cada neurona representa, por lo general, poco Pokemons.

    b. **Picos notables:** Algunas columnas superan las 5, 10 o incluso 20 activaciones. Esto nos indica la existencia de clusters de Pokemons con patrones (`against_*`) muy parecidos que se agrupan en la misma región de la red.

    c. **Amplia cobertura:** Como el mapa es extenso los Pokemons con atributos diferentes no se solapan en la misma neurona y, aún así, existen neuronas que concentran grupos más numerosos de Pokemons.

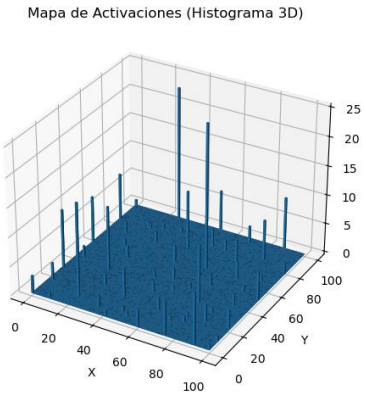

Ilustración 16. Mapa de activaciones.

3. **Mapa de distancias del dataset:** Este gráfico muestra la distancia media (en el espacio RGB) de los patrones asignados a cada neurona respecto a los pesos de esta.

    a. Del mapa de distancias podemos concluir que el SOM ha hecho un gran trabajo, ya que la distancia media máxima entre los valores de un patrón de entrada y su BMU ronda el 0.2. Es, por tanto, un modelo bastante ajustado al dataset con una calidad excepcional y poco errático

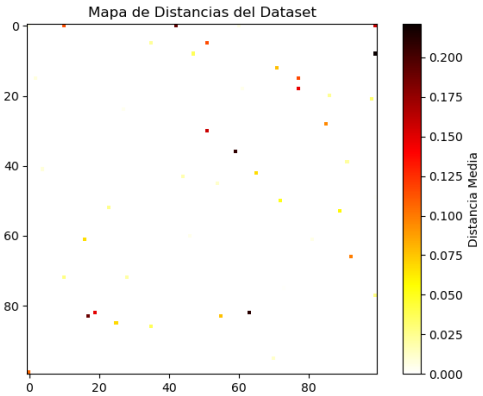

Ilustración 17. Mapa de distancias del dataset.

4. **Errores:**

    a. **Error de cuantificación (0.016438):** Este valor nos indica la distancia media patrón-BMU para todo nuestro dataset. El error es muy bajo y, evidencia que, de media, cada Pokémon se encuentra bastante cerca de su neurona ganadora.

    b. **Error topológico (0.032389):** Alrededor de un 3.2% de los Pokemons tienen su segunda BMU en una posición no adyacente a la BMU principal, reflejando que la mayor parte de patrones conserva la cercanía esperada en la red.

3.3 Resultados y mapas de las pruebas

Para comprobar que hemos entrenado y clasificado correctamente con nuestro SOM, hemos introducido los siguientes datos de prueba (`pokemon_classify.csv`):

```
1.id,against_bug,against_dark,against_dragon,against_electric,against_fairy,against_fight,against_fi
re,against_flying,against_ghost,against_grass,against_ground,against_ice,against_normal,against_pois
on,against_psychic,against_rock,against_steel,against_water,name,type1,type2
2. 24,1.0,1.0,1.0,0.5,1.0,1.0,1.0,0.5,1.0,1.0,2.0,1.0,1.0,1.0,1.0,1.0,0.5,1.0,Pikachu,electric,-
3. 79,2.0,2.0,1.0,2.0,1.0,0.5,0.5,1.0,2.0,2.0,1.0,0.5,1.0,1.0,0.5,1.0,0.5,0.5,Slowbro,water,psychic
4. 143,0.5,1.0,1.0,2.0,1.0,1.0,2.0,1.0,1.0,0.5,0.0,1.0,1.0,1.0,1.0,4.0,2.0,1.0,Articuno,ice,flying
5. 145,0.25,1.0,1.0,2.0,0.5,0.5,0.5,1.0,1.0,0.25,0.0,1.0,1.0,1.0,1.0,4.0,0.5,2.0,Moltres,fire,flying
```

Ilustración 18. Pokemons de prueba.

El proceso de prueba es similar al Laboratorio 1, pero aplicado a los datos correspondientes a los Pokemons de prueba:

1. Para cada Pokemon de `pokemon_classify.csv`, extraemos sus 18 valores (`against_*`).

2. Calculamos la BMU, es decir, aquella cuyo vector de pesos está más próximo a sus valores de daño recibido.

3. Añadimos la información de cada Pokemon a tres mapas de clasificación, activaciones y distancias, al mismo tiempo que calculamos las métricas de error de cuantificación y error topológico para todo este conjunto de prueba.

Los resultados de estas pruebas han sido:

1. **Mapa de clasificación:** El mapa de clasificación agrupa a los 4 Pokémons del dataset de classify en los grupos correspondientes.

    a. Para comprenderlo mejor, veamos donde se agruparon los pokemon de tipo ‘electric’ en la clasificación del dataset original, de 274 Pokémon:

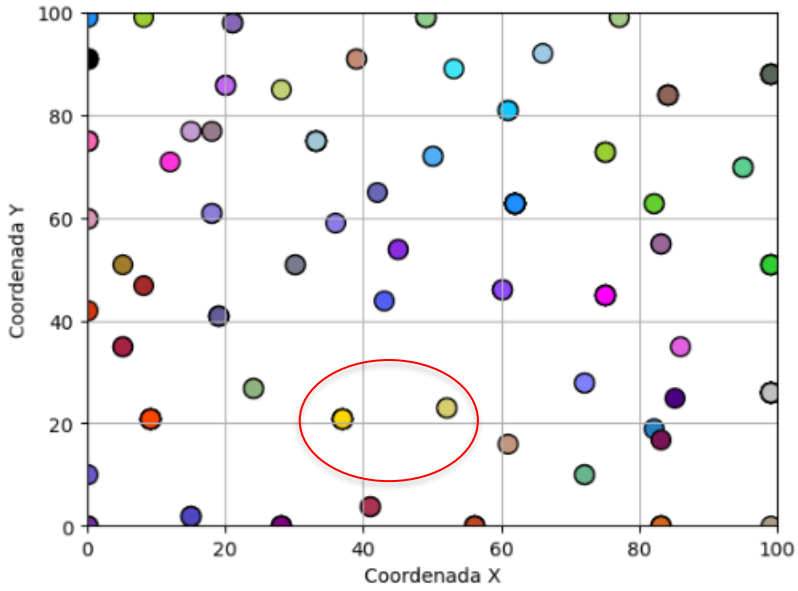

Ilustración 19. Comparación Pokemons electricos (Mapa Clasificación).

Como se puede ver, los eléctricos *("electric" y "electric/electric")* están agrupados en torno a las coordenadas (45, 20) en el mapa autoorganizativo.

b. Por otro lado, si vemos el mapa del dataset de '*pokemon_classify.csv*', vemos como ha clasificado al Pokemon Pikachu en la neurona (30, 20) aproximadamente. Por lo tanto, podemos afirmar que el Pikachu ha sido clasificado correctamente en el grupo que le corresponde, más cerca del tipo *"electric"* que del *"electric/electric"*.

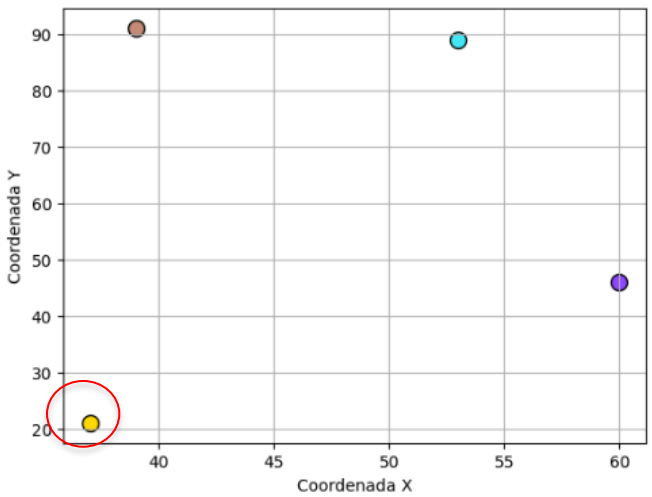

Ilustración 20. Clasificación de Pikachu (Mapa de Clasificación de Prueba)

c. La cuestión 3 también pregunta sobre el porqué clasifica al Articuno y al Moltres en el mismo grupo. Como podemos observar en la siguiente ilustración, ambos pokemons están en una zona común. Veamos primero la clasificación principal, para ver donde agrupó a aquellos pokemon de tipo 'flying':

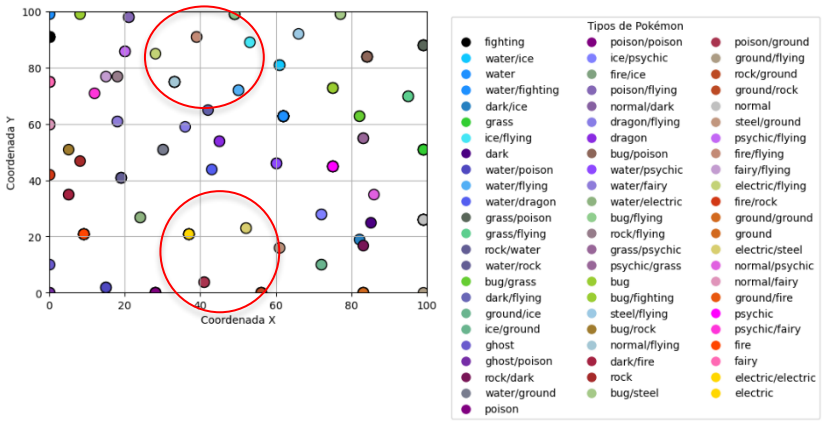

Ilustración 21. Clasificación Articuno y Moltres tipos (Mapa de Clasificación)

Echando un vistazo a la leyenda, podemos ver que en esa esquina inferior izquierda están los siguientes tipos agrupados: electric/flying, fire/ice, posion/flying, normal/flying, fire/flying e ice/flying. Todos ellos están relacionados con el tipo flying, por lo que es un grupo lógico.

d. Viendo que el SOM clasificó a estos dos Pokemon voladores en la misma zona, y, aunque el tipo fire e ice sea opuesto, la red no distingue entre tipos, ya que solamente fue entrenada en base a los valores de daños recibido por otros Pokémon. Por lo tanto, viendo los valores proporcionados en el dataset, podemos observar cómo hay una similitud bastante grande. Esta es la única razón por la que el SOM clasificó a ambos Pokémon (a priori opuestos) en el mismo grupo:

```
1. 143,0.5,1.0,1.0,2.0,1.0,1.0,2.0,1.0,1.0,0.5,0.0,1.0,1.0,1.0,1.0,4.0,2.0,1.0,
Articuno,ice,flying
2. 145,0.25,1.0,1.0,2.0,0.5,0.5,0.5,1.0,1.0,0.25,0.0,1.0,1.0,1.0,1.0,4.0,0.5,2.0,
Moltres,fire,flying
```

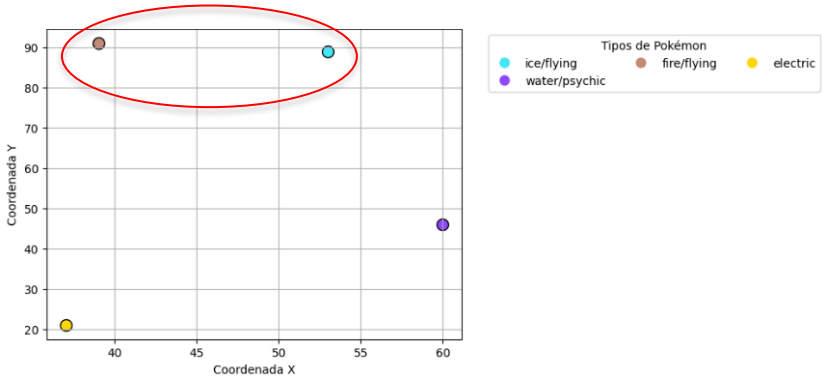

Ilustración 22. Clasificación Articuno y Moltres (Mapa de Clasificación de Prueba)

Moltres se encuentra en la neurona (38, 90) aproximadamente, mientras que Articuno es clasificado en la (52, 90), lo que significa que ambos han sido clasificados en el mismo grupo, ya que son neuronas vecinas, teniendo en cuenta que hay 100x100 neuronas.

e. Por último, podemos afirmar con total seguridad que el Slowbro se encuentra en el grupo correcto por la misma razón que en el caso del Pikachu. Slowbro pertenece al tipo wáter/psyquic, y si vemos cómo organizó el SOM los Pokémon de wáter, wáter/psyquic y psyquic, se confirma nuestra teoría. Slowbro se coloca justo entre medias de ambos grupos principales ("wáter" y "psyquic"), tal y como lo predijo el SOM durante el entrenamiento.

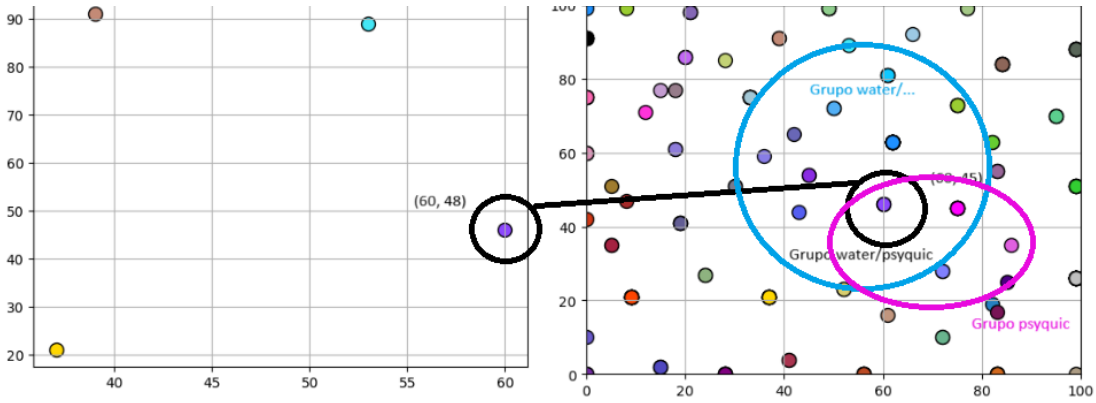

Ilustración 23. Clasificación Slowbro (Mapa de Clasificación de Prueba)

2. **Mapa de activaciones:** En el histograma 3D mostramos la cantidad de patrones que cada neurona ha clasificado.

a. Como es lógico al tratar un dataset con pocos Pokémon, no tenemos casos en los que se repiten tipos de Pokémon, por lo que las neuronas que se han activado son 4, y solamente se han activado una vez.

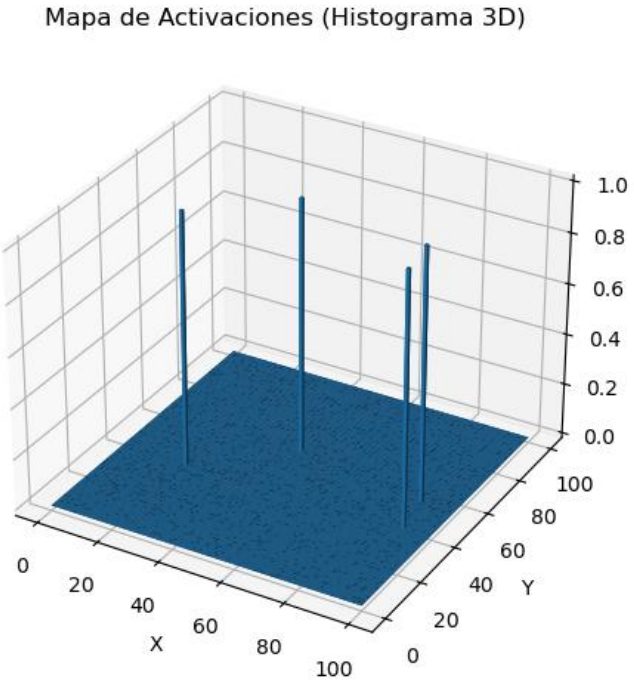

Ilustración 24. Mapa de activaciones de la prueba.

3. **Mapa de distancias del dataset:** Este mapa representa la distancia media entre el color clasificado y los pesos de la neurona BMU para cada posición.

    a. Este mapa, con obviamente pocos datos, muestra una distancia casi insignificante, y mucho menor que la distancia máxima que se encontró en el entrenamiento (0.2).

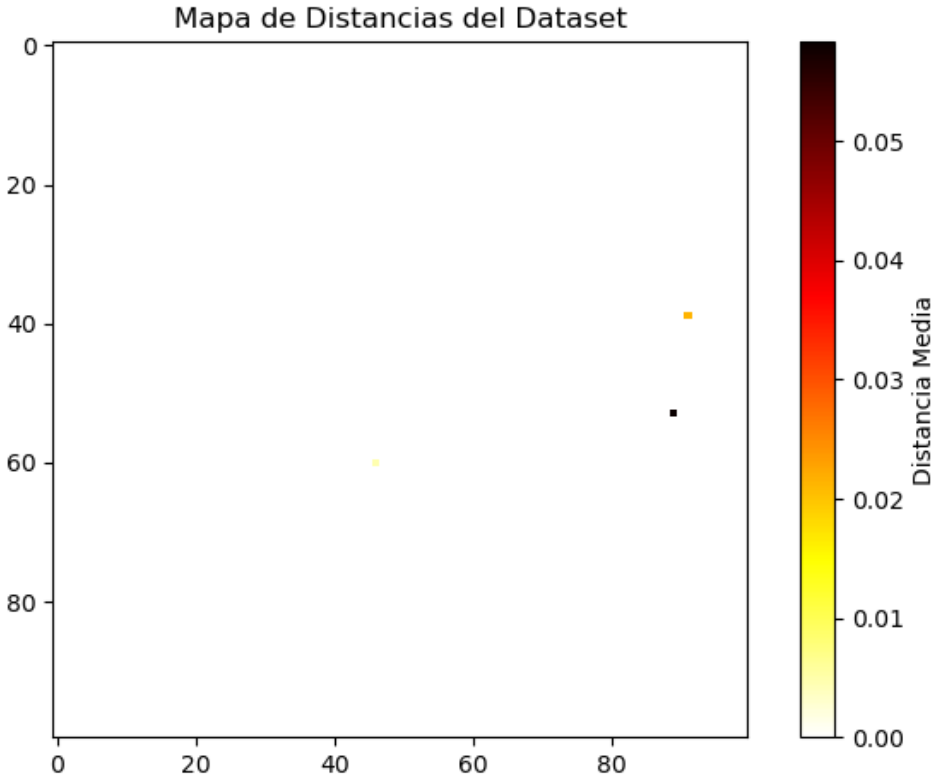

Ilustración 25. Mapa de distancias del dataset de la prueba.

4. **Errores:**

    a. **Error de cuantificación (X):** En el apartado de pruebas, nos salen errores que siguen el modelo de entrenamiento. En este caso podemos afirmar que el modelo ha conseguido generalizar muy bien, ya que contamos con errores de cuantificación muy bajos para datos totalmente nuevos para la red

    b. **Error topológico (X):** En cuanto a la topología, este modelo acata casi a la perfección las reglas de mapeo, por lo que es un claro indicativo de que el modelo es de calidad.

En conjunto, estos resultados nos confirman la capacidad de nuestro SOM para clasificar patrones que formaron parte de su entrenamiento, con un error de cuantificación y un error topológico razonables.

4 Conclusiones

Esta práctica nos ha enseñado la versatilidad de los Self-Organizing Maps a la hora de abordar problemas de distinta complejidad dimensional. En el laboratorio de colores, hemos comprobado cómo el SOM ejemplifica la convergencia de forma intuitiva, agrupando colores por gradientes con un bajo error de cuantificación y topológico. Estos resultados nos confirman la capacidad de la red para organizar y clasificar elementos basándose únicamente en su similitud, sin apoyarse en etiquetas externas.

Por otro lado, al aplicar el SOM a Pokémons con 18 características, hemos visto que la red mantiene un comportamiento robusto y conserva la topología de los datos, logrando integrar resistencias y debilidades de manera coherente. Incluso en casos donde los tipos de Pokémon parecían opuestos (por ejemplo, fire e ice), hemos observado cómo sus valores de daño recibido pueden converger en zonas cercanas del mapa si sus atributos son similares. De esta forma, aprendimos la importancia de seleccionar adecuadamente los hiperparámetros (lado del mapa, tasa de aprendizaje, etc.) y de monitorizar las métricas (error de cuantificación y error topológico) para asegurar resultados de calidad y aprovechar al máximo la capacidad de los SOM en tareas de análisis exploratorio.

5 Referencias

[1] Universidad Politécnica de Madrid, «Capítulo 6: Mapas Autoorganizativos - Introducción al Aprendizaje Automático,» Introducción al Aprendizaje Automático, [En línea]. Available: https://dcain.etsin.upm.es/~carlos/bookAA/06_mapasAutoorganizativosIntro.html.

[2] V. V. Christopher, «Self Organizing Maps Explained,» Built In, 6 Junio 2024. [En línea]. Available: https://builtin.com/articles/self-organizing-maps.

[3] A. Nanda, «Self-Organizing Maps: An Intuitive Guide with Python Examples,» DataCamp, 18 Diciembre 2024. [En línea]. Available: https://www.datacamp.com/tutorial/self-organizing-maps.

[4] FOQUM, «Mapa autoorganizado (Self-organizing map),» FOQUM, 10 Noviembre 2023. [En línea]. Available: https://foqum.io/blog/termino/mapa-autoorganizado-self-organizing-map/.