1. Objetivo del laboratorio

*Desarrollar de forma autónoma una **herramienta de Generación de Mapas Autoasociativos** que permita crear y entrenar redes **SOM** a partir de un dataset de entrada y unos parámetros introducidos en tiempo de ejecución. Usar la herramienta para analizar y resolver dos casos y responder al cuestionario que se plantea para cada uno de ellos.*

2. Elementos a utilizar:

- Lenguaje Python
- Librería numérica *NumPy*, gráfica *Matplotlib* y datasets *Sklearn*
- Entorno Anaconda y Editor Jupyter
- Noteboks de ejemplo proporcionados con el enunciado
- Datasets de entrenamiento y prueba

3. Práctica 1 (clasificar colores)

Objetivo

Construye un SOM que sirva para luego trabajar sobre los dos dataset que se incluyen en la práctica. Como es necesario probar y testear el código según se desarrolla, utiliza el dataset propuesto en el punto 1 de la "**Implementación**" de esta práctica y responde a las preguntas que se plantean en "**Cuestiones**".

Debes de usar comentarios con profusión, incluyendo celdas específicas donde expliques los algoritmos que usas y el código que has programas. Cuantos más comentarios haya, mejor será la evaluación y, probablemente, menos preguntas serán necesarias en la defensa de la práctica.

Implementación

Copia el notebook *L1P1-SOM_colores.ipynb* como *L1P1-SOM_colores-resultado.ipynb*

Cárgalo en Jupyter y usando el esqueleto que contiene, construye una red SOM. Algunas de las funciones ya están incluidas (importación de librerías y rutinas de dibujo de las salidas). El resto debe de ser codificado por el alumno.

El programa en Python deberá incluir necesariamente los siguientes elementos:

1. Implementa el código para obtener el Dataset que se va a usar en el entrenamiento. Crea una matriz de 100 colores al azar usando el código RGB (valores 0 a 255) y llámala "**datos**"
   datos = np.random.randint(valor_min, valor_max, (valores_color, num_colores))

2. **SOM Setup**: Inicializa la red, sus parámetros y atributos. Hay que diferenciar entre:
   - *Variables definidas por teclado*: aquellas que el alumno introduce en cada ejecución. Se asume que el *learning rate* (eta) final es cero.
   - *Variables calculadas*: aquellas que se obtienen mediante instrucciones del lenguaje a partir de las variables definidas en el módulo anterior o bien del *dataset* de entrenamiento.
   - *Funciones para entrenar/clasificar*:
     - *calcular_bmu*: Función para encontrar la neurona ganadora (Best Matching Unit).
       ⇒ Calcula la distancia euclídea entre cada neurona y el patrón. Guarda la neurona cuyo vector de pesos muestre la menor distancia
       ⇒ Guarda las coordenadas (x,y) de la BMU en un array de 2 posiciones (*bmu_idx*) y el vector de pesos de la BMU en *bmu*
     - *variacion_learning_rate*: Función para calcular el descenso del coeficiente de aprendizaje (eta). Devuelve el valor de eta para la i-ésima iteración. Eta final debe de ser cero (se debe de obtener de la correspondiente fórmula)

- *variacion_vecindario*: Función para calcular el descenso del vecindario (v). Devuelve el valor del vecindario para la i-ésima iteración como un float (no truncar). El vecindario final es 1 (se debe de obtener de la correspondiente fórmula)
- *decay*: Función para calcular el descenso del coeficiente de aprendizaje (eta) en función de la distancia a la BMU. Al calcular la variación del vector de peso para las neuronas vecinas, cuanto mas alejados estemos de la BMU menor será la modificación de los pesos. **La función se da completa y no hay nada que codificar aquí**

3. **SOM Entrenamiento:** Entrena la red con el dataset de entrenamiento. Para ello:
   - Dibuja la matriz de pesos iniciales como un mapa de colores RGB con *pintar_mapa*
   - Crea el bucle principal del algoritmo de entrenamiento y dentro de él:
     - Imprime cada 100 iteraciones el número de Iteración
     - Selecciona un patrón al azar del dataset y haz un reshape (¡¡importantísimo!) para convertirlo al formato de la matriz de pesos
     - Busca la BMU
     - Calcula los parámetros que corresponden a esta iteración (eta y v)
     - Actualiza el vector de pesos de la BMU, y el de todas sus vecinas, para acercarlo al patrón en un factor proporcional a la distancia 2-D a la BMU. Considera la BMU como una vecina situada a distancia = 0
       - ⇒ Calcula la distancia euclídea 2-D de cada neurona respecto a la BMU
       - ⇒ Si la distancia ≤ vecindario, calcula la amortiguación de esa neurona y actualiza el vector de pesos de la neurona usando la fórmula

         *new w = old w + learning rate * amortiguación * (patron - old w)*
   - Imprime la matriz de pesos entrenada y dibuja el mapa de colores RGB de la matriz de pesos entrenados

4. **SOM Clasificación:** Clasifica un dataset de patrones con una matriz de pesos entrenada. Para ello:
   - Define tres matrices e inicialízalas a 0
     - *Mapa de clasificación*: matriz de iguales dimensiones que la matriz de pesos para guardar en cada neurona el último patrón clasificado.
     - *Mapa de activaciones*: matriz bidimensional para guardar el número de patrones reconocido por cada neurona
     - *Mapa de distancias*: matriz bidimensional para guardar, para las neuronas con activación>0, la distancia media de todos los patrones de la clase con su vector de pesos
   - Recorre todo el dataset de entrenamiento
     - Imprime las coordenadas (x,y) de la BMU para cada patrón de prueba.
     - Calcula los mapas de clasificación, activaciones y distancias.
     - Calcula el número de clases.
     - Calcula la distancia media (*Error de Cuantificación*) y el *Error Topológico* del mapa.

Cuestiones

Elabora una memoria de la práctica en la que respondas a las siguientes cuestiones

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
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. Deberás de entregar
   2.1. **Gráfico RGB** de los pesos iniciales
   2.2. **Gráfico RGB** de los pesos entrenados

2.3. **Número** de clases**, mapa de clasificación** (gráfico RGB), **mapa de activaciones** (histograma 3D), **mapa de distancias** del dataset, el **error de cuantificación** y el **error topológico**.

3. Sin modificar el entrenamiento, clasifica el dataset de prueba y adjunta los mismos resultados que en el apartado 2.3. Explícalos

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

4. Práctica 2 (clasificar Pokemon)

Objetivo

Usa el programa SOM desarrollado en Práctica 1 para entrenar una red que clasifique los tipos de Pokemon (*Normal, Siniestro, Dragón, Eléctrico, Hada, Luchador, Fuego, Volador, Fantasma, Hierba, Tierra, Hielo, Insecto, Veneno, Psíquico, Roca, Acero y Agua*) que están en el dataset *pokemon_train.csv* y que vienen definidos por un identificador, su nombre, su tipo (podrá tener dos) y 18 valores correspondientes a cómo reaccionan ante el ataque de un Pokemon de otro tipo. Por ejemplo, *against_fire*=2.0 quiere decir que un Pokemon de tipo fuego le hará el doble de daño.

Para ello, lleva a cabo las modificaciones que se proponen en el apartado de "**Implementación**". Responde a las preguntas que se plantean en "**Cuestiones**".

Implementación

1. Copia el notebook *L1P1-SOM_colores-resultado.ipynb* como *L1P2-SOM_pokemon-resultado.ipynb.* Cárgalo en Jupyter.
2. Modifica el código para adaptarlo a este problema. No pintes las matrices de pesos ya que ahora no son tripletes de valores.
3. Entrena la red con el dataset *pokemon_train.csv.*

Cuestiones

Responde en la memoria a las siguientes cuestiones

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 Pokemon (recuerda que algunos son de dos tipos).
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.
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 Pokemon *Articuno* y *Moltres* siendo antagonistas en cuanto al tipo (hielo y fuego)? ¿Qué significa que el cluster donde se ha clasificado a *Slowbro*?

5. Forma de entrega del laboratorio:

La entrega consistirá en un fichero comprimido RAR con nombre LAB01-GRUPOxx.RAR subido a la tarea **LAB1** que **contenga únicamente**

1. **Por cada práctica** un notebook de Jupyter (archivos con extensión **.ipynb**).
2. Una **memoria del laboratorio** en Word.

Las entregas que no se ajusten exactamente a esta norma NO SERÁN EVALUADAS.

6. Rúbrica de la Práctica:

1. IMPLEMENTACIÓN: Multiplica la nota del trabajo por 0/1

Siendo una práctica de IA, todos los aspectos de programación se dan por supuesto. La implementación será:

- Original: Código fuente no copiado de internet. Grupos con igual código fuente serán suspendidos
- Correcta: Los algoritmos SOM están correctamente programados. El programa funciona y ejecuta correctamente todo lo planteado en el apartado "*Cuestiones*" de cada práctica.
- Comentada: Inclusión (**obligatoria**) de comentarios.

2. MEMORIA DEL LABORATORIO

Obligatorio redacción clara y correcta ortográfica/gramaticalmente con la siguiente estructura:

- *Portada con el nombre de los componentes del grupo y el número del grupo*
- *Índice*
- *Resultados de la Práctica 1*
- *Resultados de la Práctica 2*
- *Bibliografía*

Calificación de las cuestiones:

PRÁCTICA	CUESTIÓN	VALORACIÓN (sobre 10)
Práctica 1	Cuestión 1	4
	Cuestión 2	1
	Cuestión 3	1
Práctica 2	Cuestión 1	2
	Cuestión 2	1
	Cuestión 3	1