Practica 4: Mapas Auto-organizados

Profesor: Lauro Reyes Cocoletzi

Edkir Nava, Nancy Galicia, Miguel Sánchez, Daphne González, y Diego Castro
{enavam2001, ngaliciac2100, msanchezz2100, dgonzalezc2104 y dcastroe2100} @alumno.ipnx.mx

UPIIT: Unidad Profesional Interdisciplinaria en Ingeniería Campus Tlaxcala Instituto Politécnico Nacional, Tlaxcala, Tlaxcala, México 9000

Ingeniera en Inteligencia Artificial

20 de octubre 2023

***Resumen*— Un Mapa Autoorganizado (Self-Organizing Map o SOM) es un algoritmo de aprendizaje no supervisado que organiza datos de alta dimensión en una estructura de cuadrícula de neuronas. Cada neurona del SOM representa una región en el espacio de características de los datos, durante el entrenamiento, el SOM ajusta sus pesos para encontrar representaciones topológicas de los datos, agrupando patrones similares en vecindades cercanas en la cuadrícula, esto permite la visualización y segmentación de datos complejos en un espacio de menor dimensión. El tamaño de la cuadrícula de neuronas influye en la granularidad de la representación. Un SOM con una cuadrícula pequeña simplifica, mientras que uno grande ofrece más detalle. Los SOM son útiles para tareas como clustering, reducción de dimensionalidad y análisis exploratorio de datos**

Palabras clave — SOM, sigma, taza de aprendizaje, función de vecindario, épocas/iteraciones

I. Marco Teorico

A. SOM (Mapa autoorganizado)

Un mapa autoorganizado es una técnica de aprendizaje automático no supervisado que se utiliza para producir una representación de baja dimensión de un conjunto de datos de mayor dimensión, preservando al mismo tiempo la estructura topológica de los datos.

Se trata de un tipo de neurona artificial entrenada mediante aprendizaje competitivo.

Características.

- Los datos deben tener un grado de redundancia elevado para realizar su clasificación.
- La red divide el conjunto de datos en distintos subconjuntos (clústers), cada uno de los cuales agrupa a datos similares, con algún tipo de característica en común (clustering).
- El desarrollo de un método de clustering requiere elaborar alguna medida de la semejanza entre los datos (distancia euclidiana, Correlación, etc.).
- Cada clúster se representa por un prototipo`

B. Parametros

Son valores ajustables que permiten controlar el comportamiento y el rendimiento del algoritmo, se establecen antes o durante el proceso de entrenamiento afecta como se organiza y representa la información en la red neuronal.

1. Función de vecindario

   Se trata de una función matemática que permite determinar la actualización de las neuronas vecinas durante el proceso de entrenamiento modelando la influencia que la neurona en particular tiene sobre las neuronas vecinas durante la adaptación de datos

   - Gaussian. Esta función se asemeja a una campana de Gauss y es la más empleada, en donde las neuronas más cercanas a la neurona ganadora se activan de forma más constante a comparación de las neuronas alejadas.
   - Mexican hat. A diferencia de la función Gaussiana, esta función tiene una forma de sombrero mexicano o función Laplaciana, lo que significa que tiene un máximo en la neurona ganadora y disminuye rápidamente a medida que se aleja de ella
   - Bubble. Asigna el mismo valor de influencia a todas las neuronas dentro de un cierto radio alrededor de la neurona ganadora, y fuera de ese radio, la influencia cae bruscamente a cero. Es efectiva en algunos tipos de datos y aplicaciones.

2. Cálculo de las distancias

   - Distancia euclidiana. Fórmula matemática que permite medir la distancia en línea recta entre dos puntos en un espacio n-dimensional.
   - Distancia coseno. Se trata de una medida de similitud que se calcula entre dos vectores distintos de ceros dentro del espacio multidimensional. La distancia del coseno se basa en el ángulo entre dos vectores, en lugar de la distancia euclidiana que se basa en la magnitud de los vectores. Es especialmente útil cuando se trabaja con datos que se pueden representar como vectores de características, como documentos de texto o perfiles de usuarios.
   - Distancia Manhattan. Se trata de la suma de la diferencia absoluta de las coordenadas de los vectores. La distancia de Manhattan es menos sensible a la magnitud de las

diferencias entre las coordenadas que la distancia euclidiana y, por lo tanto, puede proporcionar una perspectiva diferente sobre la similitud o la diferencia entre puntos en un espacio multidimensional.

3. Tamaño de cuadricula de neuronas. Hace referencia a el número de neuronas dentro de una cuadricula o rejillas para organizar los datos en el espacio de características, por ello es importante tener una capacidad de retener la estructura y la viabilidad de los datos.
4. Sigma. Desviación estándar de una función de vecindario, normalmente la función Gaussiana, que se utiliza durante el entrenamiento del SOM. Este parámetro controla el alcance de la influencia de la neurona ganadora sobre sus neuronas vecinas en el proceso de adaptación a los datos.
5. Taza de aprendizaje. Se encarga de controlar la velocidad del ajuste de los pesos o parámetros en repuesta a los datos de entrenamiento, es importante mencionar que se recomienda realizar un ajuste de este parámetro con la finalidad de obtener un equilibrio de la velocidad de convergencia y la estabilidad del entrenamiento.
   - Tasa de aprendizaje baja. El modelo se ajuste a de forma más lenta y estable.
   - Tasa de aprendizaje alta. El modelo se ajusta rápidamente a los datos. Esto puede conducir a convergencia más rápida, pero también puede hacer que el entrenamiento sea inestable, con oscilaciones o saltos en la convergencia.
6. Épocas/iteraciones. Se trata de la cantidad de veces que se presenta todo el conjunto de datos de entrenamiento al modelo durante el proceso de entrenamiento.

C. Cuantización de colores con SOM

La cuantización de colores con SOM tiene varias aplicaciones y ventajas:

Reducción de colores: Permite reducir la cantidad de colores en una imagen. Esto es útil en situaciones donde se necesita reducir el tamaño del archivo de imagen o simplificar la representación de colores, como en la compresión de imágenes.

Segmentación de imágenes: Puede ayudar en la segmentación de imágenes, donde se agrupan píxeles similares en un número limitado de colores representativos. Esto es útil en aplicaciones de procesamiento de imágenes y visión por computadora.

Visualización de datos: La cuantización de colores con SOM también se utiliza a veces para visualizar datos multivariados en un espacio de menor dimensión. Cada neurona SOM representa una clase o grupo de datos, lo que facilita la comprensión de la estructura de los datos.

Mejora de la compresión de imágenes: En algunas técnicas de compresión de imágenes, se utiliza la cuantización de colores para reducir la cantidad de información de color en una imagen, lo que puede ayudar a comprimirla de manera más eficiente.

D. Dataset "Wine"

Se trata de datos obtenidos como el resultado de un análisis químico de vinos cultivados en la misma región de Italia, pero derivados de tres cultivares diferentes. El análisis determinó las cantidades de 13 componentes que se encuentran en cada uno de los tres tipos de vinos.

E. Dataset "Seeds"

Se trata de mediciones de propiedades geométricas de granos pertenecientes a tres variables diferentes de trigo: Kama, Rosa y canadiense, de 70 elementos cada uno, seleccionados al azar para el experimento.

II. Desarrollo

A. Funcionamiento básico del algoritmo SOM

1. Dato de entrada y pesos iniciales:

Antes de la implementación del algoritmo, se debe conocer cómo funciona el algoritmo, para ello utilizaremos como punto de entrada:

punto de entrada = [1.2, 0.8]

Nuestro SOM sera de 2x2 lo que significa que consta de 4 neuronas, recordemos que en nuestro ejemplo cada neurona tiene un vector de pesos bidimensional

Neurona (0,0) 1: [0.54, 0.71] Neurona (0,1) 2: [0.60, 0.54] Neurona (1,0) 3: [0.42, 0.64] Neurona (1,1) 4: [0.43, 0.89]

2. Calcular la Neurona Ganadora (distancia euclidiana):

El siguiente paso que es calcular la distancia, para ello tenemos diversos tipos, pero los podemos reducir en:

- La distancia euclidiana es una elección común, especialmente cuando estás trabajando en un espacio euclidiano.
- La distancia de coseno es útil cuando deseas medir similitud direccional y no te importa la magnitud.
- La distancia de Manhattan es adecuada cuando se trabaja en un espacio con restricciones de movimiento en líneas rectas, como una cuadrícula.

En nuestro ejemplo usaremos la distancia euclidiana, por lo que en nuestra formula tenemos

$$distancia^2 = (A_x - B_x)^2 + \left(A_y - B_y\right)^2$$

En este caso A_x, A_y son las coordenadas del punto de datos y B_x, B_y son las coordenadas de la neurona. Por lo consiguiente, sustituimos cada uno de nuestros valores

La neurona 2 (0, 1), tiene la distancia euclidiana más pequeña, por lo que es la neurona ganadora.

3. Actualización de Pesos de la Neurona Ganadora:

La fórmula para actualizar los pesos de la neurona ganadora en un mapa autoorganizativo (SOM) es:

$$NuevoPeso_{i,j} = PesoActual_{i,j} + TasaAprendizaje * (DatoEntrada - PesoActual_{i,j})$$

Donde la $TasaAprendizaje$ es la tasa de aprendizaje, un hiperparámetro que controla cuánto se ajustan los pesos en cada iteración. En nuestro ejemplo supongamos que la tasa de aprendizaje es de 0.4, por lo que sustituyendo nos queda:

$$Neurona\ 2 = 0.60 + 0.4 * (1.2 - 0.60) = 0.84$$

$$Neurona\ 2 = 0.54 + 0.4 * (0.8 - 0.54) = 0.64$$

Los pesos de las otras neuronas permanecen sin cambios.

4. Nuevos Pesos del SOM:
Después de esta única época de entrenamiento, los pesos de las neuronas del SOM se actualizarán, y los nuevos pesos de las neuronas ganadoras serán:

Neurona (0,0) 1: [0.54, 0.71]
Neurona (0,1) 2: [0.84, 0.64]
Neurona (1,0) 3: [0.42, 0.64]
Neurona (1,1) 4: [0.43, 0.89]

Con estos nuevos pesos podemos realizar más épocas de entrenamiento para ajustar más el SOM

B. APLICACIÓN CON EL DATASET DE "WINE"

Para la aplicación de SOM, se usó el dataset de "wine", donde se siguieron los siguientes pasos:

1. Carga de datos y preparación
   - Como se explicó en la parte del marco teórico, para las etiquetas se toma la primera columna, las demás columnas son las que ingresaran al algoritmo
   - Escalamos los datos para asegurarse de que todas tengan la misma escala.

2. Creación y entrenamiento del SOM:
   - Creamos un SOM de diferentes dimensiones para observar el comportamiento de los datos y podemos determinar cuál es un tamaño adecuado para los tipos de datos que tenemos. A su vez, los pesos se llenan de manera aleatoria
   - Posteriormente se entrenó el SOM con diversas épocas de entrenamiento, con la finalidad de observar los cambios que ocurren en distintas épocas, también se usó diversos valores en la tasa de aprendizaje y el radio de influencia de las neuronas

Neurona 1: $(1.2 - 0.54)^2 + (0.8 - 0.71)^2 = 0.43$
Neurona 2: $(1.2 - 0.60)^2 + (0.8 - 0.54)^2 = 0.42$
Neurona 3: $(1.2 - 0.42)^2 + (0.8 - 0.64)^2 = 0.62$
Neurona 4: $(1.2 - 0.43)^2 + (0.8 - 0.89)^2 = 0.58$

-

3. Mapeo de activación y etiquetas de clase:
   - Se realiza un mapeo de activación para obtener información de cuantas veces las neuronas fueron seleccionadas, sin embargo, no haremos un análisis en ese sentido

4. Asignación de etiquetas de clase:
   - Se realiza la relación de las etiquetas originales con los datos obtenidos, esto para conocer la distribución que tuvieron, esto nos fue muy útil en la visualización de los resultados

5. Visualización del SOM:
   - Usamos como cmap el de "Pastel2" por la gama de colores que nos presenta, esto nos sirve para mostrar el mapa de distancia
   - Además, se agregó una barra de color para indicar la escala de distancia.
   - Los ejes se etiquetan con índices correspondientes a las posiciones de las neuronas en el SOM.
   - también se graficó un mapa de dispersión para observar donde exactamente se encuentran los datos

C. APLICACIÓN EN LA CUANTIZACION DE COLORES DE UNA IMAGEN

La cuantización de colores de una imagen utilizando el algoritmo de Mapas Autoorganizados (SOM), sigue el siguiente funcionamiento:

1. Inicialización de la red SOM
   - En el caso de usar el algoritmo de SOM la cuadrícula de neuronas que se crean, nos servirán como "representantes" de los colores. La cantidad de neuronas en la cuadrícula que se elige es en función de la cantidad de colores distintos que deseamos obtener en la imagen cuantizada, entre más chica, obtendremos una imagen sin tantos colores. Nosotros usamos distintos valores del tamaño de las neuronas, para comprobar el funcionamiento de cada uno

2. Entrenamiento de la red:
   - Una vez obtenida la red de neuronas SOM, se entrena utilizando los colores de los píxeles de la imagen original. Durante el entrenamiento, la red

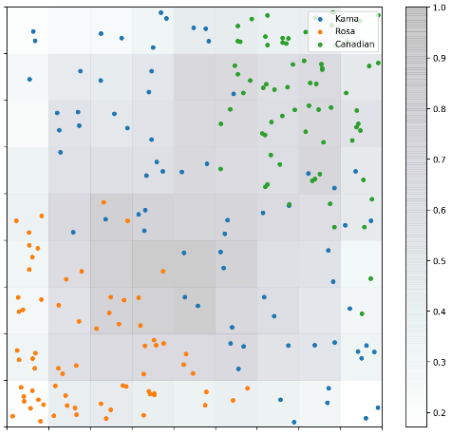
SOM ajusta sus neuronas para que cada una de ellas represente un color en el espacio de colores de la imagen, tenemos los mismos parámetros que con los que contamos en la aplicación con el data set de wine, sin embargo, agregamos la función de vecindario que determina cómo se propagan las actualizaciones de pesos

3. Cuantización
   - Después del entrenamiento, cada neurona en la red SOM representa un color específico. Para cuantizar una imagen, se asigna a cada píxel de la imagen el color de la neurona SOM más cercana en el espacio de colores.

D. Aplicación en el dataset de "seeds"

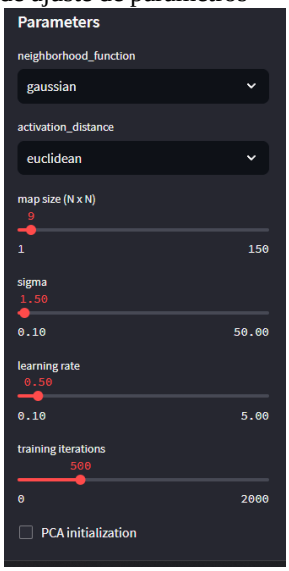
Un tercer análisis del algoritmo de SOM, se usó la página web de "streamlit.app" la cual nos presenta:

1. Área de visualización de neuronas con



En esta área podemos ver el resultado de aplicar diferentes parámetros al entrenamiento del algoritmo SOM

2. Ventana de ajuste de parámetros



En esta área podemos modificar diferentes parámetros del algoritmo de SOM, todos estos parámetros al modificarlos, podemos vere la actualización en tiempo real en el área de trabajo, esto facilita comprender de una manera visual el funcionamiento del algoritmo y cómo se comporta modificando ciertos valores

III. Resultados

A. Funcionamiento básico del algoritmo SOM

Para comprobar el funcionamiento del algoritmo descrito en la sección del desarrollo del mismo apartado, usamos el código dado y lo comprobamos con la misma entrada usa en el ejemplo, obteniendo:

Neurona (0,0) 1: [0.54, 0.71]
Neurona (0,1) 2: [0.84, 0.64]
Neurona (1,0) 3: [0.42, 0.64]
Neurona (1,1) 4: [0.43, 0.89]

Al obtener los mismos resultados, comprobamos que el enfoque es el correcto, ahora podemos experimentar con diferentes valores, comenzando con entrada, aumentaremos el tamaño del vector para obtener más cambios en nuestras neuronas

punto de entrada = [1.2, 0.8], [0.4, 0.6], [3.0, 2.9], [2.9, 2.5]]

El mapa obtenido a partir de los datos de entrada anteriores:

Neurona (0,0) 1: [0.54, 0.71]
Neurona (0,1) 2: [2.18, 1.92]
Neurona (1,0) 3: [0.41, 0.62]
Neurona (1,1) 4: [0.43, 0.89]

No obtenemos grandes cambios, pero esto por que lo ejecutamos solamente una época, cuando lo incrementamos a 100:

Neurona (0,0) 1: [1.2, 0.8]
Neurona (0,1) 2: [2.93, 2.65]
Neurona (1,0) 3: [0.4, 0.6]
Neurona (1,1) 4: [0.43, 0.89]

Observamos el comportamiento principal de esta algoritmo, el cual es que las neuronas al estar constantemente actualizando sus pesos, van a tener la tendencia de parecerse a los puntos de entrada, como es el caso de la neurona 1, el cual sus pesos después de 100 épocas es de [1.2, 0.8] y si observamos el primer dato de entrada tenemos exactamente los mismos valores ([1.2, 0.8]), esto ocurre por la cantidad de datos, donde prácticamente se le esta asignando a esa neurona ese punto, ahora comprobemos que ocurre cuando hacemos 10,000 épocas

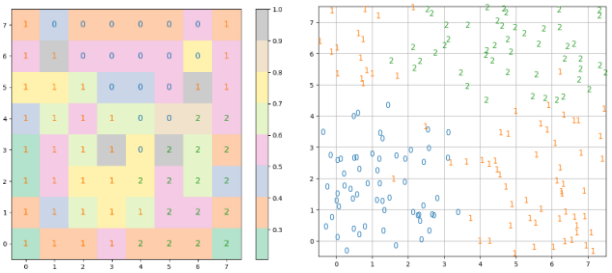
Neurona (0,0) 1: [1.2, 0.8]
Neurona (0,1) 2: [2.93, 2.65]
Neurona (1,0) 3: [0.4, 0.6]
Neurona (1,1) 4: [0.43, 0.89]

Observamos que son los mismos valores que cuando fueron 100 épocas, esto tiene que ver con la cantidad de datos de entrada, esto nos demuestra que el algoritmo SOM son altamente dependiente de la cantidad de datos y la diversidad de los datos de entrada.

B. APLICACIÓN CON EL DATASET DE "WINE"

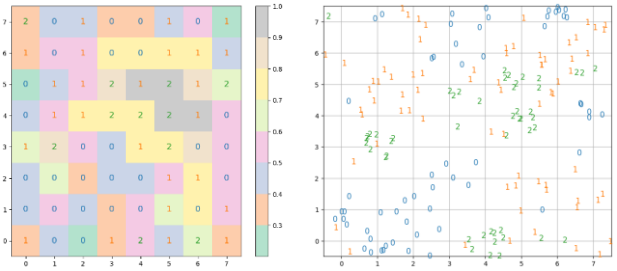
Para ir comparando los diferentes resultados que obtengamos al cambiar los parámetros de: sigma, taza de aprendizaje y épocas, establecimos el resultado más estable, donde:

- Sigma: 1
- Taza de aprendizaje: 0.5
- Épocas: 5,000
- Tamaño de cuadricula: 7x7

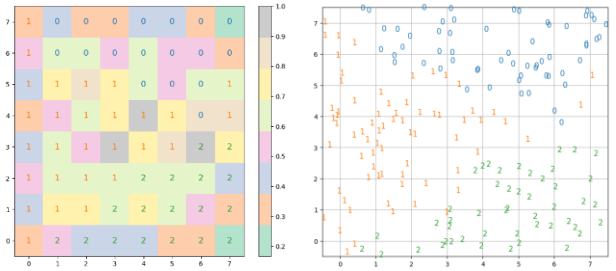


Observamos que los datos se agrupan de manera adecuada, además se ocupa todo el espacio de las neuronas, además con la gráfica de color, podemos ver la densidad de los datos, y diferencias las diferentes áreas. Y con el grafico de puntos observamos como se están distribuyendo cada dato. Partiendo de los parámetros anteriores, los modificamos para saber el comportamiento que esta tomado el algoritmo, comenzando por el número de épocas:

Época: 100

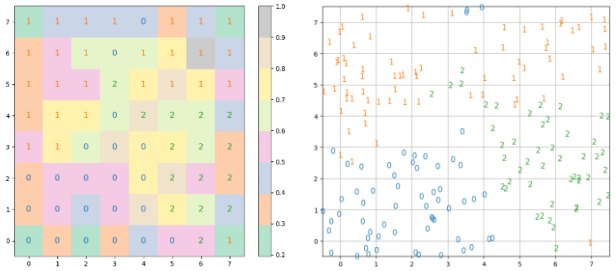


Época: 50,000

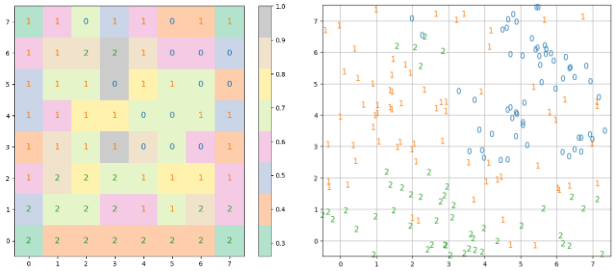


Cuando usamos un número bajo de iteraciones/épocas, como es el caso de usar 100, el SOM no tiene suficiente tiempo para ajustar completamente sus neuronas a los datos, lo que lleva a agrupaciones dispersas y menos definidas, como se observa en la imagen que de un mismo grupo hay varias agrupaciones por toda la red de neuronas. Por otro lado, cuando usamos un número alto de iteraciones, como 50,000, el SOM ajusta más sus neuronas y logra agrupaciones más definidas, sin embargo, usar un numero alto de iteraciones lleva a un mayor riesgo de sobreajuste, especialmente si los datos son ruidosos, por lo que se debe buscar el mejor parámetro

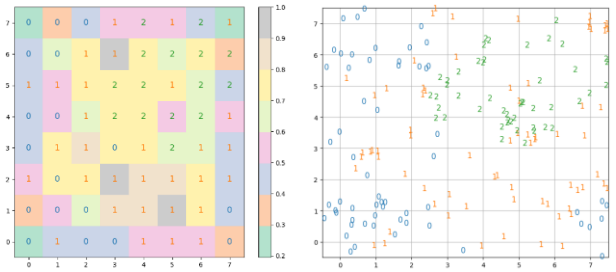
Taza de aprendizaje: 0.01



Taza de aprendizaje: 4



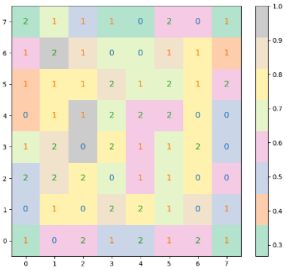
Taza de aprendizaje: 4.2

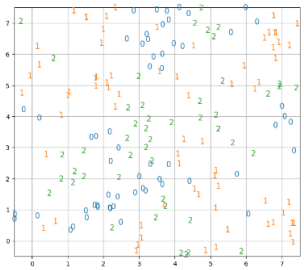


Al aumentar la tasa de aprendizaje, se observa que las neuronas se adaptan más rápidamente a los patrones presentes en los datos. Esto puede resultar en una

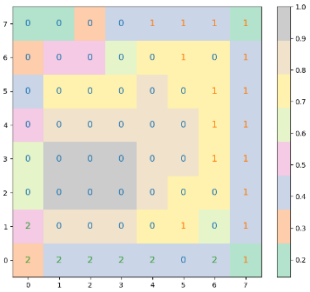
agrupación más rápida de las neuronas en áreas con alta densidad de datos, lo que a su vez puede requerir menos iteraciones para lograrlo. Sin embargo, al comparar imágenes generadas con tasas de aprendizaje ligeramente diferentes, como 4 y 4.20, se pueden notar diferencias significativas. Esto se debe a que valores de tasa de aprendizaje muy altos pueden hacer que el algoritmo SOM sea altamente sensible al ruido en los datos, lo que puede ocasionar cambios bruscos en la posición de las neuronas, lo que afecta la organización y estabilidad del mapa

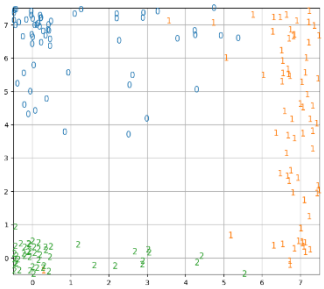
Sigma: 0.1





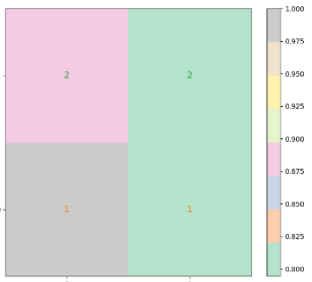
Sigma: 10

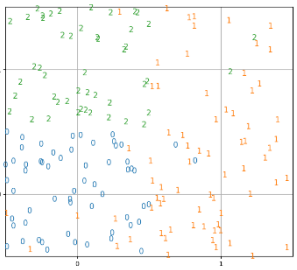




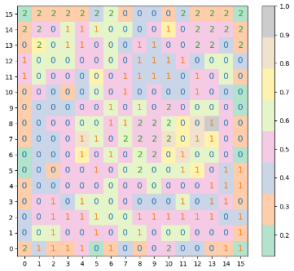
Comparando entre los dos extremos de valores para el parámetro sigma, vemos que se obtienen dos resultados muy distintos, cuando se utiliza un valor de sigma muy bajo, como 0.10, lo que se está haciendo es reducir el alcance de influencia de cada neurona. Esto significa que las neuronas se vuelven muy específicas y se especializan en representar regiones de datos muy pequeñas y específicas, en este caso vemos una gran concentración de datos en ciertas regiones, pero ocupan la mayoría de la región de las neuronas. Por otro lado, al utilizar un valor de sigma tan alto como 10, se permite que las neuronas tengan un rango de influencia mucho más amplio. En este caso, las neuronas representan regiones más grandes del espacio de características, como vemos en la imagen donde se observa como en regiones muy específicas se concentraron los datos.

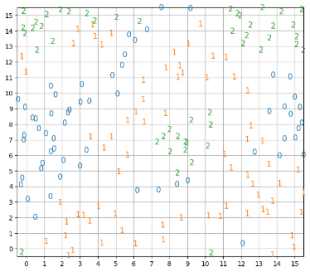
Tamaño de cuadricula: 2 x 2





Tamaño de cuadricula 16 x16



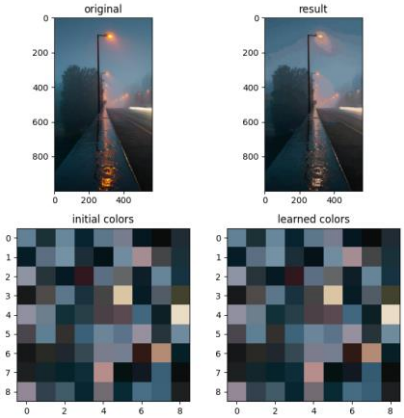


Cuando usamos una cuadricula muy pequeña observamos que los datos si bien se agrupan en zonas específicas, esto no es detallado y tienen pocas características distintivas. A diferencia de usar una cuadricula enorme donde hay neuronas que no se están ocupando, para este dataset no es recomendable usar una cuadricula enorme, en algunos donde dataset que tengamos una gran cantidad de datos, puede ser una buena opción usar cuadriculas más grandes, considerando también que el número de iteraciones será mayor.

C. Aplicación en la cuantizacion de colores de una imagen

Para comprobar el funcionamiento y aplicación de SOM en la cuantización de una imagen, contamos con diferentes parámetros, los cuales iremos usaremos como base:

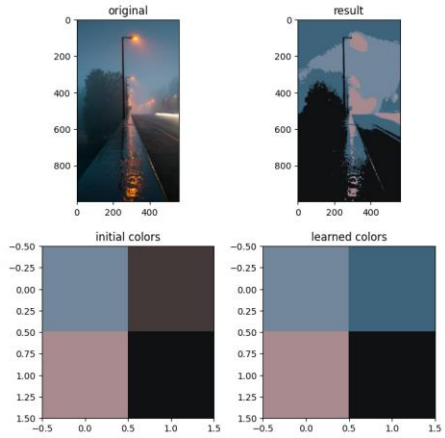
- Función de vecindario: bubble
- Tamaño de cuadricula: 9x9
- Sigma: 1
- Taza de aprendizaje: 0.20
- Épocas: 10,000



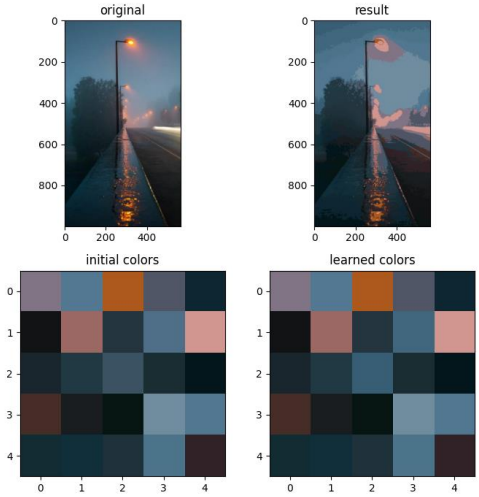
1. Tamaño de cuadrícula de neuronas

   Los tamaños de la cuadrícula de neuronas de la red SOM, los veremos de la forma de (X, Y, dimensión), donde la dimensión en todos los casos será de 3 ya que las pruebas se realizaron con una imagen RGB.

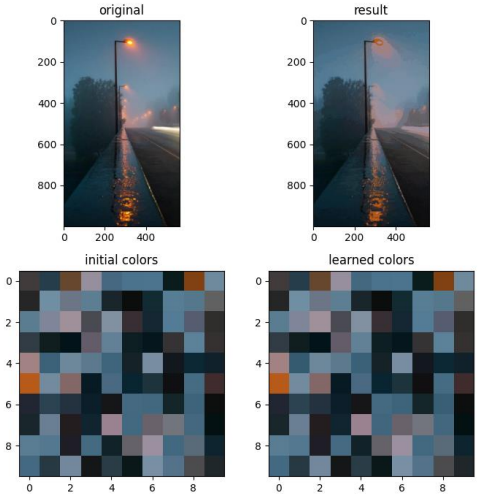
   Usando un tamaño de 2,2:



Tamaño 5,5



Tamaño 10, 10

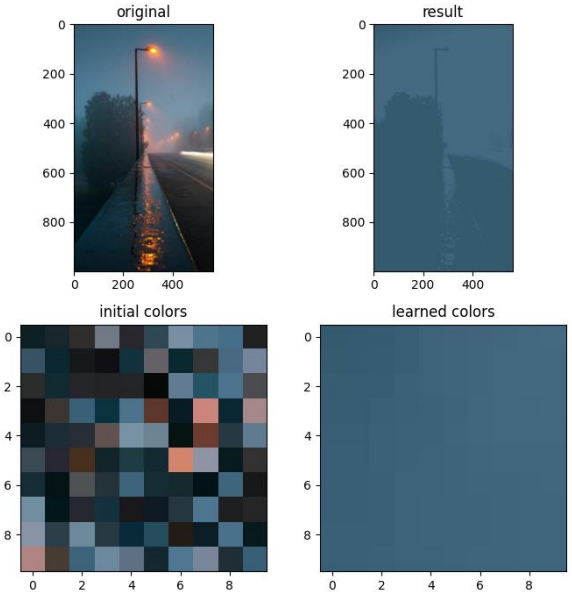


Observamos que al reducir la cuadricula, se aprecia que se reduce significativamente la capacidad de la red para representar la distribución de colores en la imagen, es decir con una cuadrícula más pequeña, cada neurona debe representar un rango más amplio de colores, lo que resulta en una representación menos definida de la imagen original.

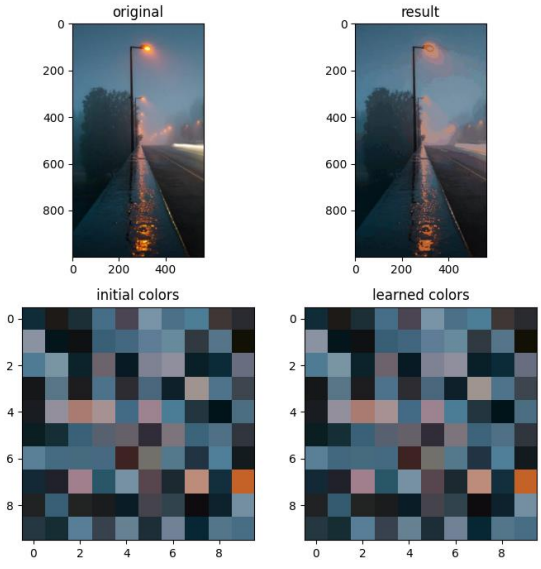
Por otro lado, cuando aumentamos el tamaño de la cuadrícula, como 5,5 o 10, 10, se está permitiendo que la red SOM tenga más neuronas para representar y organizar los colores. Esto significa que cada neurona se especializa en un rango más estrecho de colores, lo que puede resultar en una representación de mayor calidad y detalle de la distribución de colores en la imagen original.

2. Sigma

   Sigma = 2



Sigma = 0.1



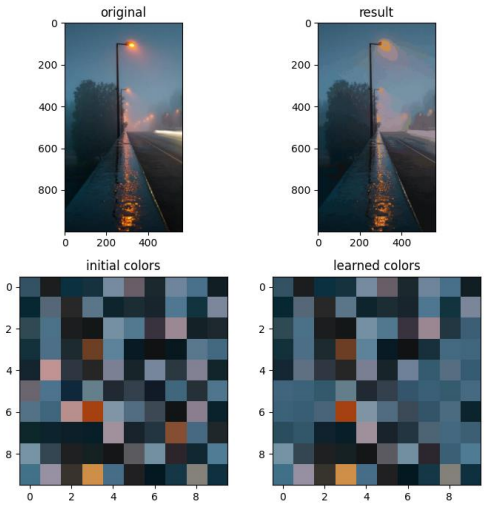
Con los resultados comparando dos extremos de sigma, es decir uno muy alto y uno muy bajo, en el caso de poner una sigma muy alto, el vecindario de influencia es grande y afecta a la mayoría o todas las neuronas en la red SOM, esto nos llevó a que la cuantización sea menos precisa debido a que las neuronas representarán colores muy diversos. En lugar de agrupar colores similares, la red SOM puede aprender a representar la totalidad del espectro de colores en la imagen.

Por otro lado, cuando sigma es muy bajo el vecindario de influencia es pequeño y afecta solo a unas pocas neuronas cercanas a la neurona ganadora, esto permitió que la cuantización sea más precisa y detallada, ya que las neuronas aprenden a representar colores muy similares entre sí, la imagen cuantizada obtenida tuvo menos matices y se acercará más a una versión simplificada

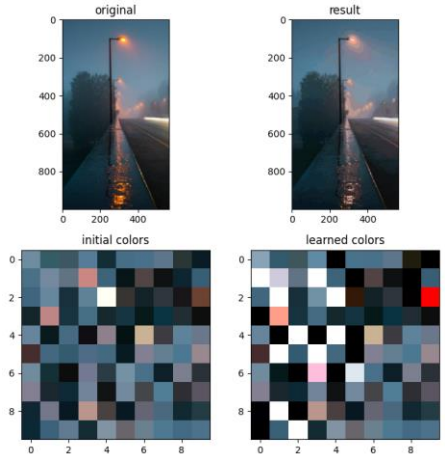
3. Función de vecindario

   La función de vecindario se parece mucho al concepto de sigma, sin embargo, la diferencia está en que sigma controla la extensión espacial del vecindario, mientras que la función de vecindario controla cómo se distribuyen las actualizaciones de peso dentro de ese vecindario.

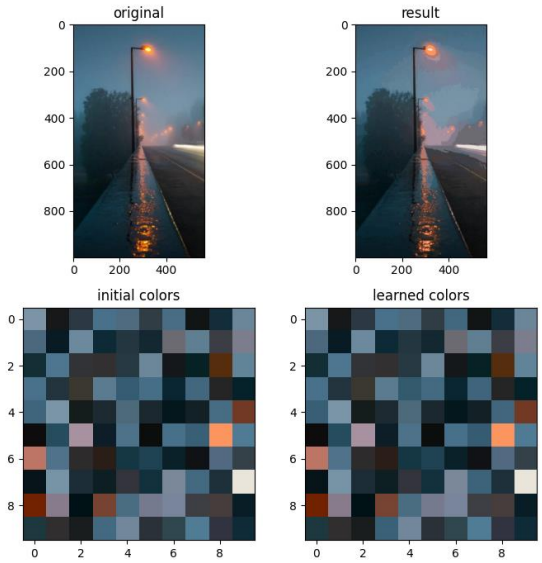
   Gaussian:



Mexican Hat:



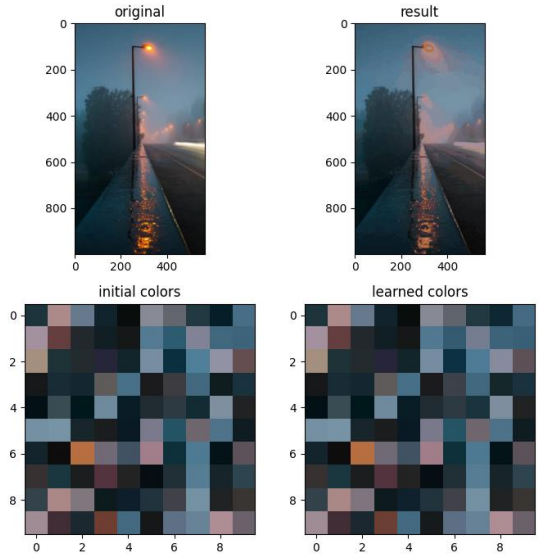
Bubble:



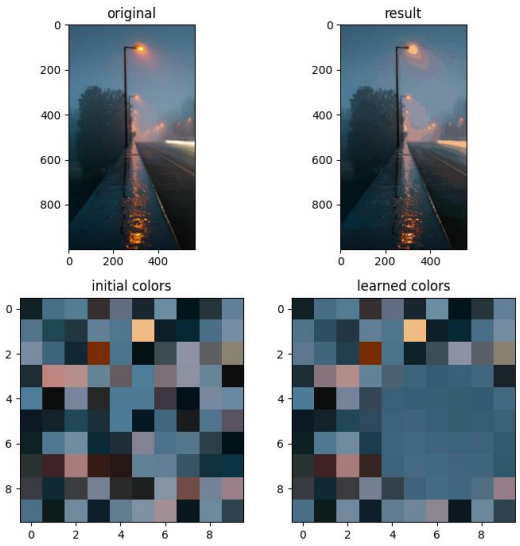
Las diferencias entre los diferentes métodos no es tan abrupta si lo vemos desde la imagen obtenida, sin embargo, si notamos los colores aprendidos logramos ver las diferentes, por ejemplo, cuando usamos la función de Gaussian podemos ver como se produce una transición suave, esto hace que notar la diferencia de algunos colores no es tan notable, a diferencia de usar Mexican Hat donde vemos como los colores se agrupan de manera más abrupta y con una diferenciación más clara, esto se nota con los blancos y rojos. Por último, usamos Bubble donde la distribución es de manera más uniforme, podemos verlo como un punto medio entre los dos algoritmos de Gaussian y Mexican Hat, en este caso es con el que se obtuvo mejores resultados, sin embargo, no es definitivo, también influyen los otros parámetros usados como la taza de aprendizaje, sigma, numero de neuronas, etc.

4. Taza de aprendizaje

Taza de aprendizaje = 0.02



Taza de aprendizaje = 2

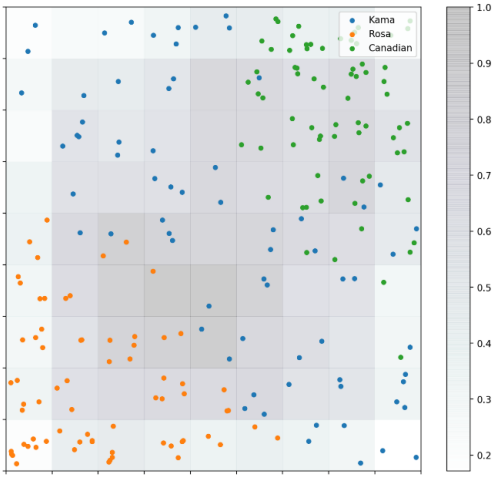


Al modificar el parámetro de la taza de aprendizaje, cuando es muy alto las actualizaciones de los pesos de las neuronas son grandes en cada iteración, es decir las neuronas se ajustan rápidamente a los datos de entrada, sin embargo, esto nos puede llevar a tener una cuantización de colores menos suave y a una mayor probabilidad de sobreajuste, donde la red SOM podría capturar demasiados detalles específicos de la imagen en lugar de resumir de manera efectiva la distribución de colores. Por otro lado, tener una taza de aprendizaje muy baja, las actualizaciones en cada iteración serán muy pequeñas, sin embargo, debemos tener cuidado porque es menos sensible a las variaciones en los datos de entrada, lo que puede resultar en una cuantización de colores menos precisa y menos representativa de la imagen original.

D. APLICACIÓN EN EL DATASET DE "SEEDS"

Al ingresar al simulador se nos presentan los parámetros recomendados, donde se utiliza:

- Función de vecindario: Gaussian
- Cálculo de las distancias: euclidiana
- Tamaño de cuadricula: 9x9
- Sigma: 1.50
- Taza de aprendizaje: 0.50
- Épocas: 500

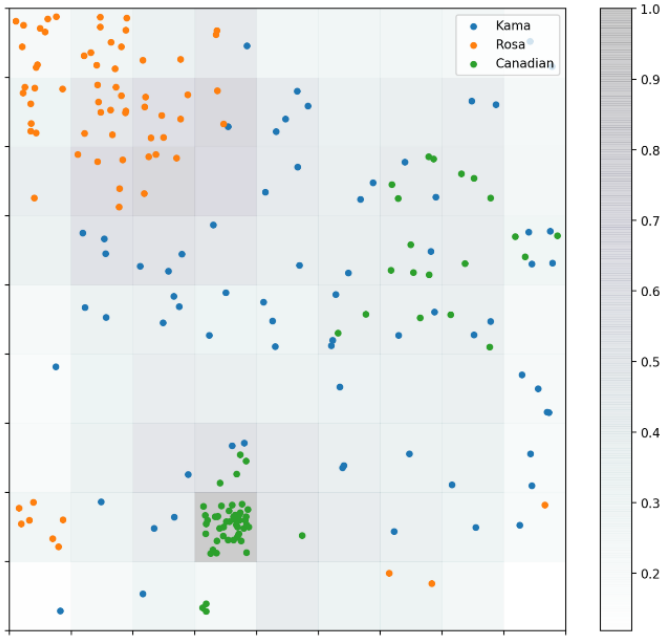


Los resultados con los parámetros sugeridos por el simulador son buenos pues inicialmente se pueden distinguir a simple vista una separación de los tres tipos de semillas: *Karma*, *Rosa* y *Canadian*.

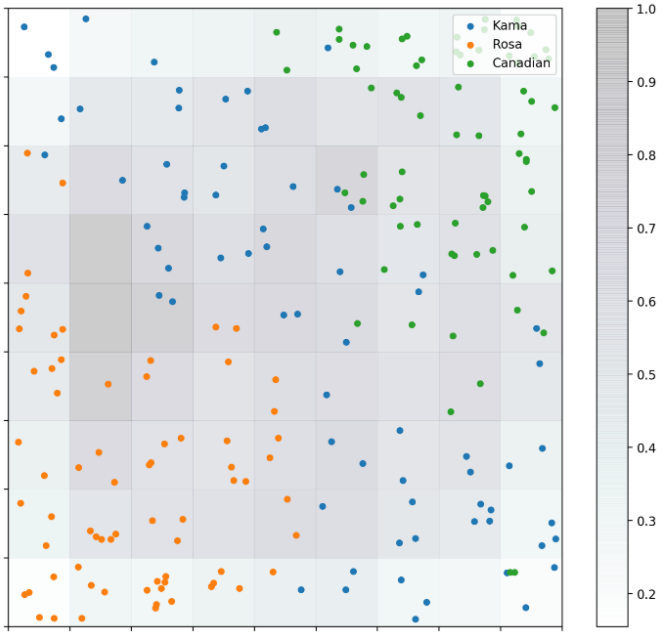
Aunque los resultados son satisfactorios, aun se pueden mejorar y nos sirve como un punto de referencia para comparar con los resultados usando otros parámetros.

Al cambiar el valor en *Época* observamos que para las semillas de tipo Canadian (puntos verdes) tienen una fuerte concentración en una neurona específica. Esto quiere decir que las neuronas aun no modifican lo suficiente sus pesos para distribuir de una manera más eficiente los datos. También observamos que existe una gran dispersión de la clase *Karma* (puntos azules).

Época:150

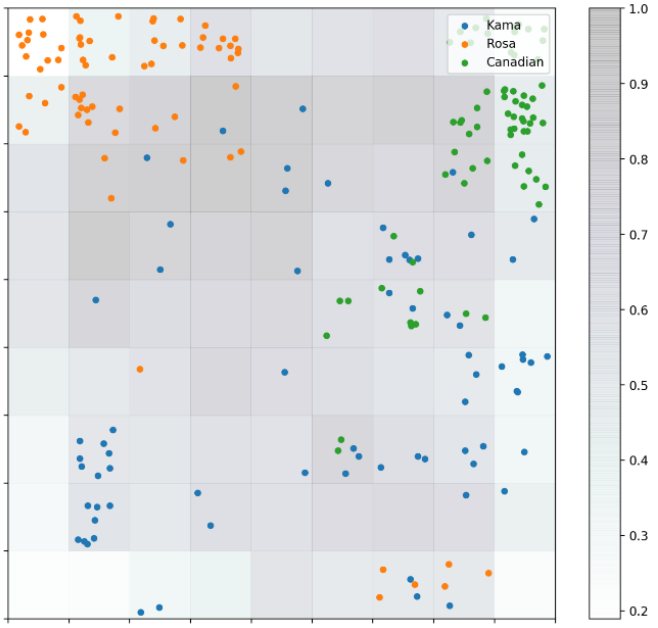


Época: 1000



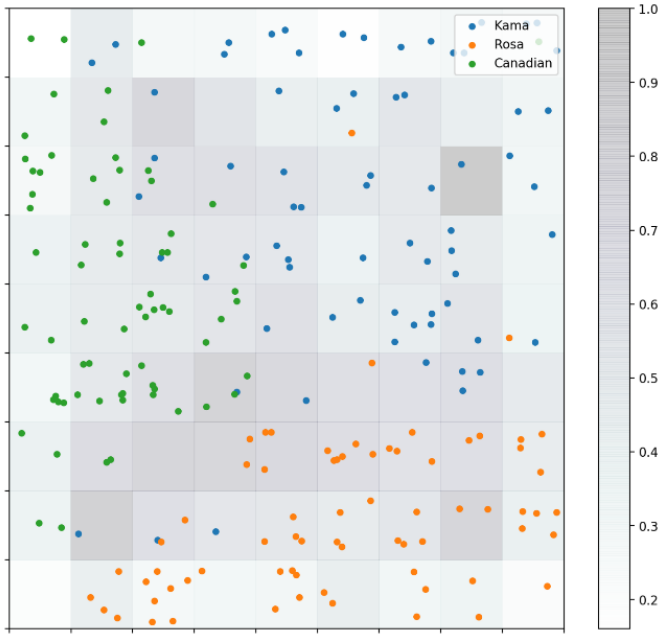
Al aumentar al doble las *épocas* originales, se observa una gran similitud a la imagen original, sin embargo, los datos se distribuyen a lo largo de todas las neuronas, lo que indica que se están adaptando adecuadamente a los datos, pero podemos caer en un sobre ajuste de los datos.

Taza de aprendizaje: 0.10

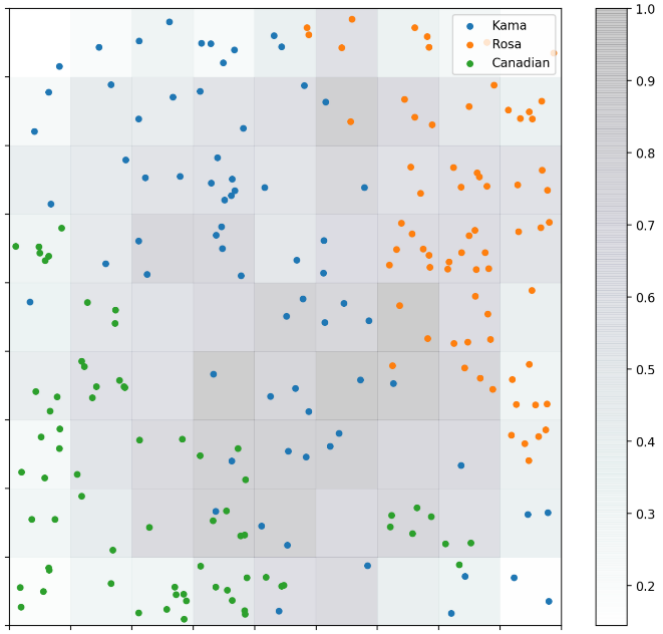


Cuando se usa una taza de aprendizaje muy lenta los pesos se actualizan lentamente y hace que las neuronas pueden quedar atrapadas en óptimos locales. Esto implica que no se ajustan adecuadamente a la distribución de los datos y forman grupos pequeños y aislados como observamos en la imagen.

Taza de aprendizaje: 4

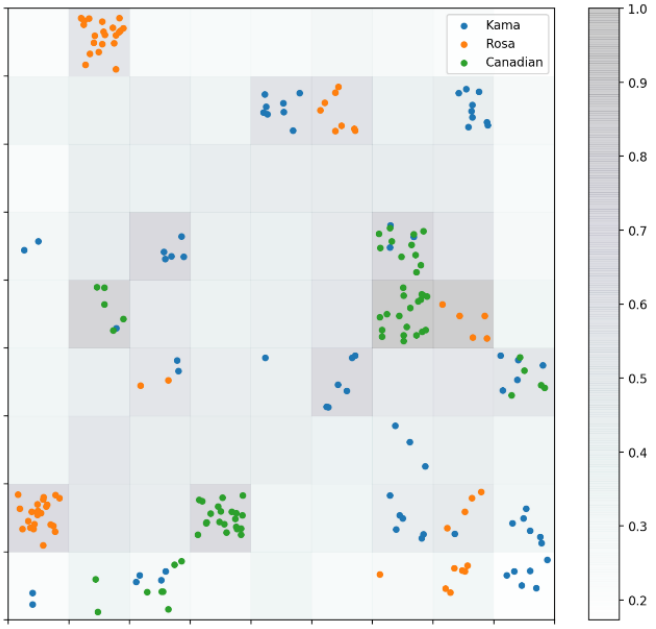


Taza de aprendizaje: 4.20

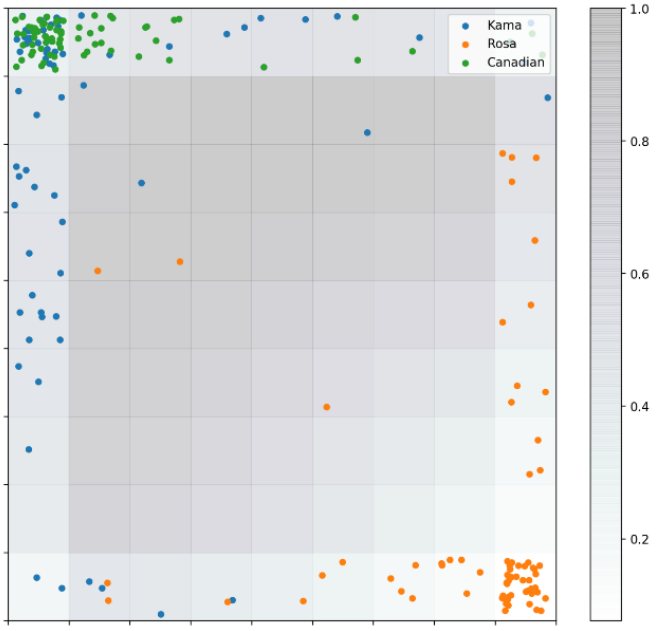


Al aumentar la taza de aprendizaje obtenemos que las neuronas se adaptan rápidamente a los patrones en los datos, lo que lleva a una agrupación rápida de las neuronas en áreas con mucha densidad de datos. Esto se obtiene con menos iteraciones, sin embargo, al observar en las imágenes para las tazas de aprendizaje iguales a 4 y 4.20, hay una diferencia notable entre ambas. Lo anterior porque ya se tratan de valores muy altos que pueden hacer que el algoritmo SOM sea sensible al ruido y las neuronas cambien de posición demasiado rápido.

Sigma: 0.10

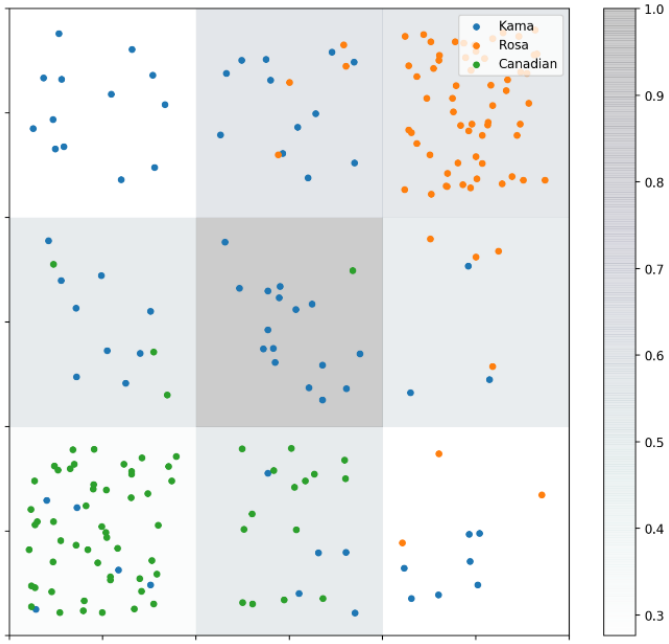


Sigma: 15

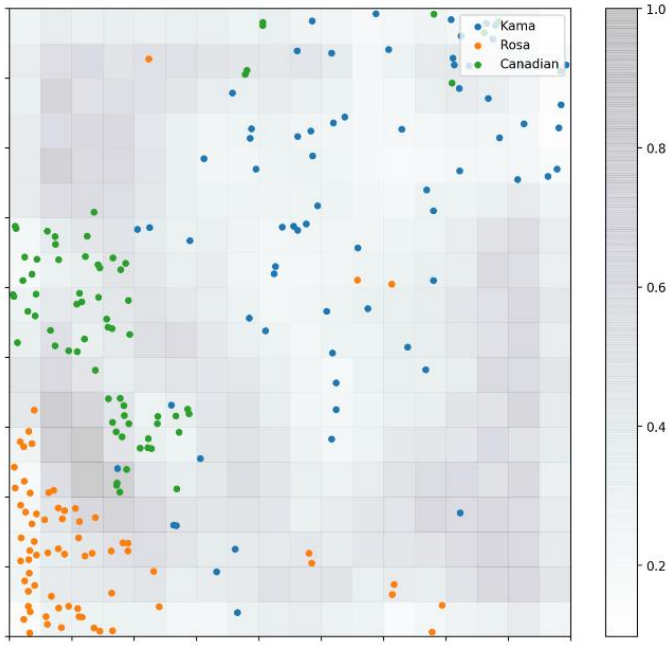


Comparando entre los dos extremos de sigma los resultados son muy distintos. Para un valor de sigma muy bajo como 0.10, se está restringiendo el alcance de influencia de cada neurona, es decir las neuronas son muy específicas y se especializan en representar regiones de datos muy pequeñas y específicas, por ello se ve como cada neurona tiende a capturar un subconjunto muy limitado de datos y a formar grupos pequeños y compactos. Por el contrario, usar un valor de sigma tan alto como 15, permite que las neuronas tengan un rango de influencia mucho más amplio. Las neuronas representan regiones más grandes del espacio abarcando una mayor variabilidad de datos.

Tamaño de cuadricula de neuronas: 3x3



Tamaño de cuadriculo de neuronas: 18x18



Al usar una cuadricula pequeña, el SOM es simple y captura patrones generales en los datos, pero no es muy detallado, a diferencia de usar una cuadricula más grande como 18 x18, donde hay neuronas que no se están ocupando. Regularmente tamaños de cuadricula grandes son recomendados cuando se tiene una gran cantidad de datos, pero se necesita de más entrenamiento para distribuirlos de manera más efectiva.

C. Conclusiones

En esta práctica, utilizamos el algoritmo de Mapas Autoorganizados (SOM) para analizar dos conjuntos de datos diferentes y una imagen para su ecualización, recapitulando, el SOM es una herramienta efectiva para analizar y agrupar datos en función de similitudes y patrones de los datos propuestos. Sin embargo, en cada sección donde aplicamos el algoritmo lo fundamental es la sintonización de nuestro algoritmo, es decir:

- Tamaño de la cuadricula: uno de los primeros parámetros que se debe ajustar, es la elección del tamaño de la cuadricula a utilizar, cuando usamos uno muy bajo, el resultado no será muy detallado, y será muy general, es decir, podemos perder características diferenciadoras, caso contrario al usar una cuadricula muy grande, la cuadricula resultado se lograran identificar grupos de neuronas que no están siendo aprovechadas, se recomienda esta opción cuando tenemos una gran cantidad de datos y estamos dispuestos a darle un mayor entrenamiento
- Sigma: al usar un numero de sigma muy bajo se vio claramente el poco alcance que llega a tener la neurona ganadora, esto genera varios grupos muy específicas y compactos, sin embargo, aumentar exageradamente el valor de sigma obtienes regularmente en las esquinas
- Taza de aprendizaje: cuando modificamos este parámetro, la taza de aprendizaje alta permite que el SOM se ajuste rápidamente a los datos, pero también puede llevar a una convergencia prematura o a una representación inestable. Por otro lado, una tasa de aprendizaje baja permite una convergencia más estable y cuidadosa, pero a expensas de una velocidad de entrenamiento más lenta. Es importante ajustar este parámetro cuidadosamente, probando diferentes valores para encontrar un equilibrio adecuado.
- Época/iteraciones: al momento de seleccionar un número muy bajo el SOM no tiene suficiente tiempo para ajustar completamente sus neuronas a los datos, lo que lleva a agrupaciones dispersas y menos definidas Por otro lado, cuando usamos un número alto de iteraciones, este ajusta más sus neuronas y logra agrupaciones más definidas, sin embargo, usar un numero alto de iteraciones lleva a un mayor riesgo de sobreajuste, especialmente si los datos son ruidosos,

D. Bibliografia

[1] "Parámetros de entrenamiento". Amazon Machine Learning. Accedido el 20 de octubre de 2023. [En línea]. Disponible: https://docs.aws.amazon.com/es_es/machine-learning/latest/dg/training-parameters.html

[2] K. Moriwaki. "Understanding Self-Organizing Map Neural Network with Python Code". Medium. Accedido el 20 de octubre de 2023. [En línea]. Disponible: https://towardsdatascience.com/understanding-self-organising-map-neural-network-with-python-code-7a77f501e985

[3] "Algoritmo de distancia euclidiana - Algoritmos de Grafos". GraphEverywhere. Accedido el 20 de octubre de 2023. [En línea]. Disponible: https://www.grapheverywhere.com/algoritmo-de-distancia-euclidiana/

[4] Vettigli, G. (2018). MiniSom: minimalistic and NumPy-based implementation of the Self Organizing Map. *URL: https://github.com/JustGlowing/minisom*.

[5] Saha, C., Baruah, N., & Nayak, S. K. (2021, July). Implementation of self-organizing map and logistic regression in dissolved gas analysis of transformer oils. In *2021 IEEE International Conference on the Properties and Applications of Dielectric Materials (ICPADM)* (pp. 131-134). IEEE.

[6] Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, *78*(9), 1464-1480.

[7] "IBM Documentation". IBM in Deutschland, Österreich und der Schweiz | IBM. Accedido el 20 de octubre de 2023. [En línea]. Disponible: https://www.ibm.com/docs/es/spss-modeler/saas?topic=SS3RA7_sub/modeler_mainhelp_client_ddita/clementine/trainnetnode_learning_rates.htm

[8] "Papers with Code - Wine Dataset". The latest in Machine Learning | Papers With Code. Accedido el 20 de octubre de 2023. [En línea]. Disponible: https://paperswithcode.com/dataset/wine