

The cover features a decorative graphic consisting of three blue circles of varying sizes, each with a lighter blue ring around its center. These circles are arranged along a diagonal line that runs from the top-left towards the bottom-right. The background is white with thin blue lines extending from the top-left and bottom-right corners towards the center.

Aplicación de mapas autoorganizados (SOM) a la visualización de datos

Modelos Computacionales
Fernando José Serrano García

Contenido

Introducción	3
Estructura	3
Entrenamiento	3
Aplicación de SOM a la visualización de datos.....	4
Desarrollo de SOMVis	5
Introducción	5
Datos de entrada.....	5
Flujo de la aplicación	6
Opciones de visualización	6
Visualización de los pesos de las neuronas.....	6
Visualización de los valores de entrada	7
Visualización de la matriz de distancias	7
Visualización de los pesos sinápticos y entradas	7
Visualización de la matriz de distancias y entradas	7
Manual de uso del programa	8
Conjuntos de datos de ejemplo	10
Datos Iris (iris.csv)	10
Desordenes del hígado (liver-disorders.csv)	10
Mamografías (mamography.csv)	11
Hepatitis (hepatitis.csv).....	11
Vinos (wines.csv)	12
Conclusiones y mejoras futuras	12

Introducción

Los **mapas autoorganizados** o **SOM (Self-Organizing Maps)**, también llamados **redes de Kohonen** son un tipo de red neuronal no supervisada, competitiva, distribuida de forma regular en una rejilla de, normalmente, dos dimensiones.

Su finalidad es descubrir la estructura subyacente de los datos introducidos en ella. A lo largo del entrenamiento de la red, los vectores de datos son introducidos en cada neurona y se comparan con el vector de peso característico de cada neurona.

La neurona que presenta menor diferencia entre su vector de peso y el vector de datos es la **neurona ganadora (o BMU)** y ella, y sus vecinas verán modificados sus vectores de pesos.

Este tipo de mapas permiten reducir la dimensionalidad de los vectores de entrada para representarlos mediante una **matriz de distancias unificada (U-matriz)** generalmente consistente en una matriz 2D, apta para la visualización como una imagen plana.

Estructura

- **Matriz de neuronas:** Las neuronas se distribuyen de forma regular en una *rejilla* de dos dimensiones, que pueden ser rectangulares o hexagonales, en las que cada neurona puede tener cuatro o seis vecinos respectivamente.
- **Espacio de entrada:** Los datos de entrada corresponden a un vector de N componentes por cada atributo que queramos comprar, siendo esta dimensión la misma del vector de pesos sinápticos asociado a cada una de las neuronas de la rejilla.
- **Espacio de salida:** Corresponde con la posición (2D) en el mapa de cada neurona.
- **Relación entre neuronas:** Entre todas las neuronas hay una relación de vecindad que es la clave para conformar el mapa durante la etapa de entrenamiento. Esta relación viene dada por una función.

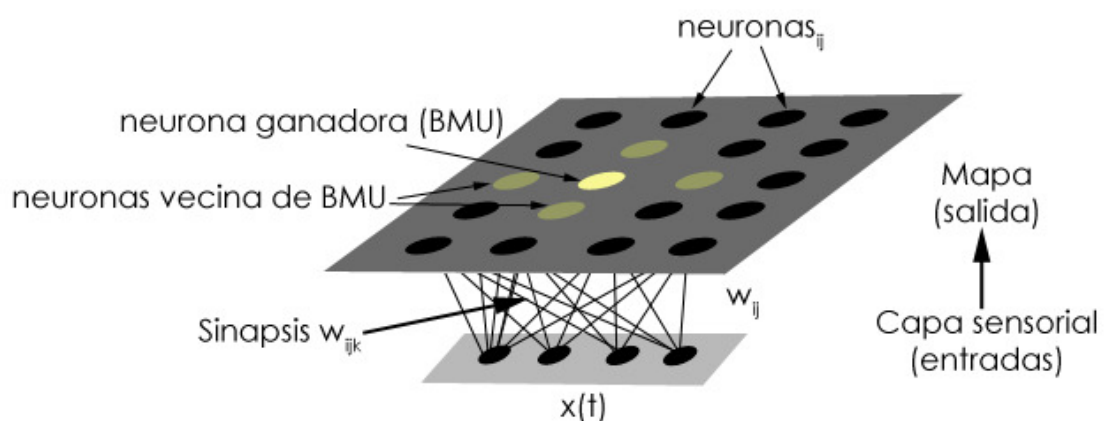


Ilustración 1. Estructura de un mapa autoorganizado

Entrenamiento

Para cada paso del entrenamiento (época) se introduce un vector de datos correspondiente a una entrada seleccionada aleatoriamente y se calcula la *similitud* entre este vector y el peso de cada neurona.

La neurona más parecida al vector de entrada será la neurona ganadora (Best-Matching Unit ó BMU). Generalmente se usa la distancia euclídea para medir esta similitud entre pesos sinápticos.

Tras esto, los vectores de pesos de la BMU y sus vecinos son actualizados de forma que se acercan al vector de entrada.

Aplicación de SOM a la visualización de datos

Las dos características descritas anteriormente hacen especialmente atractivo el uso de SOM para representación de datos:

- **Reducción de la multidimensionalidad:** Podemos representar conjuntos de datos de gran número de atributos en mapas 2D.
- **Asociación de elementos con atributos similares:** Visualmente podemos ver de forma rápida como quedan agrupados elementos que tienen valores próximos entre sí.

Aparte de estas razones a nivel técnico tenemos además muchas ventajas como son:

- **Facilidad de implementación:** La implementación de un sistema SOM es relativamente fácil y se adapta perfectamente al modelo de procesado en paralelo por lo que es también muy fácilmente optimizable.
- **Abstracción de los datos de entrada:** Los SOM son totalmente transparentes a la naturaleza de los datos de entrada, *tan solo* se limitan a comparar vectores de entrada con los pesos sinápticos de las neuronas de la rejilla. Esto hace que pueda ser usado en gran variedad de problemas sin tener necesidad de cambiar la aplicación base.
- **Facilidad de integración con otras técnicas:** Esta técnica se puede compaginar con otras técnicas de obtención de conocimiento como pueden ser las redes bayesianas realizando un preprocesado del conjunto de datos.

Recientemente se está produciendo un crecimiento del uso de mapas SOM gracias a su potencia y facilidad de uso para trabajar en campos como la estadística.

Además una vez obtenida el mapa, la representación no tiene porque limitarse a una rejilla rectangular, como se puede apreciar en la siguiente imagen.

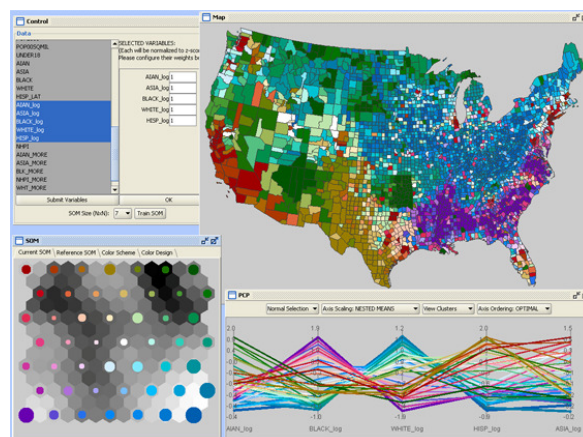


Ilustración 2. Ejemplo de uso de SOM en aplicaciones estadísticas

Desarrollo de SOMVis

Introducción

El objetivo de este trabajo ha sido desarrollar una aplicación para poder visualizar conjuntos de datos de N-Dimensiones usando un mapa SOM.

La aplicación lee un conjunto de datos de un fichero de entrada y muestra una animación correspondiente al proceso de aprendizaje de la red con esos datos.

El usuario puede modificar gran cantidad de parámetros que condicionan el comportamiento de la red.

La aplicación se ha desarrollado para Windows usando Visual C++ y el framework .net 2.0.

Datos de entrada

Los datos a visualizar se facilitarán a la aplicación en formato **CSV (Comma separated values)** con las siguientes características:

- Se usarán **punto y coma (;)** para separar los valores (columnas).
- La primera línea contendrá la **descripción de cada atributo** (columna) y las siguientes líneas corresponderán al valor de cada variable para una entrada concreta.
- Todos los **valores para los atributos deberán ser numéricos** pudiendo ser tanto enteros como flotantes. Si tuviéramos alguna columna, como la clase de un objeto, que fuera texto, deberemos convertirla a valores enteros.
- Los datos no hace falta introducirlos normalizados, ya que **la aplicación los normaliza** en base al menor y mayor de cada columna.

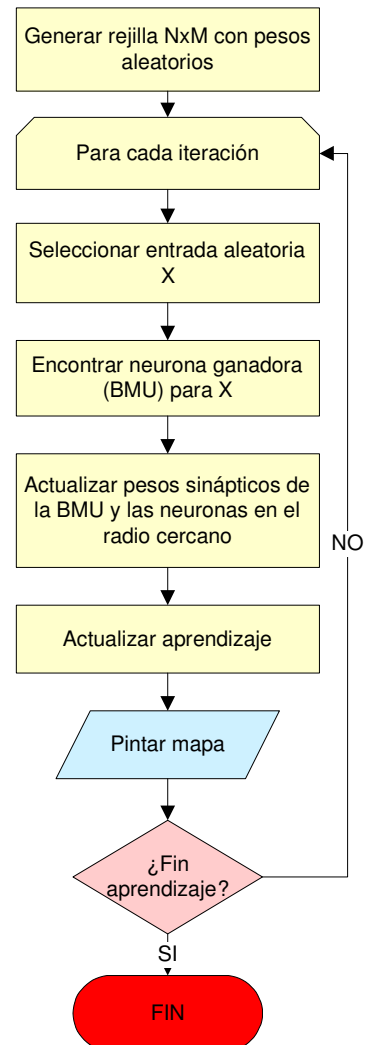
Un ejemplo de texto CSV representando los datos IRIS es (*Nótese como se ha cambiado la última columna para representar 0=setosa, 1=versicolor, 2=virginica*):

```
Sepal length;Sepal width;Petal length;Petal width;Specie
5.1;3.5;1.4;0.2;0
4.9;3.0;1.4;0.2;0
4.7;3.2;1.3;0.2;1
4.6;3.1;1.5;0.2;1
5.0;3.6;1.4;0.2;1
5.4;3.9;1.7;0.4;2
4.6;3.4;1.4;0.3;2
...
```

Flujo de la aplicación

El flujo principal de la aplicación viene definido en el siguiente diagrama.

1. Inicialmente se genera una rejilla de $N \times M$ usando pesos aleatorios para cada neurona ij .
2. Entramos en un bucle según el número de iteraciones indicado por el usuario:
 - a. Seleccionamos una entrada al azar de todo el conjunto de entrada.
 - b. Buscamos una neurona ganadora (BMU) buscando la menor distancia con todas las neuronas de la rejilla. Para ello usamos la **distancia L_p** .
 - c. Actualizamos los pesos sinápticos tanto de la neurona ganadora como de las neuronas cercanas en base al radio definido y a un valor de influencia de vecindad.
 - d. Se actualiza el radio de aprendizaje de toda la red.
 - e. Pintamos el mapa SOM a un bitmap.
3. FIN



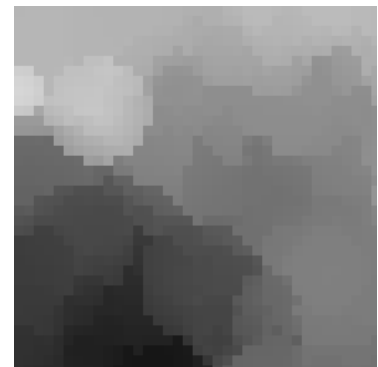
Opciones de visualización

La aplicación soporta tres modos básicos de visualización que además se pueden combinar entre ellos:

Visualización de los pesos de las neuronas

Este modo de visualización permite ver los pesos sinápticos de cada neurona de la rejilla.

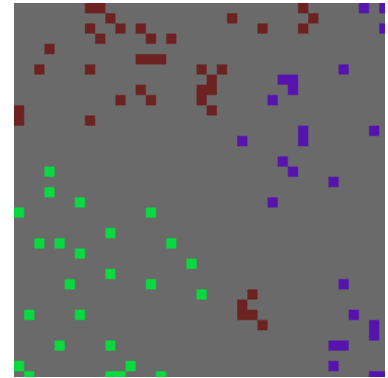
De todas las variables de entrada la aplicación permite seleccionar los atributos que nos interesa representar. El valor de cada neurona viene dado por la media de los valores de los atributos seleccionados dando lugar a un mapa de escala de grises.



Visualización de los valores de entrada

En este modo se representa la posición y la clase de cada uno de los valores de entrada.

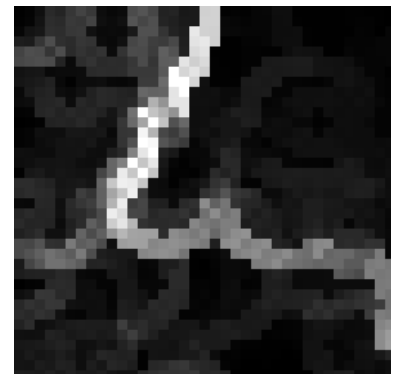
El color de cada clase viene determinado de forma aleatoria al comienzo del entrenamiento.



Visualización de la matriz de distancias

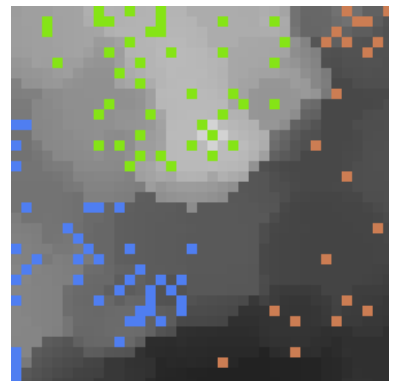
Para obtener esta representación se obtiene el valor de error para cada neurona usando la diferencia entre los valores de las cuatro neuronas vecinas.

En esta gráfica podemos ver como mientras más oscuras sean las zonas, más parecidas serán. Del mismo modo, mientras más claras sean significa que hay mayor diferencia de valores por lo que muestran una *frontera* entre neuronas que representan diferentes conjuntos de datos.



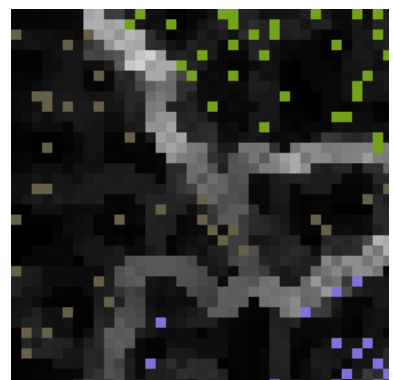
Visualización de los pesos sinápticos y entradas

Esta representación nos muestra al mismo tiempo los pesos de las neuronas así como la posición de las entradas.



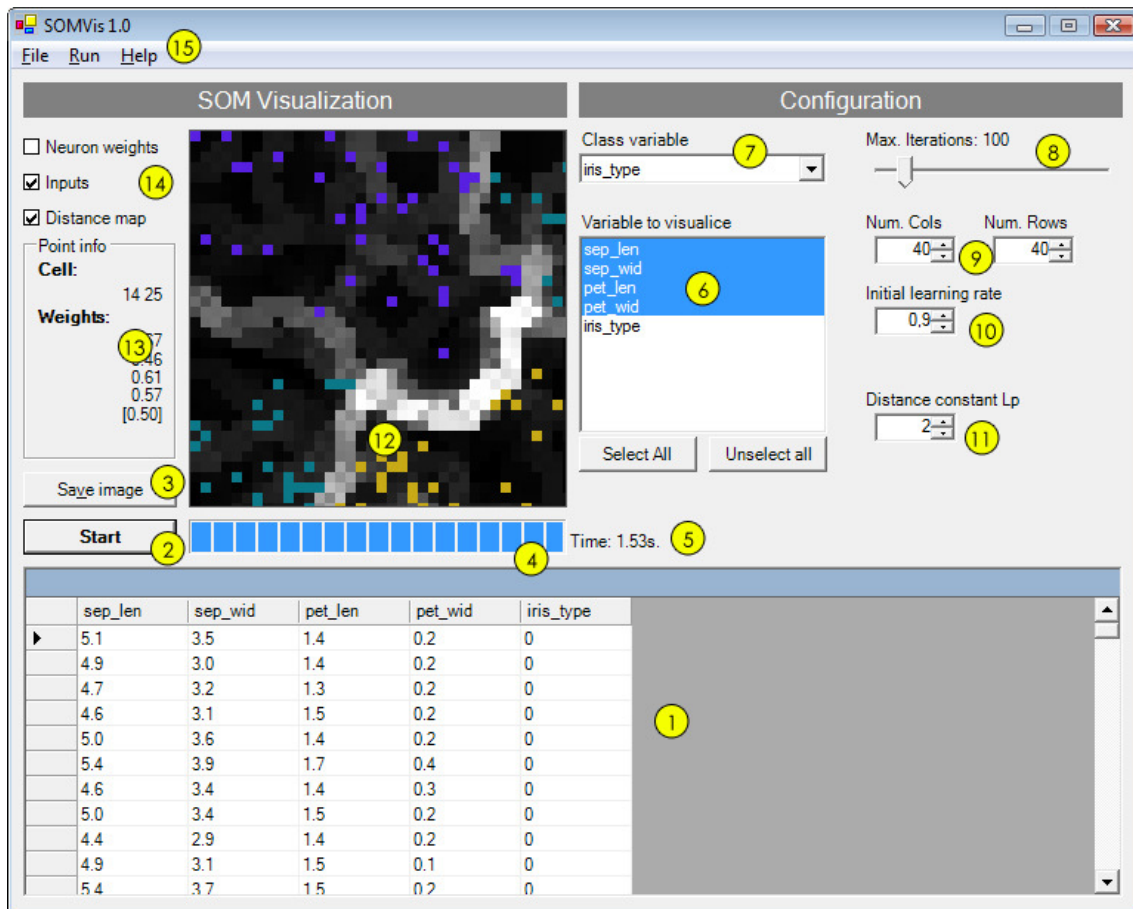
Visualización de la matriz de distancias y entradas

Muestra el mapa de distancias superponiendo la posición de las entradas de la red.



Manual de uso del programa

A continuación se muestra la interfaz del programa desarrollado así como la explicación de todas las opciones y controles disponibles:



- 1. Tabla de valores de entrada:** Aquí aparecen los valores de entrada sin normalizar tal cual se leen desde el fichero CSV.
- 2. Botón "Start":** Comienza la ejecución del aprendizaje. Una vez comenzada cambiará su nombre por **Stop** y podremos parar la ejecución.
- 3. Botón "Save Image":** Guarda la imagen actual en formato PNG.
- 4. Barra de progreso:** Muestra el progreso de aprendizaje hasta llegar al número máximo de iteraciones definido.
- 5. Tiempo de ejecución:** Muestra el tiempo empleado en la ejecución del aprendizaje.
- 6. Lista de variables a visualizar:** Cuando realizamos una visualización de los pesos de las neuronas, el color de cada neurona está determinado por la media aritmética de los valores de los atributos que tengamos seleccionados en esta lista.
- 7. Variable de clase:** Es la variable que usaremos para determinar la clase (por defecto la última columna). Y se usará para representar los colores de los valores de entrada.
- 8. Número máximo de iteraciones:** Número de iteraciones (épocas) que queremos usar para entrenar la red.
- 9. Número de filas y columnas:** Dimensión de la red de neuronas.

- 10. Valor de aprendizaje inicial:** Valor usado para comenzar el aprendizaje, que se usará en cada iteración por medio de la siguiente fórmula:

$$\text{valorAprendizaje} = \text{valorInicialAprendizaje} \cdot e^{-\frac{\text{iteración}}{\text{númeroIteraciones}}}$$

- 11. Constante Lp:** Valor de la constante Lp para el cálculo de la distancia entre neuronas, usando la siguiente fórmula para comparar los pesos de la neurona y con la entrada x:

$$\sqrt[p]{\sum_{i \in \text{Atributos}} |x(i) - y(i)|^p}$$

- 12. Mapa SOM:** Representación bidimensional del mapa.
- 13. Información de la celda actual:** Muestra el valor de cada peso sináptico de la neurona sobre la que movemos el ratón. La variable entre corchetes corresponde con el valor de la clase de esa neurona.
- 14. Modo de representación.** Seleccionamos los tres modos de representación descritos anteriormente.
- 15. Menú principal:**
- a. **File:**
 - i. **Open CSV (Ctrl+O):** Seleccionamos un fichero CSV de datos para abrir.
 - ii. **Reopen CSV (Ctrl+R):** Cargamos de nuevo el último fichero CSV abierto sin necesidad de volver a seleccionarlo.
 - iii. **Export to Image:** Guarda el fichero actual con formato PNG.
 - iv. **Exit (Ctrl+X):** Se cierra la aplicación.
 - b. **Run:**
 - i. **Start (Ctrl+S):** Inicia la ejecución del aprendizaje.
 - c. **Help:**
 - i. **About:** Muestra una pequeña información sobre la aplicación.

Conjuntos de datos de ejemplo

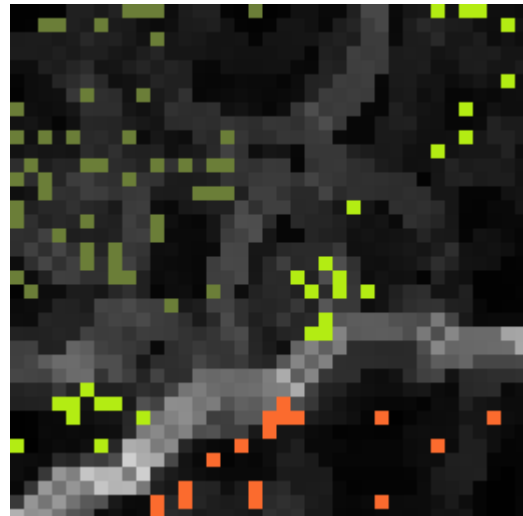
Los conjuntos de datos usados a continuación se obtuvieron de la página web de UCI Machine Learning Repository (<http://archive.ics.uci.edu/ml/>), previo procesamiento de los mismos que consistió en:

- Eliminar los valores desconocidos, representados por *.
- Cambio de valores de variables de texto por enteros.
- Cambio del formato de separación de los valores a punto y coma (;).
- Introducción de la cabecera con la descripción de la variable.

Datos Iris (iris.csv)

El famoso conjunto de datos multivariable creado por Ronald Aylmer Fisher (1936). Recoge cuatro propiedades características de tres variedades de lirios: setosa, versicolor y virginica.

Tras realizar el aprendizaje se muestra observa como la setosa (en la parte inferior) se diferencia más de las otras dos variedades al estar separadas por una frontera de color más claro. Aún así se puede apreciar como las otras dos variedades tienden igualmente a agruparse entre su especie.



Distribución de clases:

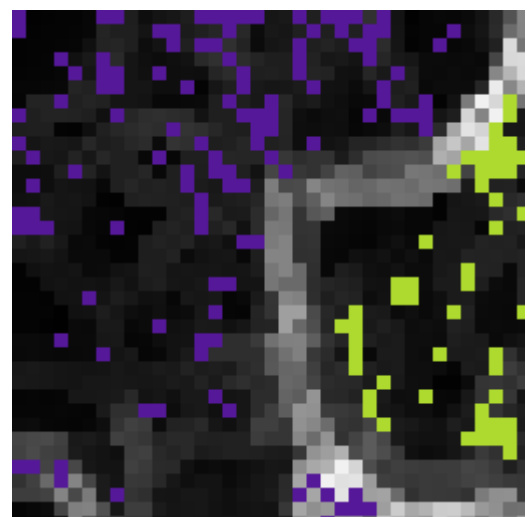
- Setosa: 50
- Versicolor: 50
- Virginica: 50
- Total: **150**

Número de atributos: 4

Desordenes del hígado (liver-disorders.csv)

Conjunto de 7 variables correspondientes principalmente a análisis de sangre en pacientes bebedores divididos en dos grupos: con problemas de hígado y sin problemas.

En la gráfica se puede apreciar claramente como los datos se agrupan por clase y además hay una gran diferencia entre una clase y otra, marcada por el desnivel de color gris claro que separa los pacientes enfermos de los sanos.



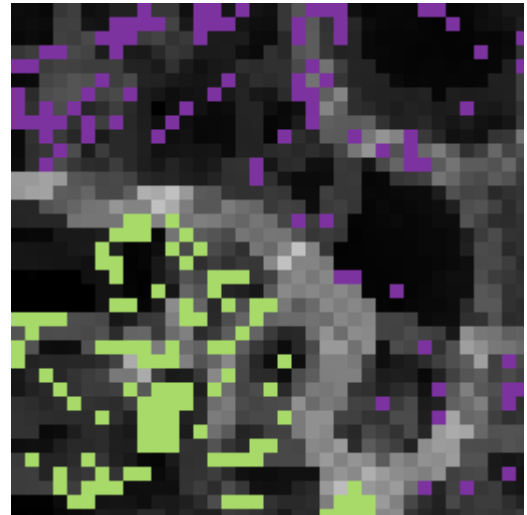
Distribución de clases:

- Con problemas: 146
- Sin problemas: 199
- Total: **345**

Número de atributos: 7**Mamografías (mamography.csv)**

Datos referentes a pacientes que se han sometido a una mamografía y se les ha diagnosticado una masa benigna o maligna.

El mapa resultante muestra una diferencia de grupos marcada por una frontera, pero igualmente se puede apreciar como en la esquina superior derecha también aparece una leve frontera indicando que estos datos difieren de los más cercanos, pudiendo dar lugar a falsos positivos.

**Distribución de clases:**

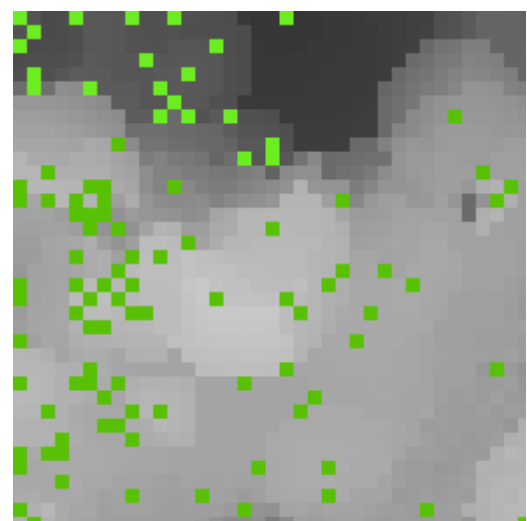
- Benigna: 516
- Maligna: 445
- Total: **961**

Número de atributos: 6**Hepatitis (hepatitis.csv)**

Corresponden con 20 datos sobre pacientes con hepatitis en los que se clasifican en dos grupos: pacientes que murieron y pacientes que están vivos

Distribución de clases:

- Muerto: 33
- Vivo: 122
- Total: **155**

Número de atributos: 19

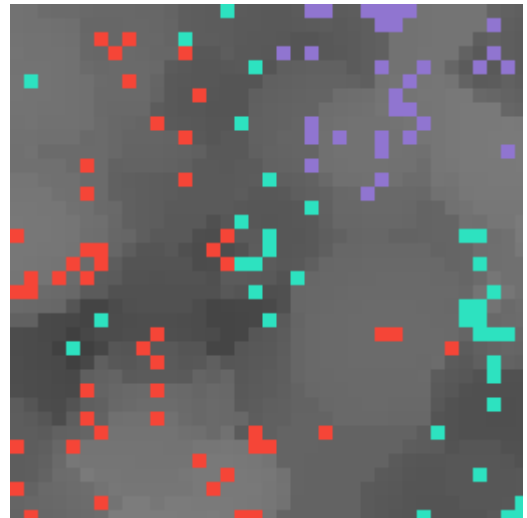
Vinos (wines.csv)

Análisis químicos de vinos cultivados en la misma región de Italia pero en diferentes viñas. Las 13 variables identifican los tres tipos de vino diferentes analizados.

Distribución de clases:

- Tipo 1: 59
- Tipo 2: 73
- Tipo 3: 46
- Total: **178**

Número de atributos: 13



Conclusiones y mejoras futuras

Ha sido una gran experiencia trabajar con este tipo de redes neuronales principalmente por la *vistosidad* de los resultados obtenidos. En ellos se puede ver rápidamente la organización de conjuntos de datos sobre los que no tenemos, a priori, ninguna idea de sus relaciones.

Me ha parecido una herramienta muy potente para el análisis de datos y muy versátil debido a la facilidad de integración con herramientas ya diseñadas, y que facilita de una forma muy rápida y limpia, una primera clasificación de conjuntos de datos para poder alimentar posteriormente sistemas de *business intelligence* más complejos si se desea.

El esfuerzo de realizar la herramienta lo más abierta posible y sin restringir el número o valores de los datos de entrada me ha permitido poder comprobar varios conjuntos de datos y verificar así la implementación realizada.

Algunas ideas de mejoras futuras podrían ser:

- Permitir guardar video de la animación del aprendizaje.
- Permitir editar la tabla y refrescar la red con los datos introducidos.
- Poder exportar a formato CSV los datos tras el aprendizaje.
- Sin usar la clase inicial estimar la clase a la que pertenece una entrada y comprobar el error cometido.