Universidad de Burgos

Computación Neuronal y Evolutiva

P2: Self-organizing Maps

Diseñar y entrenar un mapa auto-organizado con una rejilla bidimensional para el agrupamiento de un único conjunto de datos de muestra de tres dimensiones. Estudiando el efecto que tienen una serie de parámetros durante el entrenamiento de este.

Estudiantes:

David Miguel Lozano
Javier Martínez Riberas

Profesor de la asignatura:

Álvaro Herrero Cosío

1º semestre 2016

Índice

A. Introduction

El objetivo de la práctica es diseñar y entrenar un mapa auto-organizado para una rejilla bidimensional con un único conjunto de datos de muestra de tres dimensiones.

Se realizará un estudio sobre el impacto que supone variar los siguientes parámetros en el entrenamiento del mapa para el mismo conjunto de datos:

1. Dimensiones de la topología.
2. Tamaño del vecindario.
3. Función de topología.
4. Función de distancia entre neuronas.

B. Descripción del conjunto de datos

El conjunto de datos utilizado para esta práctica ha sido generado aleatoriamente mediante la función `nngen`. Se ha intentado conseguir un conjunto en donde se diferencien 7 clústeres próximos entre ellos pero que sean linealmente separables.

Los datos han sido generados de la siguiente manera:

```
nngenc([0 5; 0 5; 0 5], 7, 200, 0.30);
```

El primer parámetro (`bounds`) se corresponde con los rangos del espacio en donde generar los datos. Nuestros datos están en un espacio de tres dimensiones en el rango [0, 5] en cada una.

El segundo parámetro (`clusters`) es el número de clústeres a generar. Generamos 7 clústeres linealmente diferenciables.

El tercer parámetro (`points`) es el número de puntos por clúster. Nuestros clústeres tienen 200 puntos cada uno.

Por último se indica la desviación estándar de los clústeres (`std_dev`). Asignamos una desviación de un 30 % para que los clústeres no estén muy localizados y su agrupamiento no sea tan trivial.

Una vez generados los datos deseados, los hemos guardado en un fichero CSV para su posterior explotación en el estudio. En la siguiente figura podemos ver una representación de estos:

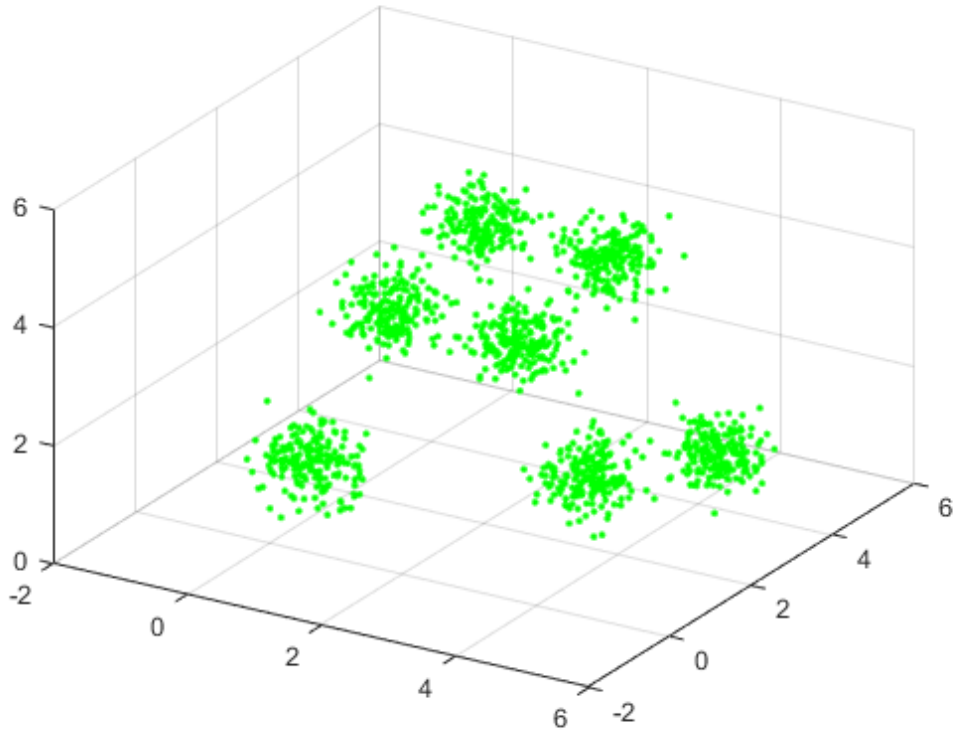

Figura 1: Representación de los datos.

C. Descripción del procedimiento

Para automatizar el estudio lo máximo posible, se ha realizado un script que entrena un mapa auto-organizado variando los parámetros de dimensiones de la topología, tamaño del vecindario, función de topología y función de distancia entre neuronas.

Por cada configuración, se generan las siguientes figuras y se guardan como imágenes `png`:

- `plotsomnd`: en esta representación se puede observar la topología del SOM (*self-organizing map*), las conexiones directas entre neuronas y la distancia entre estas. Las neuronas se dibujan como hexágonos azules, las conexiones mediante líneas rojas y las distancias se representan mediante el coloreado de las parcelas, correspondiendo el color amarillo a distancias cortas y el negro a distancias grandes. [2]

- `plotsomhits`: en esta representación se visualiza el número de datos para los que se está activando cada neurona (el número de vectores de entrada que clasifica). Se dibuja la topología del SOM y en cada parcela, correspondiente a una neurona, se muestra el número total de datos clasificados, así como un hexágono azul de tamaño relativo a este. [1]
- `plotsompos`: muestra un gráfico en 2D en donde se visualizan las posiciones de las neuronas (puntos azules), las conexiones entre estas (líneas rojas) y las posiciones de los datos (puntos verdes). Permite visualizar cómo el SOM clasifica el espacio de entrada según evoluciona el entrenamiento. [3]

Los comandos `plotsomnd` y `plotsomhits` están limitados a topologías `hextop` o `gridtop`, no pudiendo ser utilizados con `randtop`.

**Todo el código generado en esta práctica se encuentra disponible en el siguiente repositorio: P2_SelfOrganizingMap.*

D. Estudio

Antes de comenzar el estudio, se jugó un poco con los datos para tantear qué valores podían merecer la pena estudiar. Finalmente elegimos los siguientes parámetros:

- Función de distancia entre neuronas: [4]
  1. `dist`: distancia Euclídea.
  2. `linkdist`: distancia en número de enlaces.
  3. `mandist`: distancia de Manhattan.
  4. `boxdist`: distancia cuadrada.
- Función de topología: [5]
  1. `gridtop`: rejilla cuadrada.
  2. `hextop`: rejilla hexagonal.
  3. `randtop`: rejilla aleatoria.

1. Dimensiones de la topología: de 7 a 10 con incrementos de 1.
2. Tamaño del vecindario: de 2 a 7 con incrementos de 1.

D.1. Criterio

De todas las combinaciones anteriores (288 casos) se seleccionaron manualmente los mejores representantes de cada función de topología y función de distancia, quedándonos con 12 casos a analizar.

La principal limitación que hemos tenido en esta práctica ha sido el no tener un parámetro que nos indicase cómo de bueno o malo era cada caso. Por ello, la clasificación se ha realizado de forma manual (y por tanto con cierta componente de subjetividad).

Para el análisis, nos hemos basado sobretodo en las representaciones `plotsomnd` y `plotsomhits`. Ya que, nos proporcionan información de calidad sobre cómo se han distribuido las neuronas sobre los datos. En cambio, `plotsompos`, al realizar una representación en dos dimensiones de datos tridimensionales, puede llevar a interpretaciones erroneas, aunque en combinación con los anteriores también es de utilidad.

A continuación se exponen dos ejemplos de un mejor y un peor caso según el criterio seguido.

Cómo vemos en el 'caso malo', no existe una diferenciación clara entre los siete clústeres. Tanto en distancias entre neuronas como en datos captados por cada una. Además, existen neuronas muertas (sin datos o muy pocos) que no favorecen la distinción de ninguna clase.

Por el contrario, en el 'caso bueno' podemos observar una clara diferenciación de los siete clústeres. Vemos como la distancia entre las neuronas fronterizas entre clases es alta, mientras que la distancia entre neuronas de la misma clase es corta. Además, las neuronas muertas se encuentran en fronteras, lo cual favorece la clasificación.

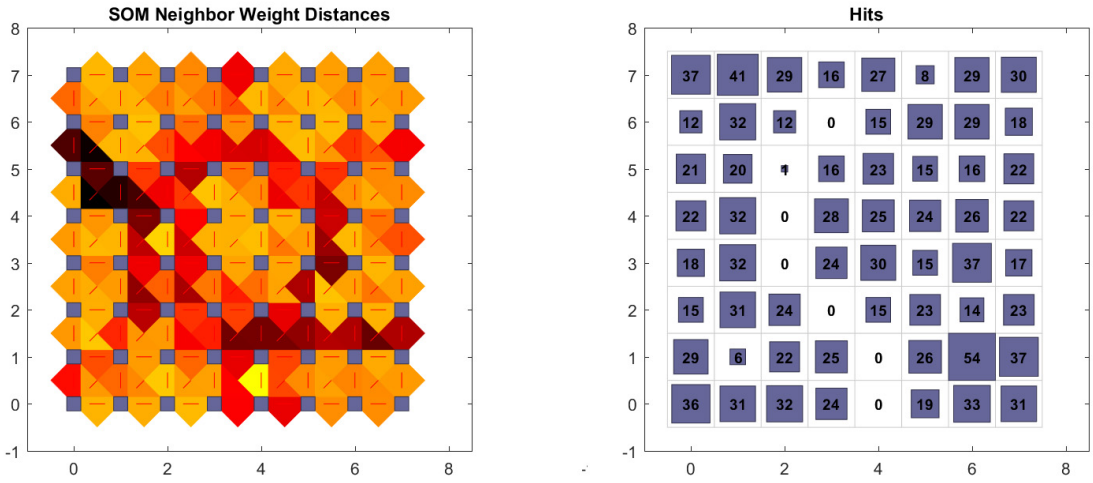

Figura 2: Caso malo.

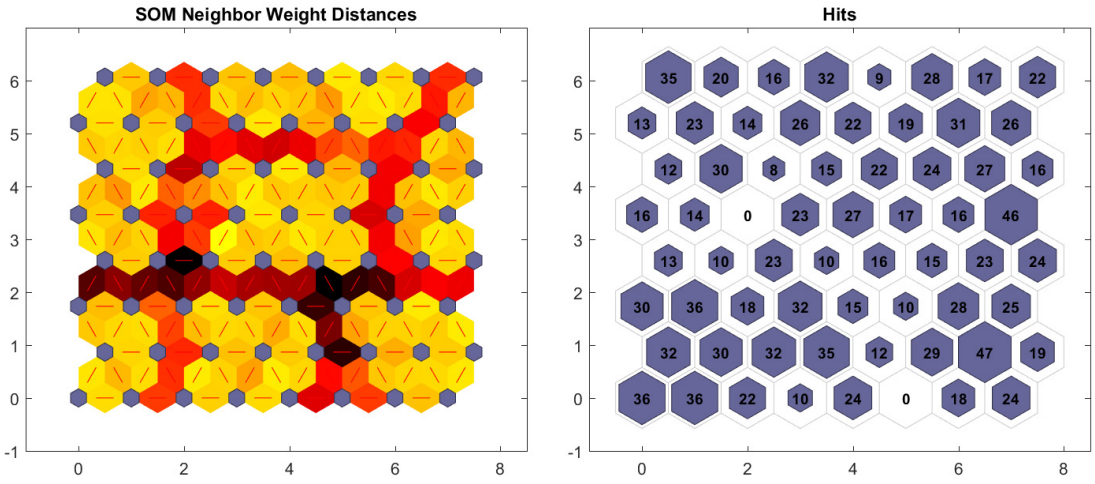

Figura 3: Caso bueno.

D.2. Resultados

Las imágenes de los resultados de las 288 ejecuciones con las diferentes configuraciones se encuentran ordenadas en el repositorio de la práctica: Resultados. No se incluyen en el informe por no aumentar demasiado la longitud de este.

En la siguiente tabla se resumen parte de nuestras observaciones de los resultados, calificando cada caso como 'malo', 'medio' o 'bueno'.

Rejilla	Distancia	Resultado
gridtop	dist	medio
	linkdist	bueno
	mandist	medio
	boxdist	malo
hextop	dist	bueno
	linkdist	medio
	mandist	error
	boxdist	medio
randtop	dist	medio
	linkdist	malo
	mandist	medio
	boxdist	malo

Cuadro 1: Resumen de los resultados

D.3. Resultados reseñables

Las conclusiones de los resultados de las ejecuciones con los diferentes parámetros las comentamos en la siguiente sección. En esta, queremos resaltar alguno de los resultados que más nos ha llamado la atención.

En primer lugar, comentar el caso de las neuronas aisladas que nos hemos encontrado en varias ejecuciones con la topología `randtop`.

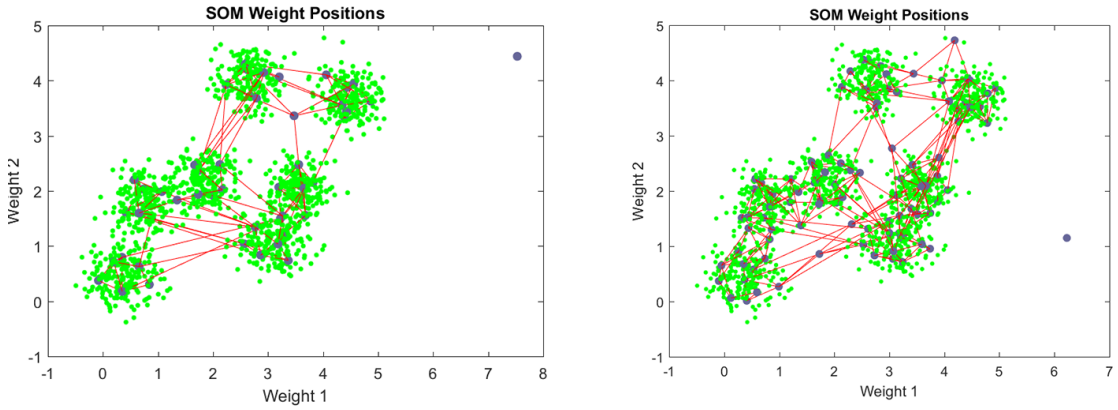

Figura 4: Neuronas aisladas.

En segundo lugar, queremos mostrar el error que obtuvimos al intentar representar una topología `randtop` con las funciones `plotsomnd` y `plotsomhits`. Como ya hemos comentado anteriormente, estos comandos solo pueden ser utilizados con topologías `hextop` o `gridtop`.

```
Only HEXTOP and GRIDTOP topology functions supported.
```

Figura 5: Error al intentar representar `randtop`.

Y por último, comentar los casos peculiares que obtuvimos con topología `hextop` y la función de distancia `mandist`. Vemos como todas las neuronas están unidas como si de una cadena se tratase.

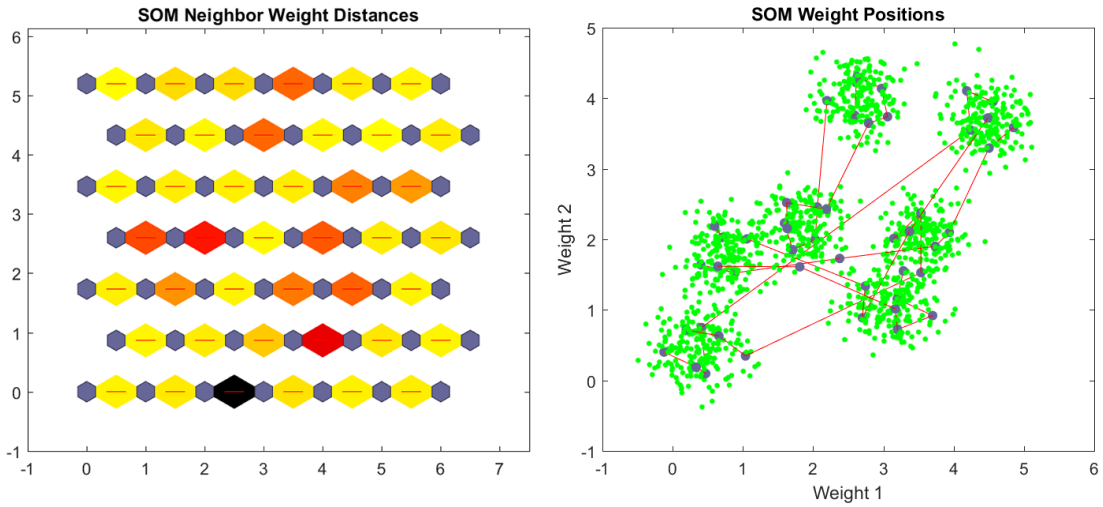

Figura 6: Topología `hextop` con `mandist`.

E. Conclusiones

Reiterar que el análisis ha sido realizado de forma manual con una cierta componente de subjetividad. No se poseía ningún parámetro (cómo en la práctica anterior) que nos permitiese clasificar los casos objetivamente y de forma automática.

A continuación exponemos nuestras conclusiones sobre cada parámetro.

- **Dimensiones de la topología:** según aumentamos el número de neuronas baja, por lo general, la distancia entre estas. Aparecen más neuronas muertas en las regiones fronterizas distantes. Y cada neurona se especializa en un conjunto menor de datos. Pero a la vez, aparecen algunos problemas en la clasificación de clases con fronteras cercanas.
- **Tamaño del vecindario:** vemos que es recomendable ampliar el tamaño del vecindario según se amplía las dimensiones de la topología. Pero no hemos encontrado ninguna relación directa del efecto de ampliar el tamaño del vecindario con una dimensión de la topología fija.
- **Función de topología:**
  - `gridtop`: ha proporcionado unos resultados bastante constantes, obteniendo sus mejores resultados con la función de distancia `linkdist`. En líneas generales, han sido algo peores que con `hextop`.
  - `hextop`: en general nos ha proporcionado buenos resultados. Sin embargo, utilizando la distancia de Manhattan obtenemos resultados peculiares. Las neuronas se distribuyen como si se tratase de una topología

lineal.

  - `randtop`: en primer lugar, comentar que sólo se ha podido representar con `plotsompos`, lo cual ha dificultado su análisis. En líneas generales, ha sido la función que peores resultados nos ha proporcionado. Hemos observado mucha variabilidad entre ejecuciones. En algunos casos el ajuste era correcto, pero en otros obteníamos varias neuronas aisladas en zonas aleatorias. Creemos que si el número de vecinos fuese mayor de 0 se podrían obtener mejores resultados.

- **Función de distancia entre neuronas:** los resultados son muy dependientes de la topología. Aunque observamos resultados algo mejores con distancias Euclídeas y distancias de número de enlaces.

Referencias

[1] Matlab documentation. plotsomhits command, 2016. URL http://www.mathworks.com/help/nnet/ref/plotsomhits.html. [Online; accessed 08-October-2016].

[2] Matlab documentation. plotsomnd command, 2016. URL http://www.mathworks.com/help/nnet/ref/plotsomnd.html. [Online; accessed 08-October-2016].

[3] Matlab documentation. plotsompos command, 2016. URL http://www.mathworks.com/help/nnet/ref/plotsompos.html. [Online; accessed 08-October-2016].

[4] MathWorks. Distance funct. (dist, linkdist, mandist, boxdist), 2005. URL http://matlab.izmiran.ru/help/toolbox/nnet/selfor12.html. [Online; accessed 10-October-2016].

[5] MathWorks. Topologies (gridtop, hextop, randtop), 2005. URL http://matlab.izmiran.ru/help/toolbox/nnet/selfor11.html. [Online; accessed 10-October-2016].