PRÁCTICA 4

NEUROCOMPUTACIÓN

José Antonio Guerrero Avilés
Nº Lista: 14
cany@correo.ugr.es

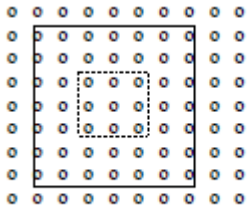
INDICE

1.- Descripción de los ficheros.

- **Datos_Microarrays_Cancer.xls**: Conjunto de datos dado por el profesor sin modificar en la hoja de cálculo.
- **DLCL_Supervivencia.txt**: Conjunto de individuos de los que vamos evaluar si sobreviven o no, estos datos están sin modificar.
- **procesarP4.m:** Programa en Matlab que lee los datos del fichero Datos_Microarrays_Cancer.xls y crea el fichero DLCL_Supervivencia.txt.
- **EntradaP4.txt:** Conjunto de datos directamente leídos de la hoja de cálculo sin procesar los elementos perdidos.
- **datosPerdidosP4.m:** Programa en Matlab que recupera los datos perdidos leídos de EntradaP4.txt mediante la técnica de los vecinos más cercanos.
- **Datos_Microarrays_Cancer.txt:** Conjunto de datos del problema con los datos perdidos ya estimados.
- **SalidaP4.txt**: Conjunto de las salidas que debería dar la red neuronal.
- **transformarValoresP4.m:** Programa en Matlab que elimina los genes que tienen menos picos de la cuenta.
- **resultado.txt:** Conjunto de datos que tienen más de 7 picos.
- **redKohonen.m:** Programa en Matlab que ejecuta la red de Kohonen.

2 Estudio teórico detallado de la red de Kohonen, que demuestre que se ha entendido su funcionamiento.

La red neuronal de Kohonen tiene una capa de entrada, una capa de salida y una estructura de vecindad, la cual suele ser unidimensional, lineal (dimensión 1) o bidimensional (dimensión 2). Aquí se muestran los tipos bidimensionales rectangular y la hexagonal:



Se entrenan con aprendizaje competitivo, de tal forma que en cada ejecución de la red, sólo una unidad de salida (o conjunto vecinal) se activará.

Los pesos de la unidad de salida se adaptarán de tal forma que se preserve el orden. Los datos que en el espacio de estados están cercanos (la cercanía vendrá dada por la distancia utilizada) se deben corresponder con unidades de salida que también estén cercanas.

Cada valor numérico representa un vector de pesos, a los cuales a veces se les llama neuronas.

Para cada entrada que tengamos se irán entrenando los pesos de la siguiente forma:

- Se coge el peso que más cerca esté de la entrada.
- A ese peso se le sumara el valor que tenía + un valor.
- Los pesos vecinos también se variaran pero en menos proporción.
- Esto hay que realizarlo para todas las entradas.

A la hora de ver la salida, para ver que entradas pertenecen a cada clase se mira cada entrada con un vector de pesos concreto y al que más cercano este a esa clase pertenece.

Tamaño de la red:

Cuanto mayor número de unidades se utilicen, mejor se cubrirá el espacio de entrada, aunque ello también signifique mayor tiempo de cálculo, hemos de llegar a un acuerdo entre ambos parámetros. Suele rondar en unas 100 neuronas. (10 x 10)

Inicialización:

Cuando no se conoce de antemano la distribución estadística de la población de la que proceden los ejemplos de entrada, se recomienda 0 y 0.1 como valores para los pesos.

Numero de iteraciones:

El número de iteraciones se establece empíricamente en función del tiempo de CPU que se quiera dedicar o del grado de exactitud que se quiera alcanzar.
Kohonen afirma que el número de patrones de patrones que se deben presentar debe ser aprox. 500 veces el número de unidades de red.

Con esto, el número de épocas se calculará como:

Nº épocas = 500 *(M * L) / Tamaño conjunto de entrenamiento (inputs)

Donde (M * L) = tamaño de la malla de salida.

A la hora de aplicar una red de Kohonen a unos datos concretos, hay que tener en cuenta que si la vecindad es muy grande, la red nunca convergerá, y si es muy pequeña, no se llevará a cabo la auto-organización, ya que no habrá efectos globales, sino locales y la red no convergerá.
Para alcanzar la convergencia en el coeficiente de aprendizaje, la amplitud de la modificación de los pesos debe disminuir con el tiempo.

3 Resultados de la aplicación de la Red de Kohonen al problema considerado.

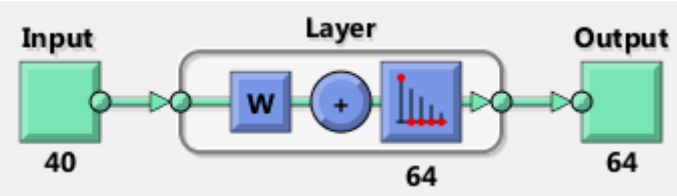
Para la red de Kohonen he usado los siguientes parámetros:

Num. clases posibles:	64 (8x8)
Épocas:	800

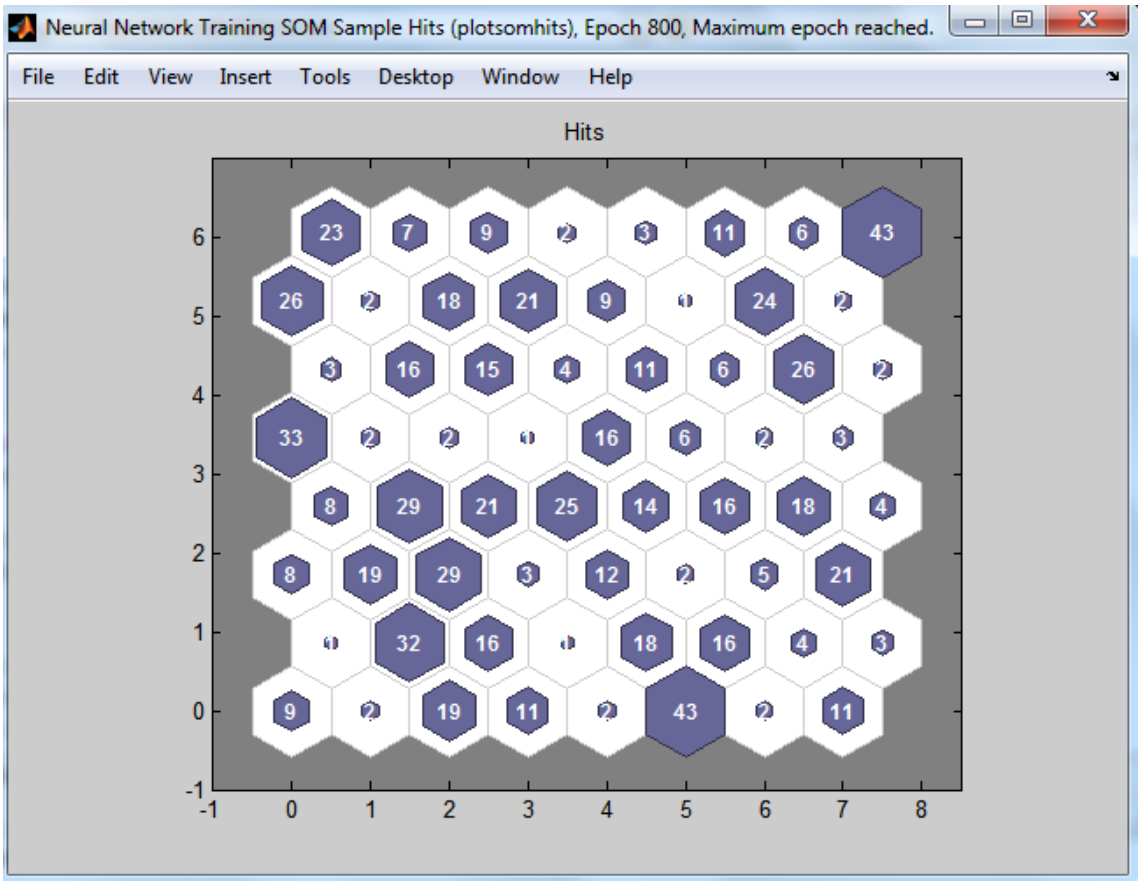
Las épocas se calculan como se ha dicho anteriormente,

Nº épocas = 500 *(M * L) / Tamaño conjunto de entrenamiento (inputs)

en este caso 500*64/40 = 800.



El número de clases importantes que salen son 31, como podemos ver en la siguiente imagen (Para mí un grupo importante es aquel que tiene más de 10 genes):



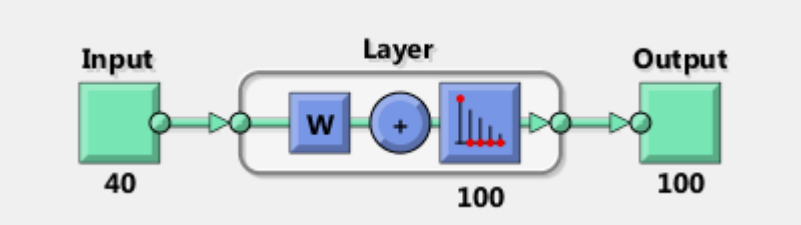
Ahora, para ver cómo cambia la red, voy a modificar los parámetros, aumentando y disminuyendo el número de clases posibles, lo que hará que se modifique también las épocas. Los parámetros que voy a usar son:

Num. clases posibles:	100 (10x10)
Épocas:	1250

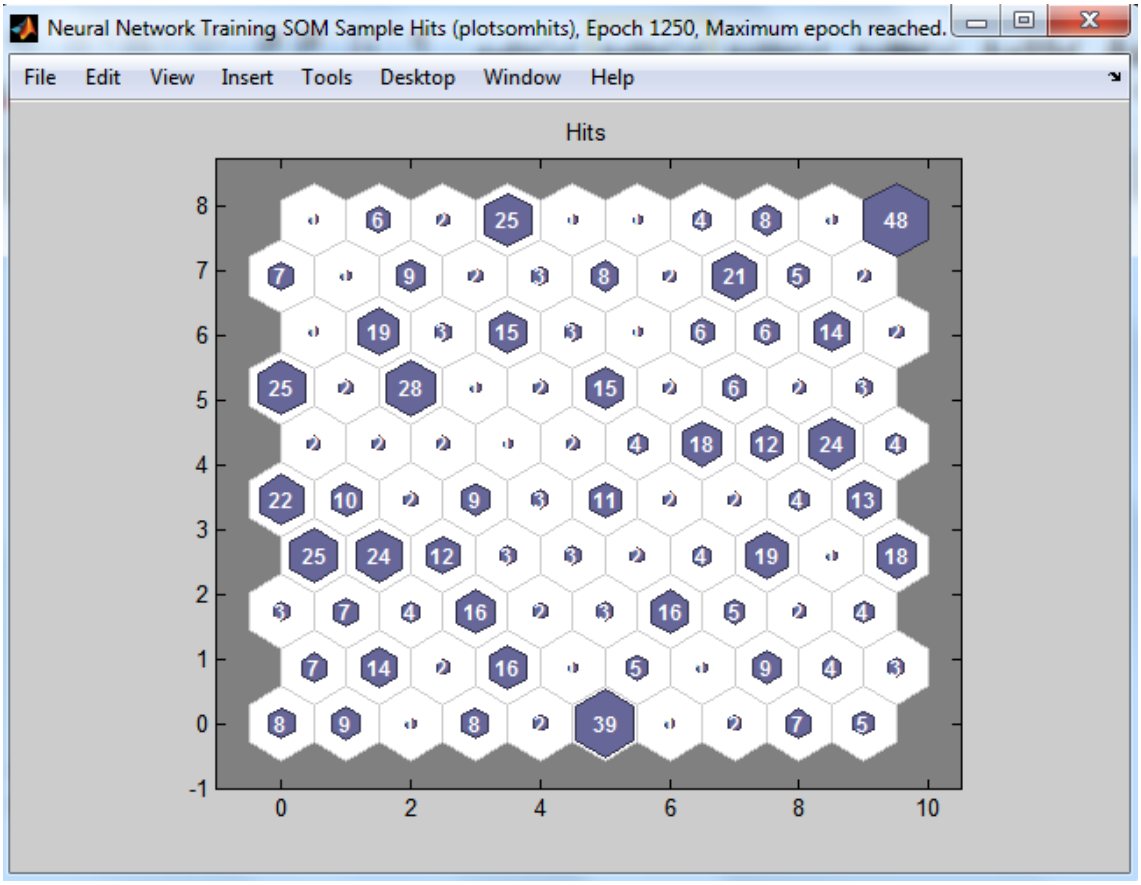
Las épocas se calculan como se ha dicho anteriormente,

Nº épocas = 500 *(M * L) / Tamaño conjunto de entrenamiento (inputs)

en este caso 500*100/40 = 1250.



El número de clases importantes que salen son 25, como podemos ver en la siguiente imagen (Para mí un grupo importante es aquel que tiene más de 10 genes):



Cabe destacar, que ha tardado bastante más que con las demás pruebas y que los datos, a partir de la época 600 aproximadamente, los resultados han ido variando poquísimo.

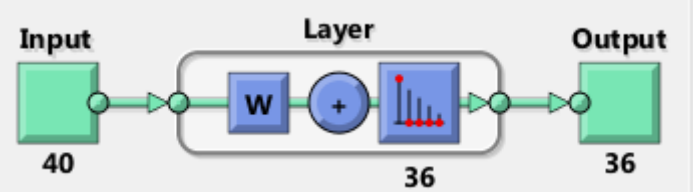
Como última prueba voy a usar:

Num. clases posibles:	36 (6x6)
Épocas:	450

Las épocas se calculan como se ha dicho anteriormente,

Nº épocas = 500 *(M * L) / Tamaño conjunto de entrenamiento (inputs)

en este caso 500*36/40 = 450.



El número de clases importantes que salen son 24, como podemos ver en la siguiente imagen (Para mí un grupo importante es aquel que tiene más de 10 genes):