PRÁCTICA 3

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 que se nos da, cambiando los puntos en los decimales por comas para su procesamiento.
- **DLCL_Supervivencia.txt:** Conjunto que se nos da de individuos a los cuales vamos a evaluar si sobreviven o no.
- **Procesar.m:** Programa en Matlab que procesa los dos ficheros anteriores y obtiene EntradaP3.txt y SalidaP3.txt.
- **SalidaP3.txt:** Conjunto de salidas que debería de dar la red neuronal.
- **EntradaP3.txt:** Conjunto de datos extraídos de la hoja de cálculo sin sustituir los elementos perdidos.
- **procesarEntradaP3.m:** Programa en Matlab que sustituye los datos perdidos del anterior fichero con la técnica de los vecinos más cercanos y genera entradaProcesada.txt.
- **entradaProcesada.txt:** Conjunto de datos del problema con los datos perdidos sustituidos.
- **faseA.m:** Programa en Matlab que ejecuta la fase A de la practica
- **faseBC.m:** Programa en Matlab que ejecuta las fase B y C de la practica
- **mejoresGenes.txt:** Fichero con el conjunto de los genes más representativos en el problema obtenidos en la fase C.
- **faseD.m:** Programa en Matlab que ejecuta la fase D de la práctica.
- **obtenerMejoresGenes.m:** Programa en Matlab que genera el archivo **datosMejoresGenes.txt** que contiene los genes con los que nos hemos quedado.
- **funcioncriterio.m:** Funcion que utilizamos como criterio para poder obtener los genes en SFS
- **SFS_mejoresGenes.m:** Programa en Matlab que te muestra por pantalla los mejores genes, los cuales tendremos que copiar a mano en el fichero de abajo.
- **mejoresGenesSFS.txt:** Los mejores Genes obtenidos en SFS, se usaran e SFS_faseD..
- **SFS_faseD:** Programa en Matlab que ejecuta la fase D para la clasificación libre (en mi caso SFS).

2.- Clasificación semi-guiada.

En esta sección voy a exponer los datos que he obtenido al realizar las diferentes pruebas que he considerado llevar a cabo.

2.1 Pruebas Fase A

No he podido realizar las pruebas pertinentes a la red de la fase A ya que mi ordenador no dispone de memoria suficiente para ejecutar esta red neuronal.

2.2 Pruebas Fase B

En esta fase vamos a tener 4 redes neuronales con 67 neuronas en la capa oculta, 8052 (4026*2) entradas y 1 sola salida.

Las pruebas que voy a realizar son las siguientes:
Iré probando diferentes configuraciones de la red como puede ser cambiando la función de entrenamiento, las épocas, el goal o el validation check. Función de activación por defecto es purelin.

Primera red

Entrenamiento: 1-30
Test: 31-40

Los datos con los que voy a empezar son:

Función entrenamiento:	trainrp
Épocas:	300
Goal:	0,2
Validación check:	6

Con los que obtengo:

INTENTO	errores
1	5
2	3
3	4
MEDIA	**4**

Voy a empezar a cambiar datos. Mencionar que de momento, no voy a tocar las épocas, ya que me he fijado en la ejecución y no llega a alcanzar ni la mitad de las que he usado.

Los datos con los que he realizado la siguiente prueba son:

Función entrenamiento:	trainrp
Épocas:	300
Goal:	0,1
Validation check:	12

Es decir, aumentamos el validation check y disminuimos el goal. He obtenido los resultados que se muestran en la siguiente tabla:

INTENTO	**errores**
1	7
2	4
3	7
MEDIA	**6**

Con los parámetros escogidos, se empeoran los resultados así que voy a probar con los siguientes datos:

Función entrenamiento:	trainrp
Épocas:	300
Goal:	0,01
Validation check:	6

Y los resultados que obtengo son:

INTENTO	**errores**
1	5
2	5
3	6
MEDIA	**5.33333**

Realizando más modificaciones en el goal y en el validation check no consigo mejorar los resultados (pueden verse todas las pruebas en el archivo MedicionesPruebas.xls).

Llegado a este punto, me he fijado en las ejecuciones y las épocas continúan sin alcanzarse, así que no tiene sentido tocarlas, por tanto, lo que he probado ha sido modificar la función de entrenamiento y la función de activación.

Los datos que he usado han sido:

Función entrenamiento:	trainscg
Épocas:	300
Goal:	0,2
Validation check:	6

Uso estos datos, porque han sido los que me han dado mejores resultados en las anteriores pruebas y lo único que hago es cambiar la función de entrenamiento. Los resultados que obtengo son:

INTENTO	**errores**
1	4
2	4
3	4
MEDIA	**4**

Como se ve, con los parámetros indicados anteriormente, mejoramos los resultados que teníamos hasta el momento, por tanto, esta será nuestra nueva configuración.

Por último, manteniendo esta configuración, voy a probar a modificar la función de activación de purelin a logsig:

Función entrenamiento:	trainscg
Épocas:	300
Goal:	0,2
Validation check:	6
Función de activación:	logsig

Y los resultados que obtengo son:

INTENTO	**errores**
1	4
2	4
3	4
MEDIA	**4**

Como los resultados son idénticos, mantengo la configuración anterior, así que para la red 1, los parámetros de configuración con los que obtengo mejores resultados y por tanto, con los que voy a configurar esa red son:

Función entrenamiento:	trainscg
Épocas:	300
Goal:	0,2
Validation check:	6

Segunda red

Entrenamiento: 11-40
Test: 1-10

Para esta y las dos siguientes redes, no voy a describir absolutamente todo el proceso de pruebas ya que creo, carece de sentido, puesto que la manera de proceder ha sido en esta segunda red, y en las restantes, la misma que con la red 1. He realizado una prueba con unos ciertos parámetros y en base a los resultados obtenidos, he ido comparándolos con los resultados de las siguientes pruebas.
Estas pruebas se pueden consultar en el archivo MedicionesPruebas.xls.
Mencionar que cuando aumento un valor, es porque la ejecución se sale por ese valor, por ejemplo, en el caso del goal. Esto se puede ver cuando se realiza la ejecución del programa y vemos cómo se va entrenando la red

Tras las pruebas realizadas, la mejor configuración para la segunda red es:

Función entrenamiento:	trainrp
Épocas:	300
Goal:	0,1
Validation check:	6
Función de activación:	purelin

Tercera red

Entrenamiento: 1-10 y 21-40
Test: 11-20

Tras las pruebas realizadas, la mejor configuración para la tercera red es:

Función entrenamiento:	trainrp
Épocas:	300
Goal:	0,1
Validation check:	12
Función de activación:	purelin

Cuarta red

Entrenamiento: 1-20 y 31-40
Test: 21-30

Tras las pruebas realizadas, la mejor configuración para la tercera red es:

Función entrenamiento:	trainrp
Épocas:	300
Goal:	0,1
Validation check:	12
Función de activación:	purelin

Una vez que he obtenido la mejor configuración para mis redes, voy a realizar 10 ejecuciones para obtener el % de aciertos medio. Estos resultados son:

EJECUCIÓN	%Aciertos
1	60
2	62,5
3	45
4	60
5	72,5
6	55
7	55
8	45
9	60
10	60
MEDIA	**57,5**

Como vemos, no es un resultado muy bueno, así que obtendré los 36 mejores genes e intentaré mejorar los resultados con ellos en las siguientes fases.

2.3 Pruebas Fase C: Datos de los mejores genes

Una vez que hemos ejecutado la fase B y hemos obtenido los mejores genes (fase c), ejecutando el archivo datosMejoresGenes.m se genera datosMejoresGenes.txt obtenemos el listado con los mejores genes y su valor.

41,15152
710,20335
1257,13199
1339,21339
2022,20695
2423,17246
2437,17708
2534,20238
2640,18390
2698,18262
2823,17256
2857,18373
3211,21350
3297,17293
3385,20431
3411,15434
3425,14388
3629,20355
3634,13538
3645,18538
3653,17547
3659,15862
3679,18401
3680,16550
3745,19368
3835,14272
3844,17177
3879,16916
3949,14738
3960,21218
3990,19320
3992,21111
3997,19457
4015,21091

2.4 Pruebas Fase D

Para configurar correctamente las 10 redes neuronales creadas, procedo de la misma manera que en la fase C o que en las anteriores prácticas: he comenzado con unos parámetros iniciales y, a partir de los datos obtenidos con esos parámetros, he ido comparándolos con nuevas pruebas; si mejoraban el resultado, actualizaba los parámetros, si no, continuaba con los que tenía hasta configurar así las 10 redes como habíamos hecho también en las anteriores prácticas. Los datos referentes a las pruebas se pueden consultar en el fichero MedicionesPruebas.xls

Con las redes configuradas, obtengo los siguientes porcentajes de acierto:

EJECUCION	%Aciertos
1	82,5
2	95
3	87,5
4	85
5	92,5
6	87,5
7	90
8	87,5
9	87,5
10	90
MEDIA	88,5

Que como se ve claramente, mejora muchísimo al resultado obtenido en la fase anterior (menos del 60% de acierto).

3 Clasificación libre

Para la clasificación libre he utilizado el método Sequential Feature Selection. Este es un método de selección de características que tiene dos componentes:

- Una función objetivo (criterio): el método intenta disminuir el número de fallos.
- Un algoritmo de búsqueda secuencial: añade y elimina características de un conjunto de candidatos. El algoritmo siempre se mueve a mejor, nunca a peor y cuando no puede mejorar, finaliza.

El método Sequential Feature Selection tiene dos variantes:

1. Sequential forward selection (SFS)
2. Sequential backward selection (SBS)

Yo he utilizado la primera de ellas (SFS). En esa variante lo que se hace es ir añadiendo características mientras no se disminuya el criterio, y vaya mejorando. En cuanto se deje de mejorar se termina el proceso y obtenemos cuáles son las columnas que tenemos que escoger para obtener un buen resultado.
El criterio se calcula con una función, que yo he añadido en el fichero "funcioncriterio.m". Esta función se le pasa a la función que usamos a la hora de obtener los genes: "sequentialfs", la cual devuelve los genes.

Este procedimiento es básicamente es un algoritmo greedy, el cual normalmente no da muy buenas soluciones ya que empieza muy agresivo pero luego se estanca encontrando no muy buenas soluciones, aunque en este caso si las encuentra y nos es válido. Este algoritmo se suele utilizar en programación para inicializar una búsqueda y así, en lugar de empezar desde un lugar aleatorio, se empieza desde una solución que seguro será mejor que la aleatoria.

El listado de mejores genes que obtengo en este punto del estudio es el siguiente: 134, 2229, 3140 y 3982.

Y usando estos genes con la red de la fase D, obtengo los siguientes porcentajes de aciertos:

EJECUCION	%Aciertos
1	90
2	95
3	97,5
4	92,5
5	90
6	92,5
7	90
8	95
9	92,5
10	97,5
MEDIA	93,25

Como se puede ver, nos da una buena mejora de resultados respecto a la fase D y obviamente, respecto a la fase B. Esto puede ser debido a que, aunque en la fase D se cogen más "mejores genes" puede existir alguna contradicción entre ellos o algo similar, por lo que, si aceptamos esta teoría, es lógico pensar que cuanto mayor sea el número de genes usados, es más probable que existan conflictos entre dichos genes, aunque obviamente, el resultado es más robusto ya que lo hemos estudiado en un abanico de posibilidades más amplio.