ANEXO I.- Clasificación mediante el uso de Redes Neuronales

I.1.- Introducción a las Redes Neuronales Artificiales (RNA)

I.1.1.- Definición de las RNA

Las RNA son sistemas adaptables, con procesamiento de datos distribuido y en paralelo, que expuestos a un entorno de información desarrollan representaciones internas para generar respuestas coherentes ante tales estímulos, y que imitan el funcionamiento fisiológico y psicológico del cerebro para resolver ciertos problemas. Estos métodos no necesitan el desarrollo de algoritmos específicos para cada problema.

Son adecuadas para:

- Problemas con grandes cantidades de datos ruidosos.
- Problemas sin reglas conocidas (algoritmos) para efectuar la tarea de procesamiento.
- Problemas en los que existe algoritmo pero:
    - No es una solución portable.
    - El costo de la implementación algorítmica es desmesurado.
    - Existe la necesidad de una solución adaptativa continuada.
- Problemas en los que es imprescindible un procesamiento muy rápido (tiempo real).
- Aplicación de un sistema “general” de asociación o transformación de patrones:
    - Clasificación.
    - Reconstrucción de patrones.
    - Predicción.
    - Modelización.
    - Optimización.
- Aprendizaje de máquinas (‘machine learning’).

Las RNA no son convenientes en todos los casos. Aunque ofrecen una buena relación eficiencia/coste, presentan algunas desventajas con respecto a los sistemas logarítmicos:

- Se hace difícil la explicación de las reglas utilizadas por la red neuronal entrenada.
- Necesitan ejemplos con suficiente representatividad.
- Tienen aún problemas sin resolver:

- Dado un problema, se desconoce la arquitectura de la red más eficiente.
- Precisa altos requisitos de cómputo en el aprendizaje (no en ejecución).

Además, no siempre sobrepasan en eficiencia a las técnicas estadísticas, muy usadas también en los tipos de problema comentados anteriormente.

I.1.2.- Modelo formal de Neurona Artificial

I.1.2.1.- Operación

$$\eta = \eta_\beta + \sum_{i=1}^{n} \xi_i \omega_i$$

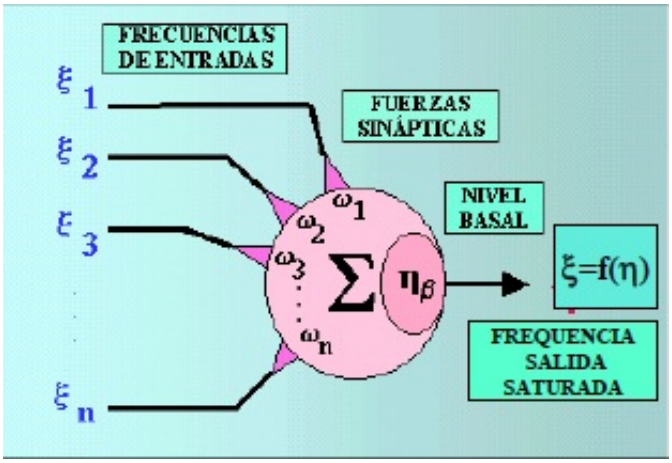


Fig. I.1.2.1.1: Esquema de la neurona artificial. Excitación, activación y salida.

Una neurona combina múltiples señales, que recibe de las sinapsis, mediante determinados pesos, y guarda el resultado en una memoria o variable local. En base a esto genera una salida, a través de una función que introduce no-linealidad, como vemos a continuación:

I.1.2.2.- Saturación de la respuesta: función no-lineal

$$\xi = \mathbf{f}(\eta)$$

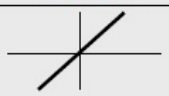
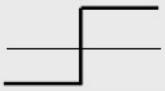
	Función	Rango	Gráfica
Identidad (lineal)	$y = x$	[-oo, +oo]	
Escalón	$y(x) = sign(x)$ $y(x) = \theta(x)$	[-1, +1] [0, +1]	
Lineal a Tramos	$y(x) = \begin{cases} -1 & si \quad x \le -1 \\ x & si \quad -1 < x < +1 \\ +1 & si \quad +1 \le x \end{cases}$	[-1, +1]	
Sigmoidea	$y(x) = \tanh(x)$ $y = \frac{1}{1+\exp(-x)}$	[-1, +1] [0, +1]	
Gausiana	$y = A \cdot \exp(-\sigma \cdot x^2)$	[0, +1]	

Fig. I.1.2.2.1: Tipos de funciones de salida.

I.1.3.- Modelo de Red Neuronal Artificial

Las RNA son redes de elementos procesadores simples interconectados masivamente, con organizaciones jerárquicas (capas o mapas), trabajando en paralelo, habitualmente adaptables y orientadas a interactuar con información imitando a los sistemas nerviosos.

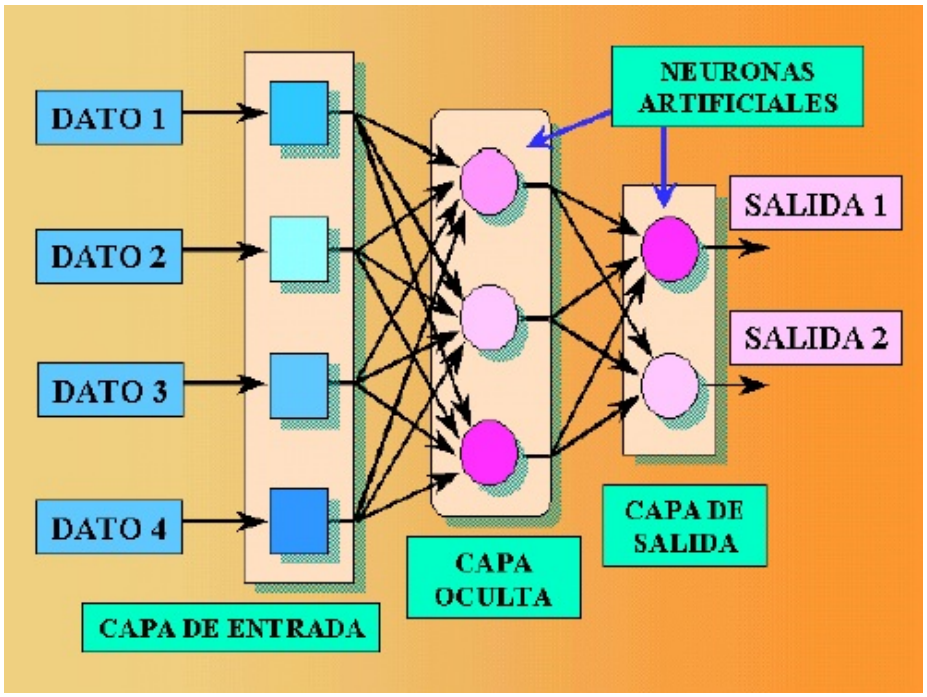


Fig. I.1.3.1: Organización de la red en capas.

Las RNA poseen las siguientes características:

- Aprendizaje: Ante un conjunto de entradas-salidas se autoajustan para producir respuestas consistentes.
- Generalización: Una red entrenada puede ser insensible a variaciones menores (ruido) en las entradas y responde correctamente a ejemplos nunca vistos anteriormente.
- Abstracción: Algunas RNA son capaces de abstraer la información útil de un conjunto de entradas.

I.2.- Objetivos

En este anexo se expondrá la metodología seguida para la implementación de los distintos clasificadores basados en redes neuronales que se utilizan en este proyecto. El conjunto de datos elegido para ilustrar este proceso será el obtenido en el experimento piloto, considerando sólo la información relativa a los biosensores (para más detalles, véase el capítulo 7 de la memoria). La estrategia para determinar el tipo de red neuronal y su arquitectura, así como el desarrollo del código necesario para su funcionamiento, serán similares tanto para el EEG como para los biosensores en posteriores experimentos.

Los pasos a seguir son:

- Realización del experimento. Montaje de biosensores. Adquisición de señales.
- Procesado de señales para la obtención de características fisiológicas básicas.
- Empleo de redes neuronales para la clasificación o detección de estados.

Una de las tareas previas más importantes es la obtención, a partir de las señales generadas por los sensores, de nuevas características o variables más adecuadas para efectuar la distinción entre los estados, éstas serán realmente las entradas del clasificador.

Como resultado de este tratamiento de las señales, y en base a toda la documentación estudiada previamente, las entradas a la red neuronal serán las siguientes:

- HRV (Heart Rate Variability, calculado a partir de la señal electrocardiograma).
- BVP-HRV (Heart Rate Variability, calculado a partir de la señal volumen de pulso sanguíneo).
- PINCH (Dilatación Vascular, calculado a partir de la señal volumen de pulso sanguíneo).
- RPM (Tasa de respiraciones por minuto, calculado a partir de la señal respiración).
- PROFRESP (Profundidad de la respiración, calculado a partir de la señal respiración).
- SCVAR (Skin Conductivity Variability, calculado a partir de la señal conductividad de la piel).

Con estas entradas alimentaremos una red neuronal basada en perceptrón multicapa (MLP), con la siguiente arquitectura: seis neuronas para la capa de entrada, una o dos capas ocultas con un número de neuronas en cada una aún por determinar, y dos neuronas en la capa de salida, una por cada clase (Relajación, estrés).

Disponemos de 826 patrones para cada variable, algunas de las cuales han sido submuestreadas para que todas tengan el mismo número de patrones, con lo que la matriz de datos de entrada a la red tendrá 6 filas (variables) y 826 columnas (patrones).

Con estos datos ya definidos, procederemos en primer lugar a hacer un preprocesado y a analizar la dependencia entre variables para usar Análisis de Componentes Principales (PCA) en caso de que sea necesario.

A continuación, buscaremos la arquitectura óptima mediante diversas técnicas que se comentarán más adelante, para finalmente evaluar la red con un método de validación cruzada, presentar los resultados obtenidos y las conclusiones.

I.3.- Resolución

I.3.1.- Organización de los datos de entrada a la red

Una vez que tenemos las 6 características o variables con las que vamos a trabajar, para cada una de ellas separamos los datos correspondientes a intervalos de actividad y descanso y los introducimos en un vector distinto. De esta forma nos quedan 12 vectores, uno por variable y estado (actividad, descanso), y podemos ver cuántas muestras tiene cada uno de ellos (aunque la frecuencia de muestreo inicial era de 256 Hz para todos los biosensores, las características que hemos obtenido tienen un número de puntos distinto). El que menos muestras tiene (413) es el vector que contiene la variable "RPM" en actividad, con lo cual submuestreamos las demás señales para que todas tengan el mismo número, y así todas las filas (variables) de la matriz de datos tengan el mismo número de columnas (patrones). Este submuestreo conlleva una pérdida de información que en alguna de las señales puede ser importante, más adelante veremos si esto puede influir en la bondad del clasificador.

Como resultado de las operaciones anteriores, tenemos 12 vectores de 413 muestras cada uno. Para construir la matriz de datos, en cada fila (variable) colocamos primero los elementos del vector correspondiente a esa variable y al estado descanso, y después los del estado actividad. De esta forma obtenemos una matriz de dimensiones 6 variables x 826 patrones, de los cuales los 413 primeros pertenecen a estados de descanso y los 413 restante a estados de actividad. Las variables son, en este orden:

1. RPM
2. PROFRESP
3. HR
4. BVPHR
5. PINCH
6. SCVAR

Además, en un vector columna de tamaño 826 llamado "clase" marcaremos con un "1" los 413 primeros elementos (salida deseada "Relaxation"), y con un "2" los 413 restantes (salida deseada "Stress"). Tanto este vector como la matriz "datos", además de la versión escalada entre -1 y 1 de la matriz de datos "dataesc" y otras variables necesarias para la ejecución de los programas, han sido guardadas en el archivo "matrizdatosescalados.mat", que será necesario cargar en el workspace de MATLAB© antes de iniciar dicha ejecución.

I.3.2.- Preprocesado de los datos

I.3.2.1.- Visualización de los datos

Contamos ya con un fichero de MATLAB© que contiene los datos necesarios para abordar el problema de clasificación. En primer lugar, visualizaremos los datos mediante la función *preprocess* de la SOM-Toolbox . Para ello ejecutamos el archivo "preprocess1.m".

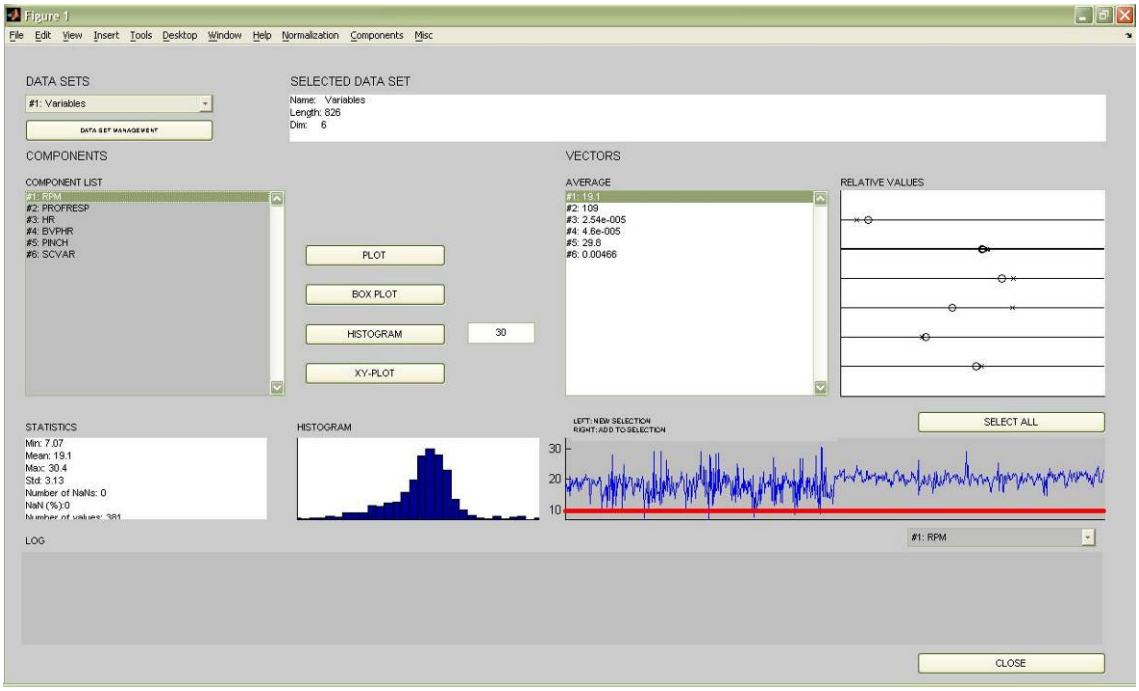


Fig. I.3.2.1.1: Herramienta *preprocess* con los datos del problema.

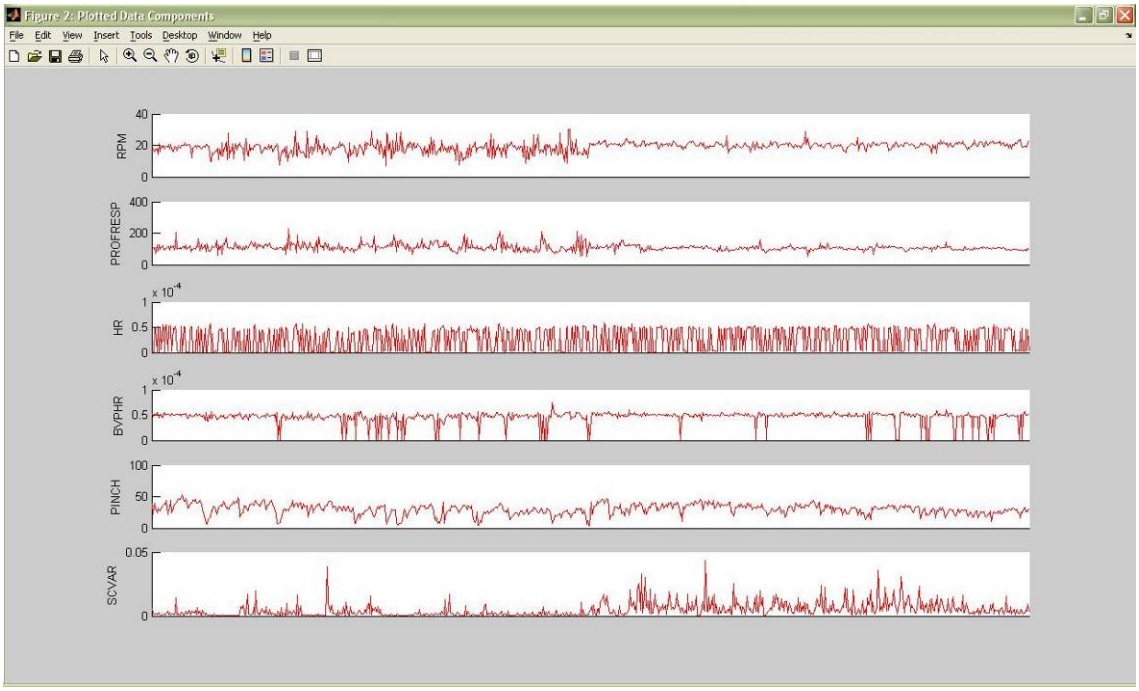


Fig. I.3.2.1.2: Visualización de variables con la opción *plot*.

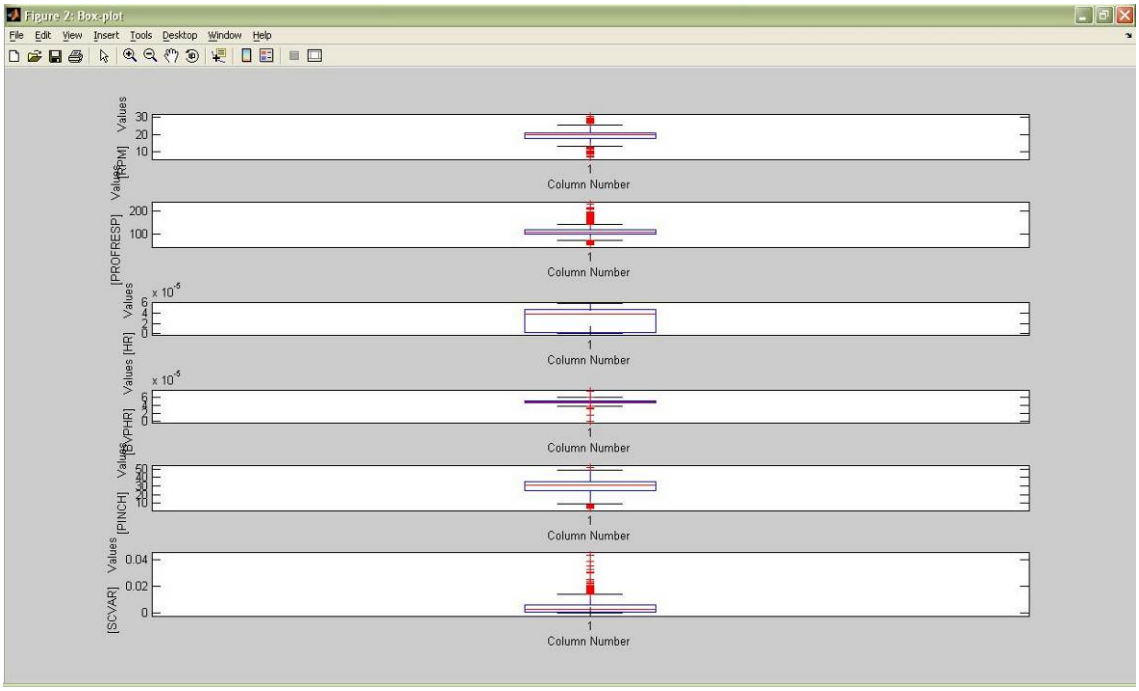


Fig. I.3.2.1.3: Visualización de variables con la opción *boxplot.*

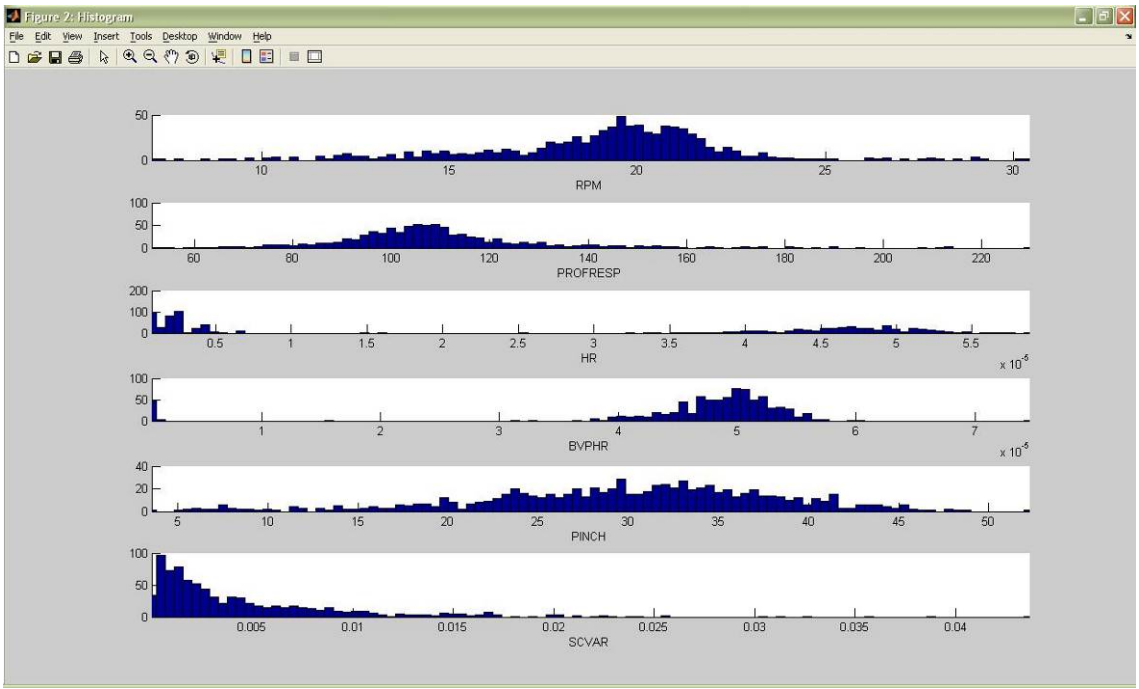


Fig. I.3.2.1.4: Vista de los histogramas de las variables con la opción *histogram.*

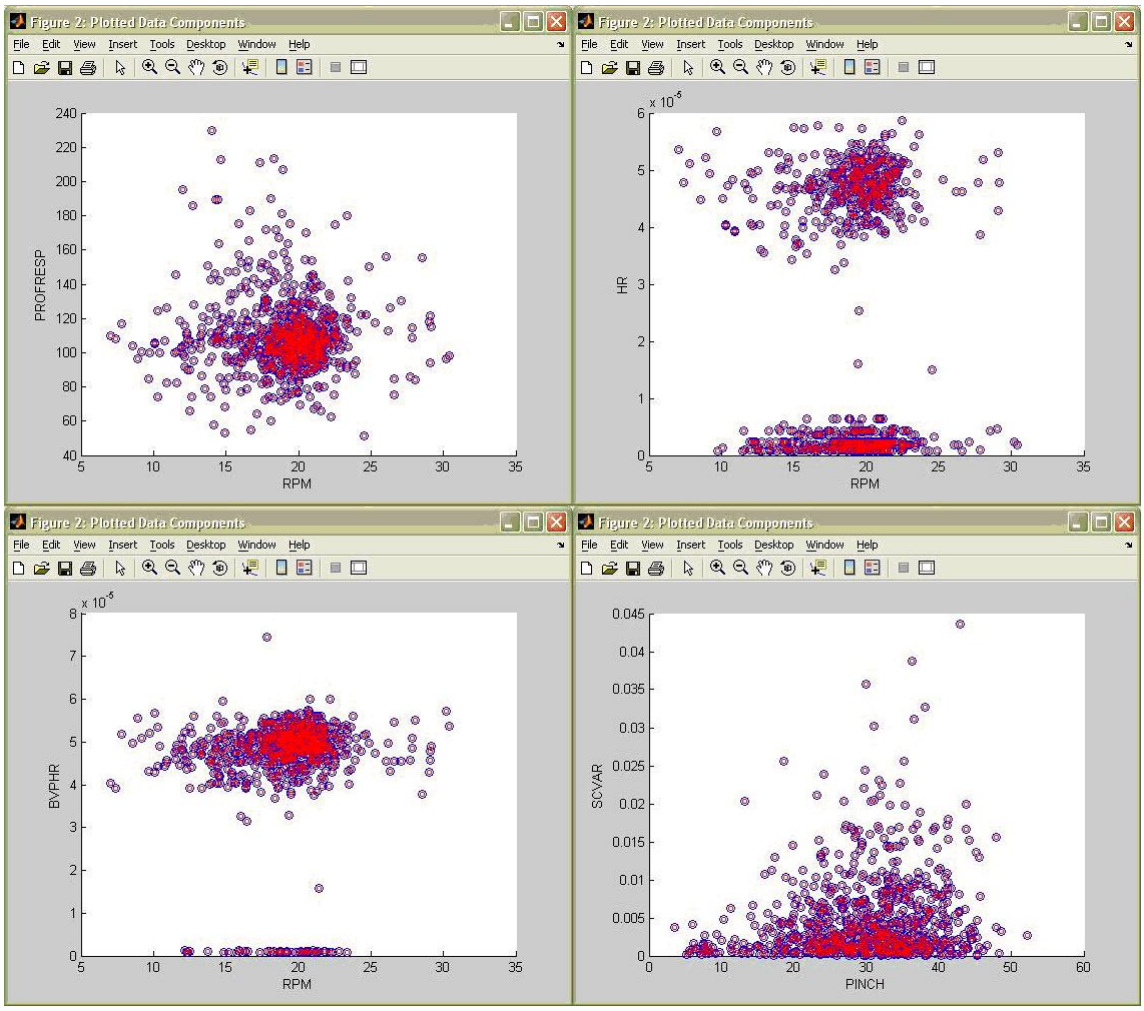


Fig. I.3.2.1.5: Visualización por pares de variables con la opción *XYplot.*

Como nos muestran las figuras anteriores, parece que hay problemas con las variables HR y BVPHR, que presentan distribuciones un poco extrañas. Es posible que esto afecte a la hora de hacer la clasificación, así que si los resultados son malos, ya sabemos dónde puede estar la clave. En cualquier caso, cabe comentar que todas las variables son fuertemente no lineales, cosa que habrá que tener en cuenta posteriormente.

El archivo "preprocess2.m" escala los datos a media nula y desviación estándar unidad, mientras que "preprocess3.m", además de normalizarlos, los escala entre -1 y 1. Los resultados son similares a los anteriores, por lo que no se incluyen las figuras.

I.3.2.2.- Análisis de dependencia de variables

Como comentamos anteriormente, las variables son no lineales, con lo que difícilmente van a ser linealmente dependientes entre ellas. Podemos comprobarlo ejecutando el fichero "variablesdependientes.m":

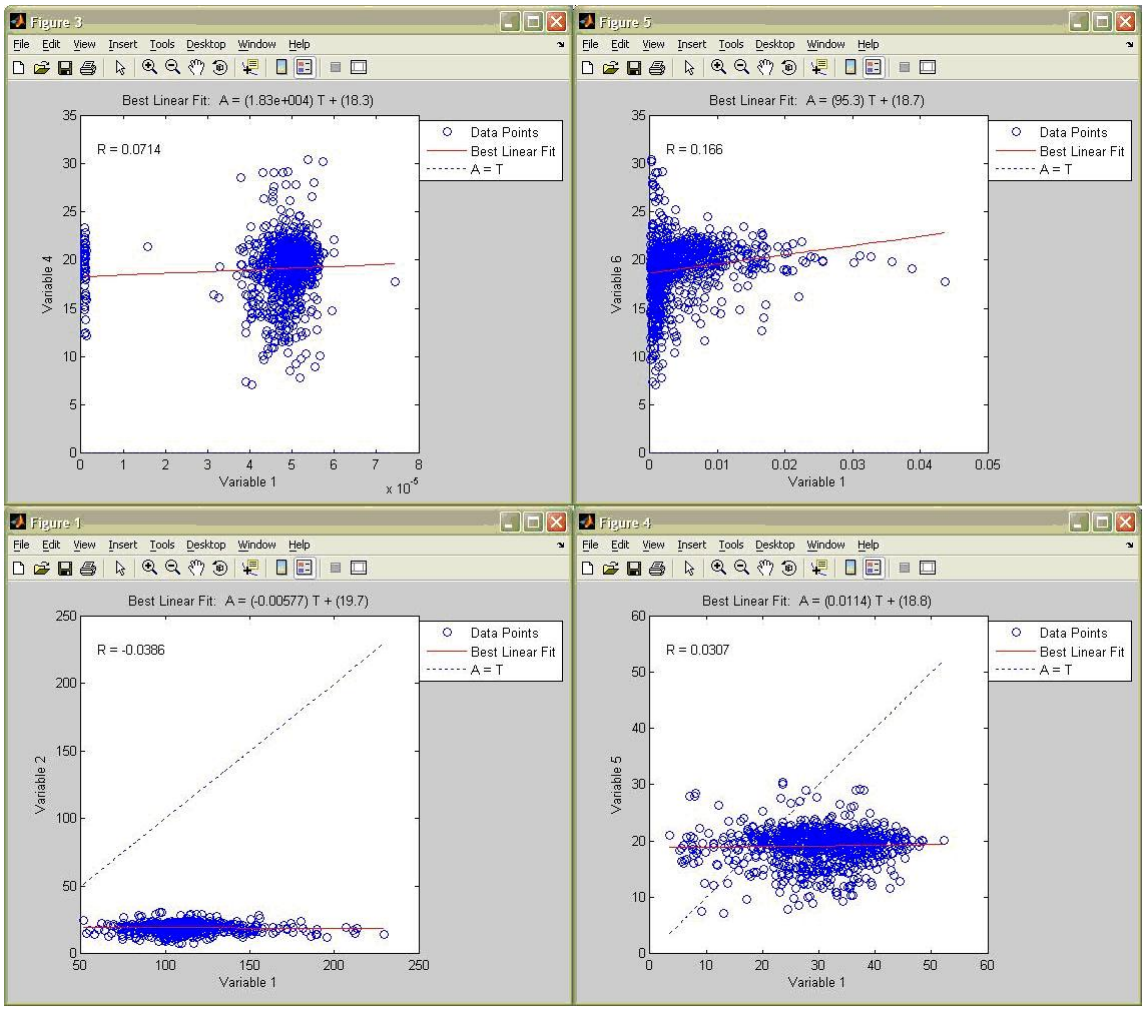


Fig. I.3.2.2.1: Análisis de dependencia lineal entre variables.

I.3.2.3.- Análisis de Componentes Principales (PCA)

A continuación efectuaremos un análisis PCA, presentaremos los valores propios de la matriz de covarianzas y mostraremos algunas representaciones de pares de componentes PCA:

Valores propios de la matriz de covarianzas:
ans =

0.5925 0.1481 0.0801 0.0745 0.0538 0.0509

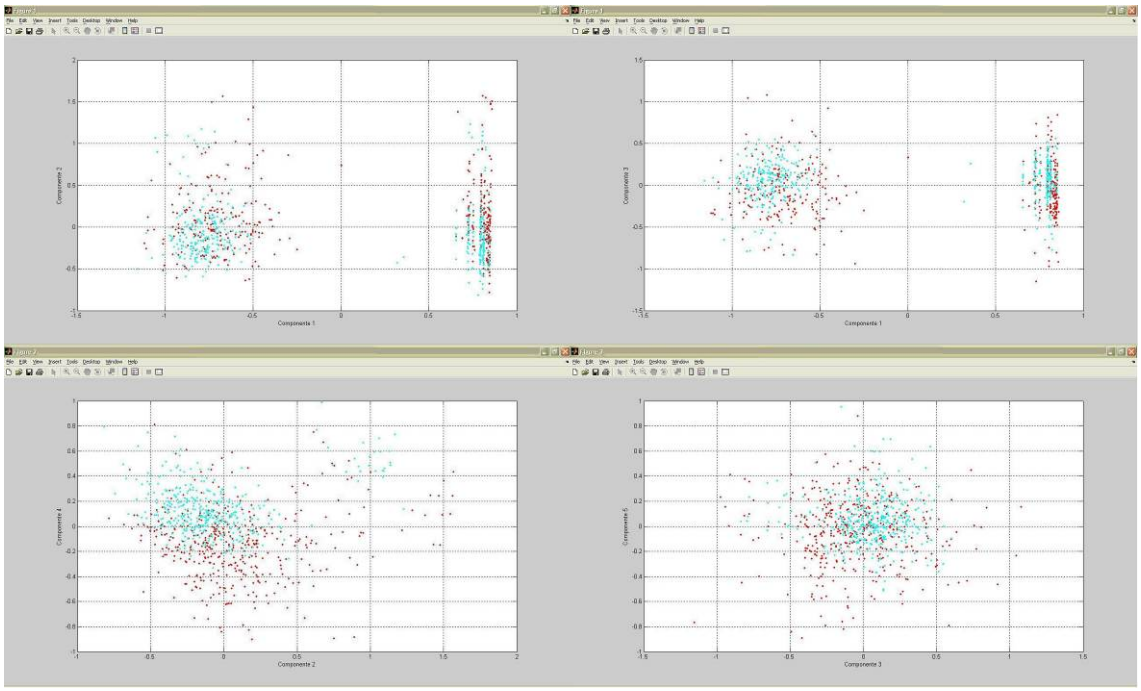


Fig. I.3.2.3.1: Proyecciones de pares de componentes PCA.

Podemos observar que ninguna de estas componentes PCA nos permite separar las clases con claridad, por lo que, al ser el número de variables pequeño y no ofrecernos este método ninguna ventaja, de aquí en adelante desestimaremos el uso de esta herramienta para este problema concreto.

I.3.3.- Solución con una capa oculta

I.3.3.1.- Primera aproximación

Para hacernos una idea del tamaño de la red, ejecutaremos el programa "biosignalsMLP", sin aplicar PCA, con una capa oculta de 15 neuronas, y 100 ciclos de entrenamiento mediante el método de regularización bayesiana.

```
TRAINBR, Epoch 100/100, SSE 134.732/0.001, SSW 253.679, Grad 7.06e-001/1.00e-010, #Par 1.27e+002/137
```

Observamos en la respuesta de la red en el último ciclo que se han utilizado 127 pesos de 137, con lo que el número óptimo de neuronas posiblemente estará más o menos cercano a 15.

I.3.3.2.- Estimación de la arquitectura óptima

A continuación, mediante la función "MLParquitectura.m" simularemos un pool de 10 redes para cada arquitectura que elijamos, con el objetivo de comprobar cuál es la óptima. Utilizaremos redes con salida binaria en las neuronas de salida, y salidas entre -1 y 1 en la capa oculta. Entrenaremos las redes con el método de Levenberg-Mardquart

durante 100 ciclos. Simularemos 9 arquitecturas de 7, 8, 9, 10, 11, 12, 13, 14 y 15 neuronas ocultas cada una. Los resultados son los siguientes:

```
Resultados para el promedio de las 10 redes en la arquitectura con 9 neuronas ocultas:

Matriz_Confusion =

   356    45
    57   368


ans =

    0.8321


muestras_nulas =

     0
```

```
Resultados para el promedio de las 10 redes en la arquitectura con 13 neuronas ocultas:

Matriz_Confusion =

   355    41
    58   372


ans =

    0.8356


muestras_nulas =

     0
```

```
Red con error minimo en arquitectura con 8 neuronas ocultas y prueba 3.

Matriz_Confusion =

   357    39
    56   374


Kappa =

    0.8422


muestras_nulas =

     0
```

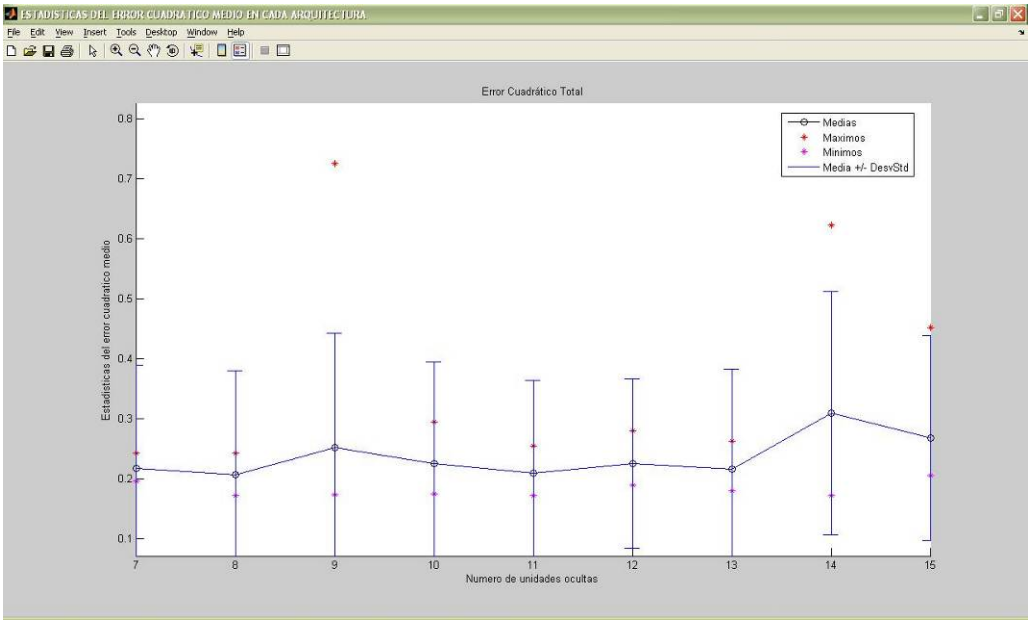


Fig. I.3.3.2.1: Estadísticas del error para todas las redes simuladas.

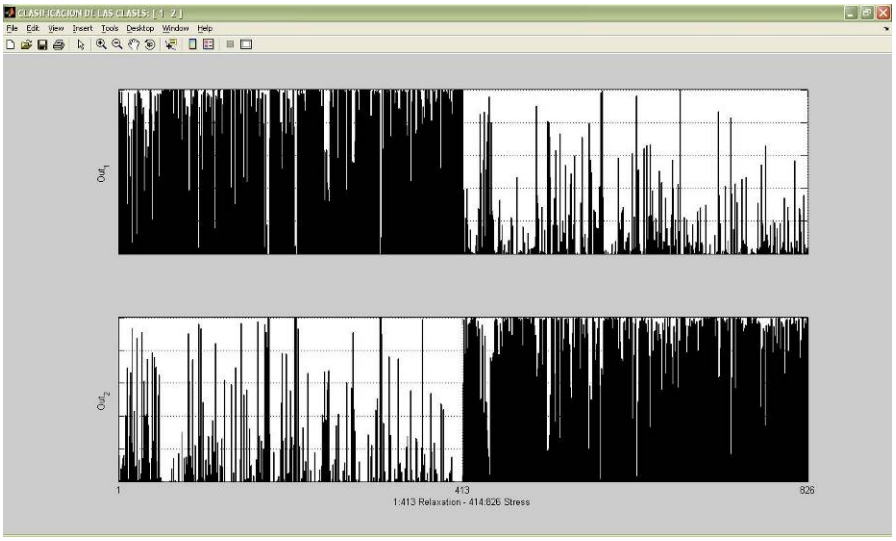


Fig. I.3.3.2.2: Clasificación, usando todas las muestras, de la red con error mínimo.

Por tanto, al ser muy parecidos los resultados a partir de 9 neuronas ocultas (aunque la red con 8 ha dado el mejor resultado, en promedio es peor), elegiremos la red 6:9:2 como arquitectura óptima con una capa oculta.

I.3.3.3.- Evaluación de la arquitectura elegida con validación cruzada

Ejecutamos la función "MLPCV.m", entrenamos un pool de 10 redes para la arquitectura de 9 neuronas ocultas, con el método de Levenberg-Marquardt durante 50 ciclos. Efectuamos 3 validaciones cruzadas, obteniendo estos resultados:

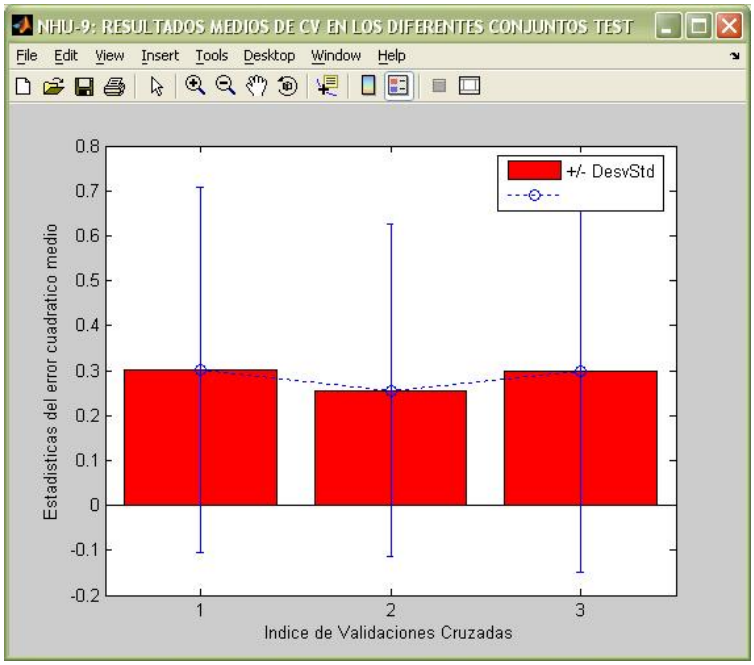


Fig. I.3.3.3.1: Error cuadrático medio en cada conjunto de prueba.

Por último, evaluamos nuestra red mediante la rutina “MLPCVevaluacion.m”.

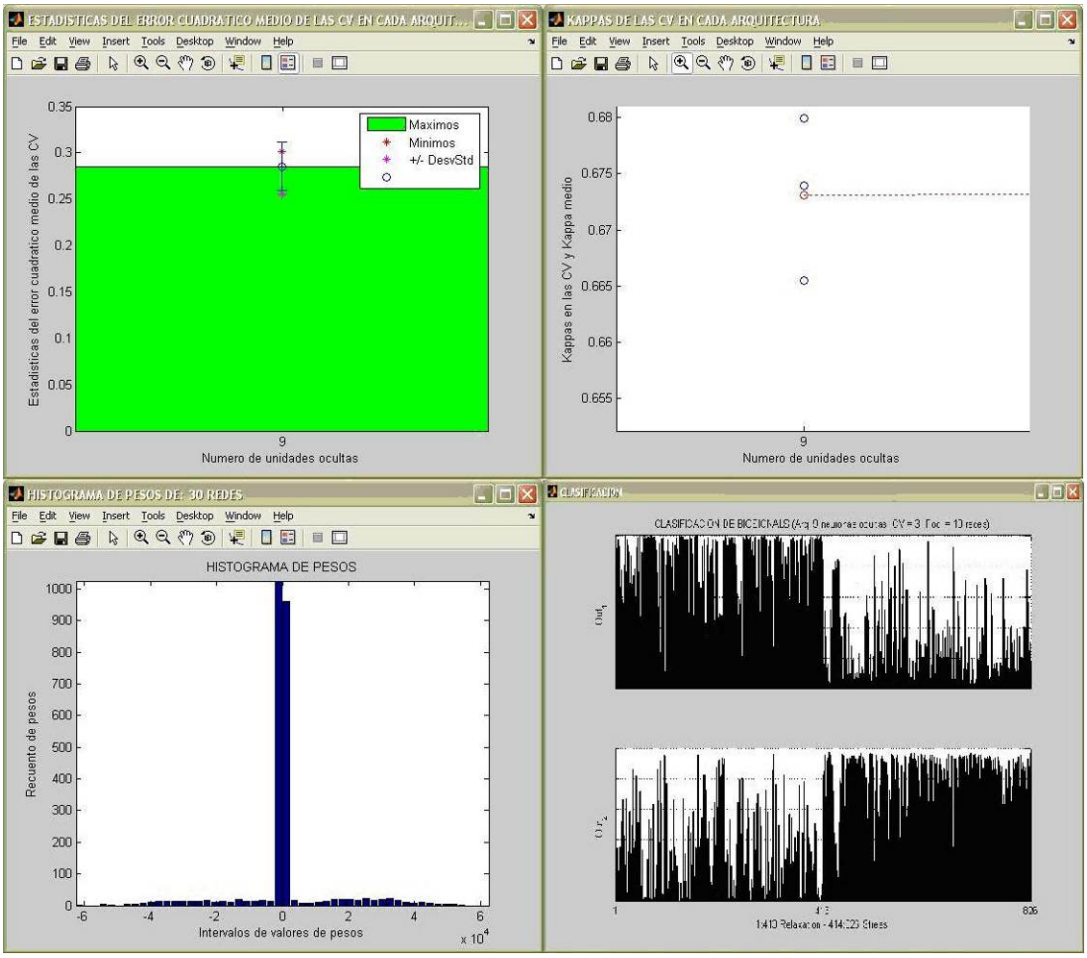


Fig. I.3.3.3.2: Estadísticas del error, kappa medio, histograma de pesos y clasificación.

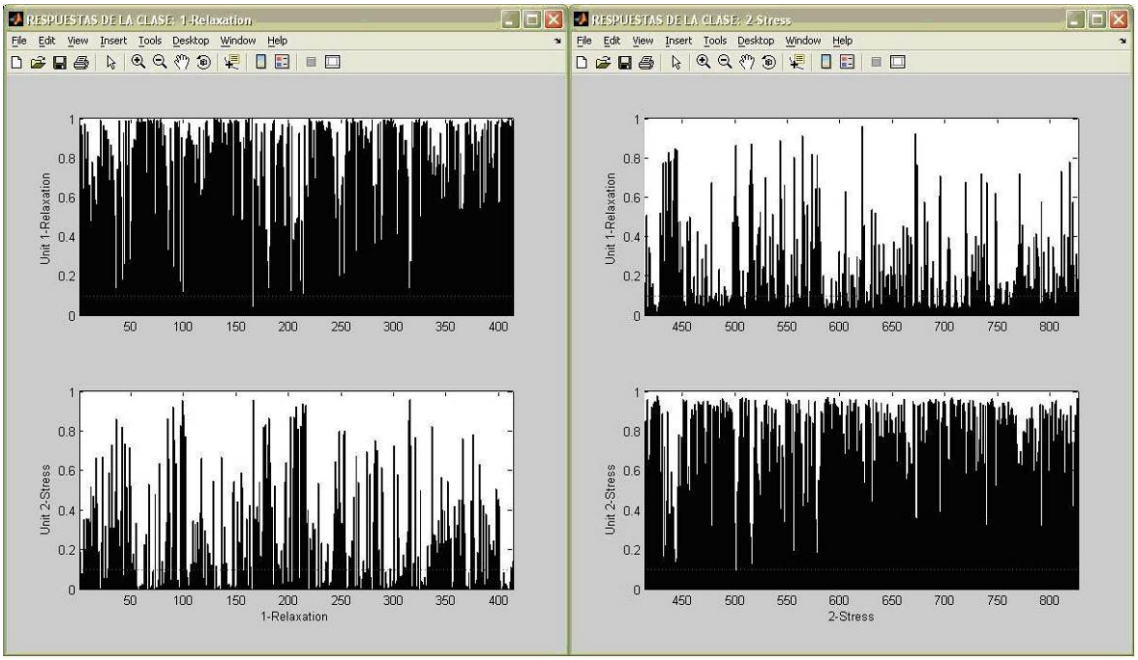


Fig. I.3.3.3.3: Respuestas de la clase 1.-Relaxation (izquierda) y 2.-Stress (derecha).

I.3.4.- Solución con dos capas ocultas

I.3.4.1.- Estimación de la arquitectura óptima

Entrenaremos varias arquitecturas con la función “MLParquitectura2capas1fija”, de la misma forma que para una capa oculta. Probaremos con una primera capa oculta de 15 neuronas, e iremos variando el número de elementos de la segunda: 5, 7, 10, 12 y 14:

```
Resultados para el promedio de las 10 redes en la arquitectura con 10 neuronas ocultas:

Matriz_Confusion =

   380    26
    33   387


ans =

    0.9037


muestras_nulas =

     0
```

```
Red con error minimo en arquitectura con 12 neuronas ocultas y prueba 6.

Matriz_Confusion =

   394    15
    19   398


Kappa =

    0.9448


muestras_nulas =

     0
```

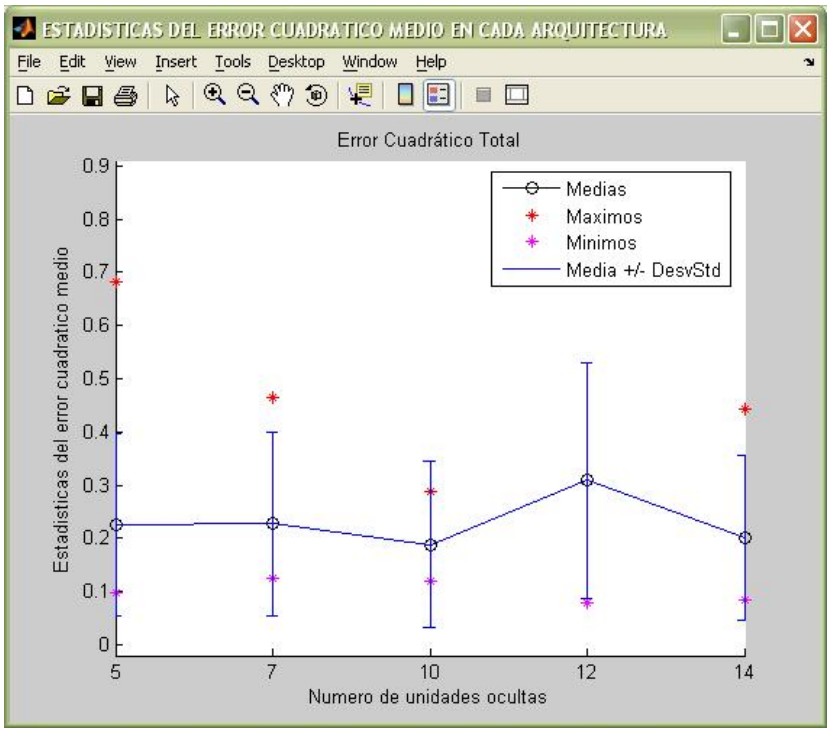


Fig. I.3.4.1.1: Estadísticas del error para todas las redes simuladas.

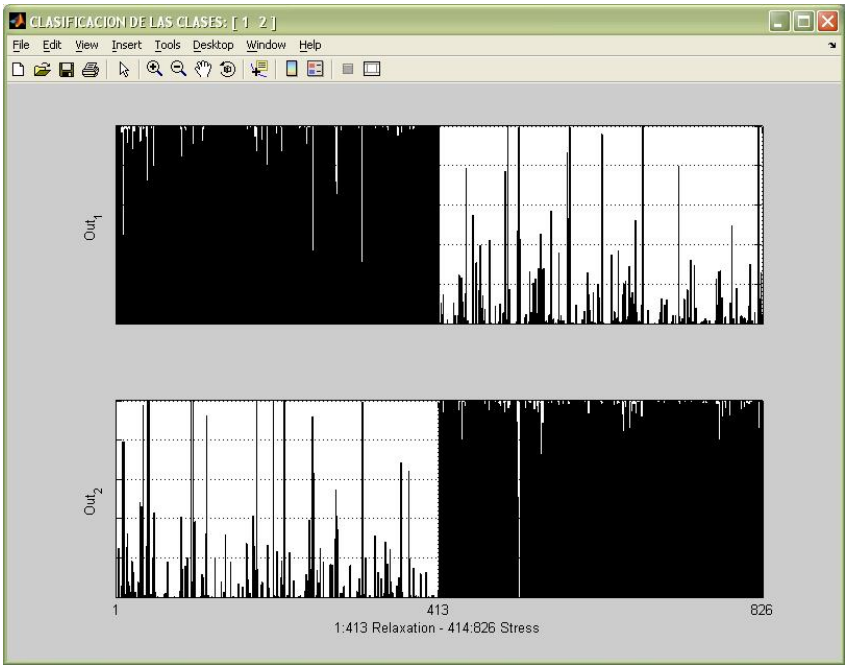


Fig. I.3.4.1.2: Clasificación, usando todas las muestras, de la red con error mínimo.

Seleccionamos, por tanto, la red con 10 neuronas en la segunda capa, y la arquitectura será 6:15:10:2.

I.3.4.2.- Evaluación de la arquitectura elegida con validación cruzada

Entrenamos la red elegida en el programa "MLPCV2capas.m", exactamente como hicimos con la red de una capa.

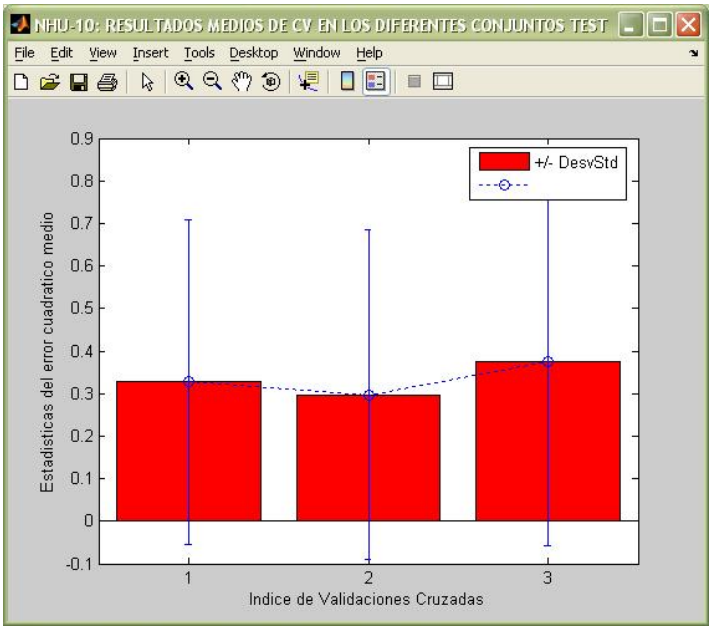


Fig. I.3.4.2.1: Error cuadrático medio en cada conjunto de prueba.

Y de nuevo, ejecutamos la función "MLPCVevaluacion.m":

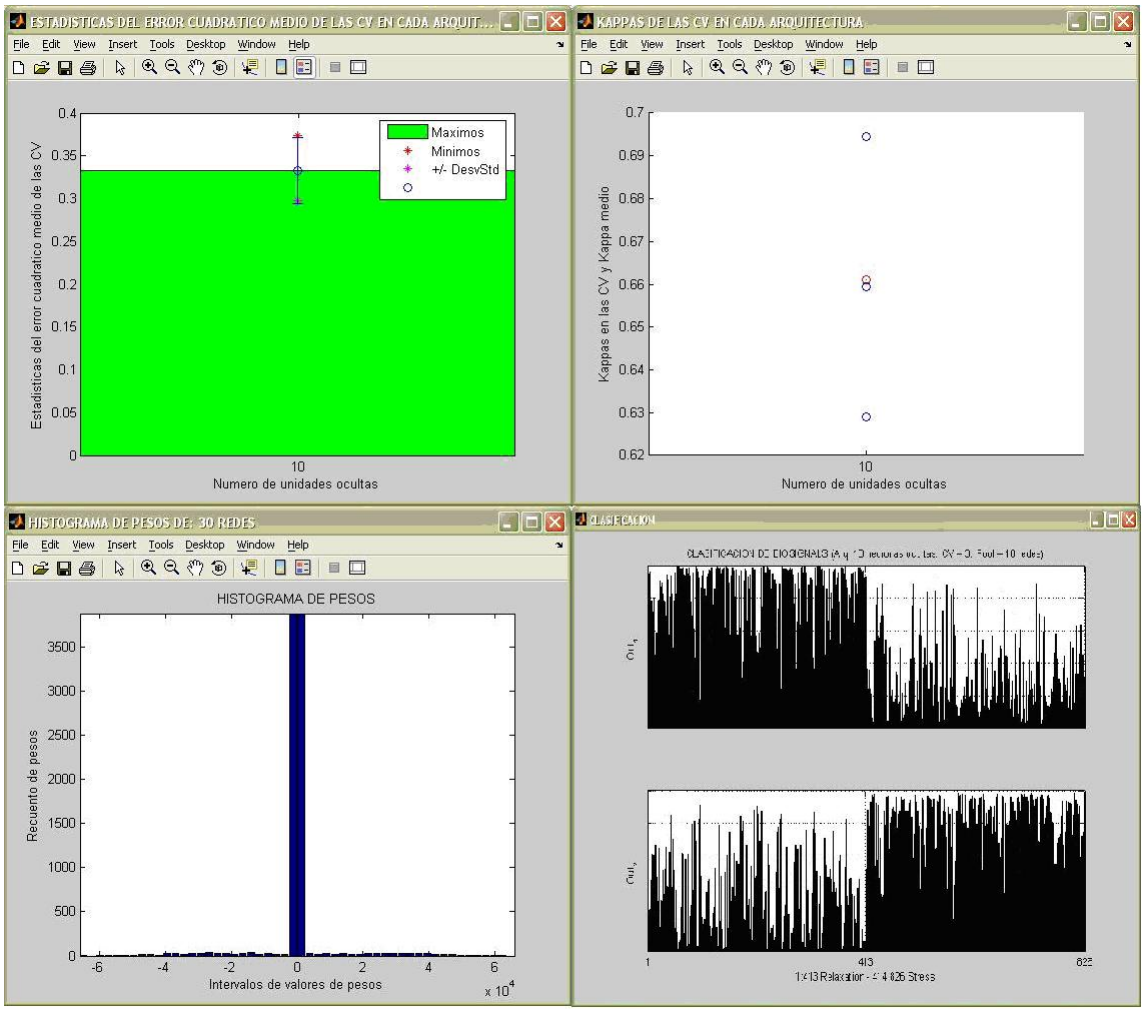


Fig. I.3.4.2.2: Estadísticas del error, kappa medio, histograma de pesos y clasificación.

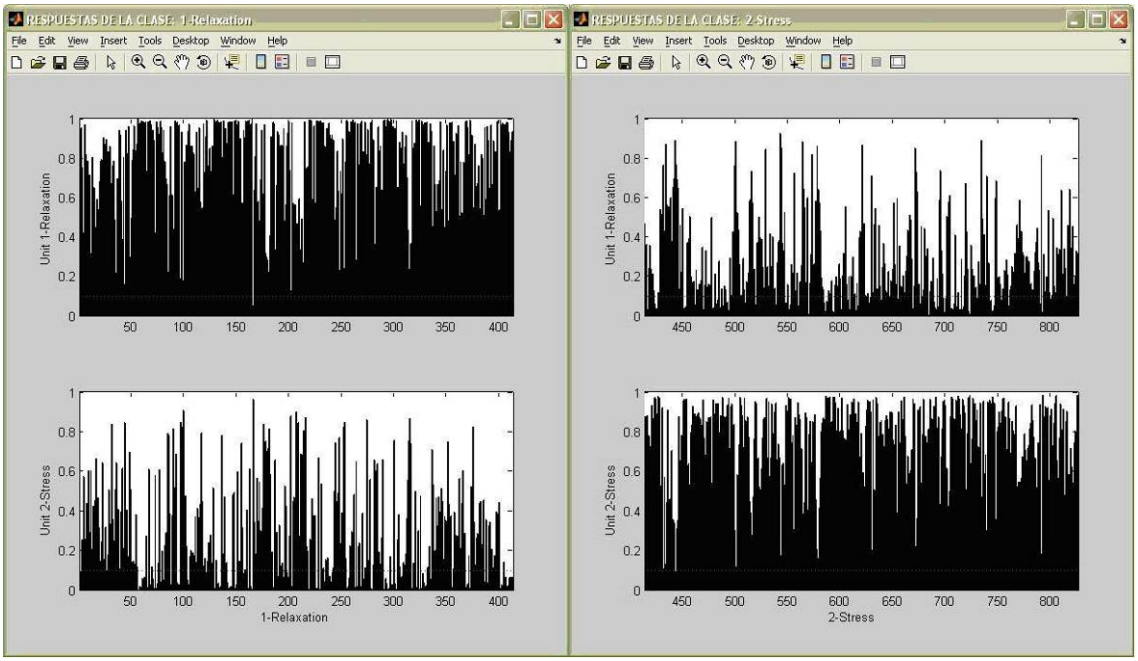


Fig. I.3.4.2.3: Respuestas de la clase 1.-Relaxation (izquierda) y 2.-Stress (derecha).

I.4.- Resultados preliminares

En las gráficas anteriores para la arquitectura con una capa oculta podemos observar que los resultados empeoran al evaluar la red con validación cruzada, lo cual es lógico ya que en ese caso usamos una parte de los patrones para entrenar la red y otra parte para evaluarla, en lugar de usar todos los patrones para el entrenamiento. De este modo, pasamos de un valor de Kappa de 0.8321 a 0.6731 (valor medio) en la validación cruzada. También comprobamos que al ser un problema no lineal, una capa oculta no es suficiente para hacer una buena clasificación.

Probamos entonces con una arquitectura de dos capas ocultas, y la clasificación usando todas las muestras mejora bastante, aunque otros factores como el tiempo de cómputo empeoran ostensiblemente. Ello prueba que la red se aprende mejor los datos, pero esto no significa que vaya a generalizar mejor, porque de hecho en la validación cruzada pasamos de una Kappa de 0.9037 a 0.6610, valor incluso peor que en el caso anterior.

Como curiosidad comentaremos que se ha probado una arquitectura con tres capas ocultas (6:25:15:7:2), obteniendo un valor de Kappa igual a 0.9421, y a pesar de ello en la evaluación con validación cruzada el Kappa medio es de 0.6271.

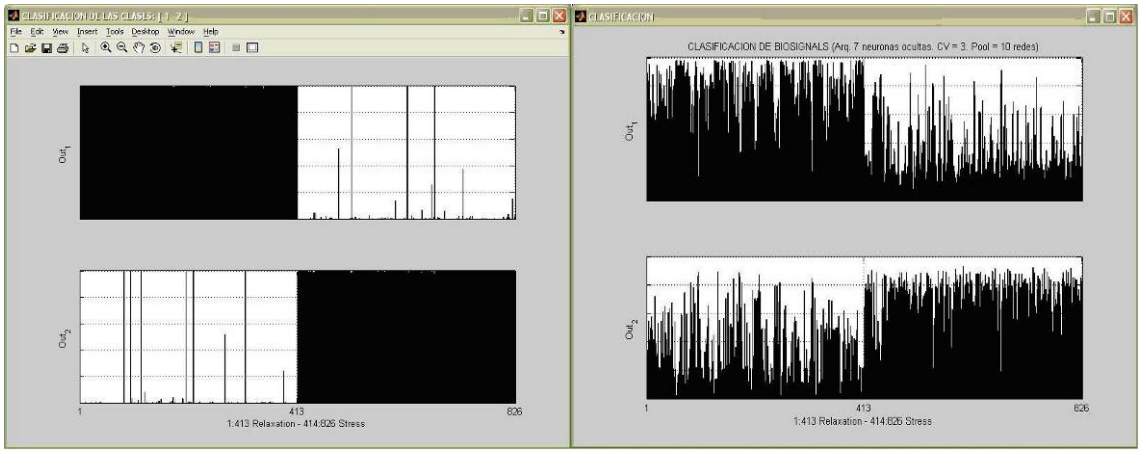


Fig. I.4.1: Clasificación usando todas las muestras para entrenamiento y validación cruzada.

Esto significa que para mejorar el clasificador habría que centrar los esfuerzos en depurar los datos de partida, en lugar de proponer redes más complejas.

I.5.- Modificaciones y mejoras

A la vista de los resultados anteriores, revisaremos los datos de entrada a la red, poniendo un énfasis especial en las variables HR y BVPHR, que, como mostramos en el preprocesado, son las que presentaban distribuciones más confusas.

Se han suavizado estas señales con un filtro de media móvil de 50 puntos, y se han depurado outliers (ahora tenemos 375 patrones para la clase “Relaxation” y 413 para la clase “Stress”, 788 en total), de manera que una nueva visualización de las variables quedaría de la forma siguiente (cargamos previamente en el workspace de MATLAB© el archivo “/datosrevisados/matrizdatosescaladossinoutliers”):

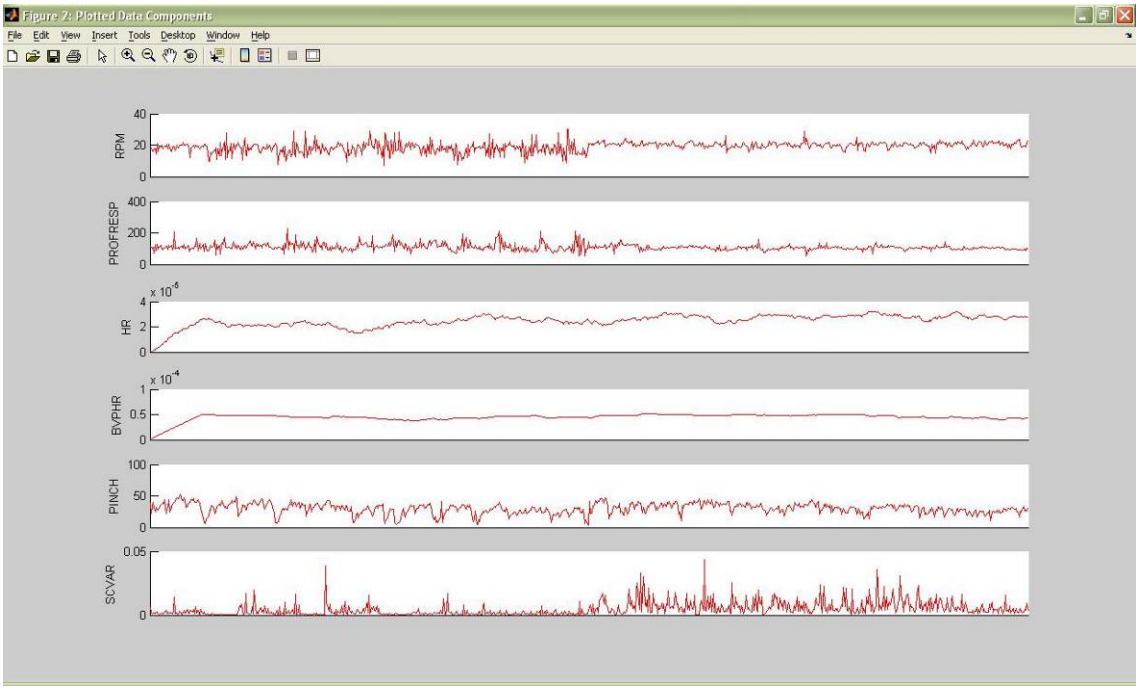


Fig. I.5.1: Nuevos datos de entrada a la red.

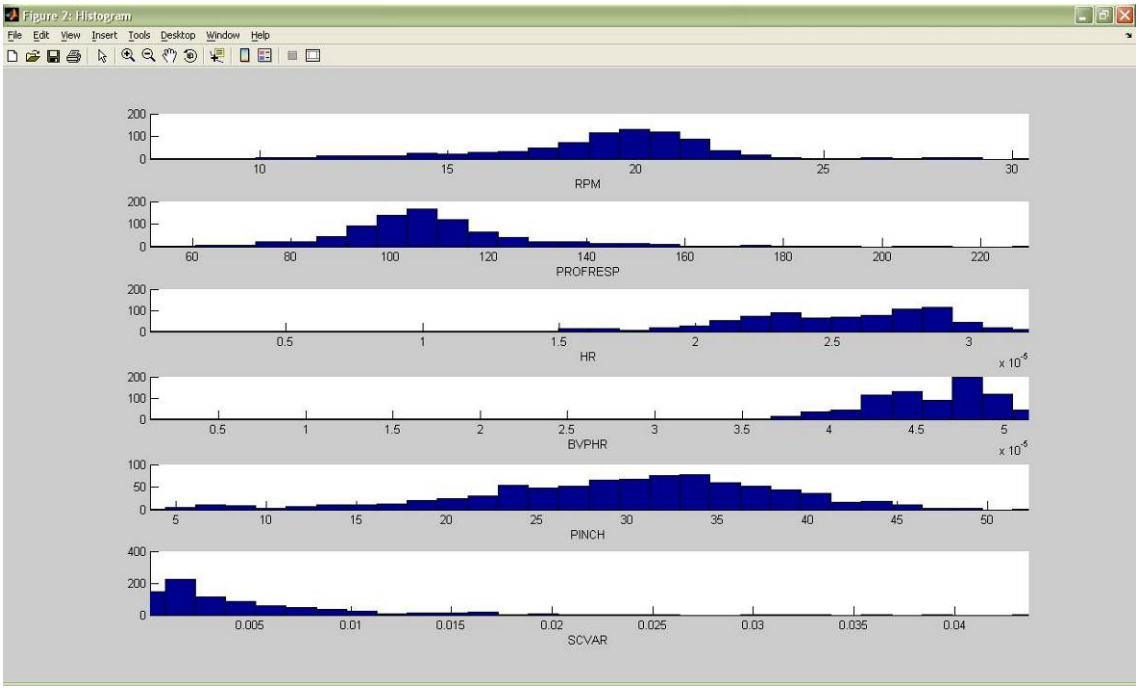


Fig. I.5.2: Histogramas de las nuevas variables.

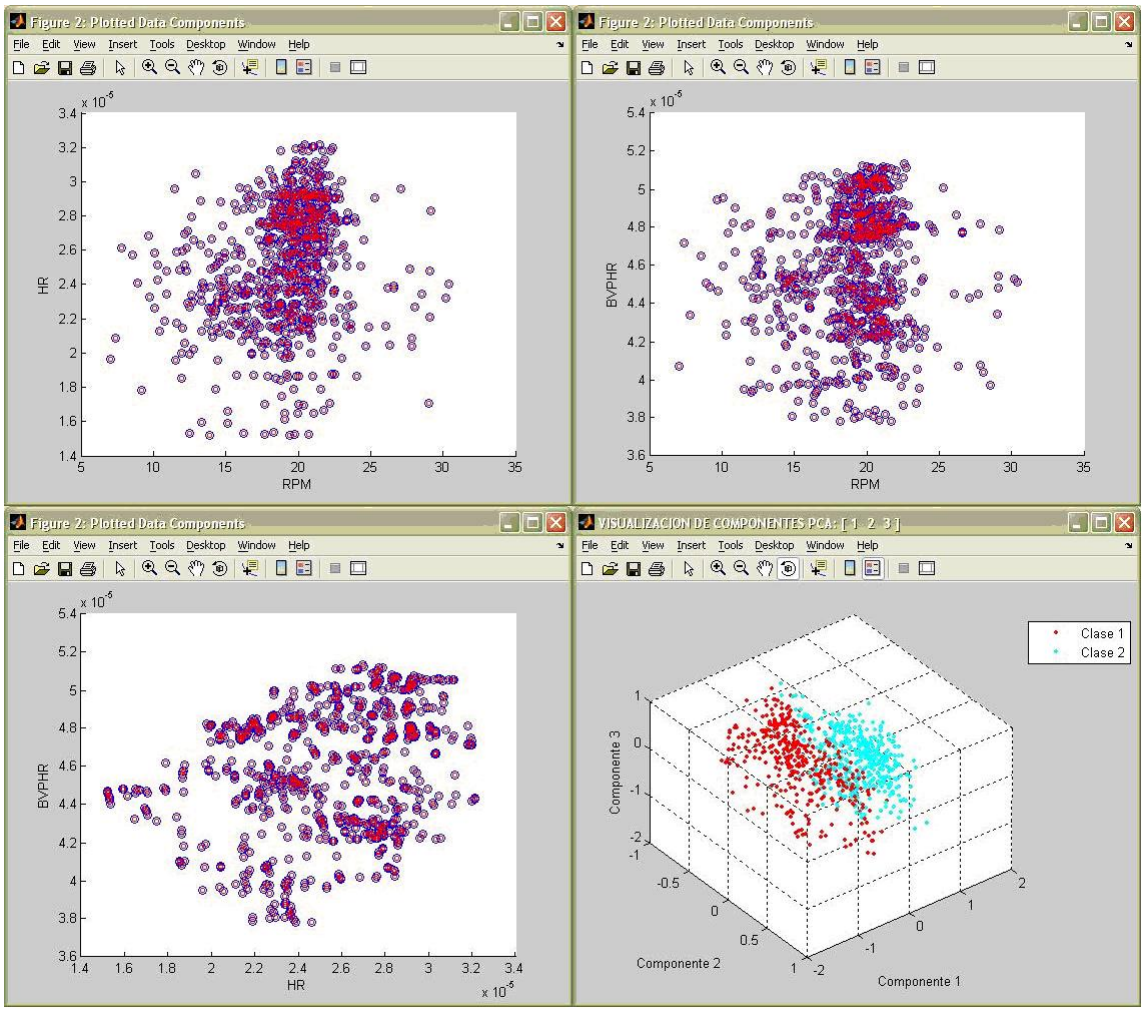


Fig. I.5.3: Representación XY de algunas variables y análisis PCA (abajo, derecha).

Además, en la figura podemos comprobar que con los datos depurados, la aplicación de PCA quizá nos ayude a separar las clases.

Por lo tanto, a continuación probaremos una red de una capa oculta. Estimaciones preliminares nos indicaron que de 10 a 15 neuronas podría ser un número óptimo, con lo cual, ejecutamos el programa "/datosrevisados/ MLParquitecturaconPCAsinoutliers.m", aplicando PCA y con las mismas opciones que anteriormente. Tras varias inspecciones, nos decantamos por una red **6:15:2**.

Resultados para el promedio de las 10 redes en la arquitectura con 15 neuronas ocultas:

Matriz_Confusion =

```
   375     0
     0   413
```

ans =

```
     1
```

muestras_nulas =

```
     0
```

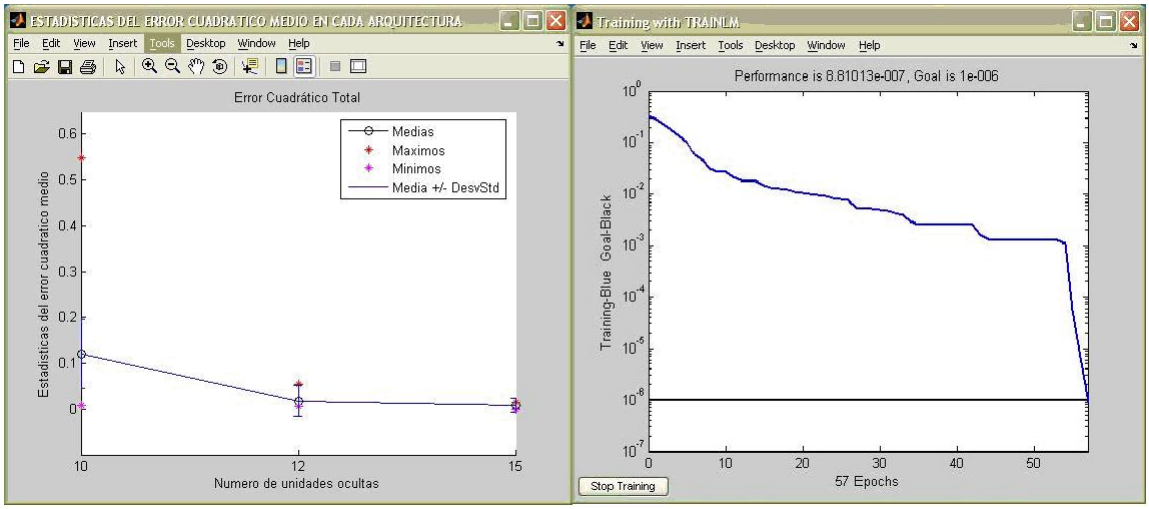


Fig. I.5.4: Estadísticas del error para algunas de las redes simuladas y entrenamiento.

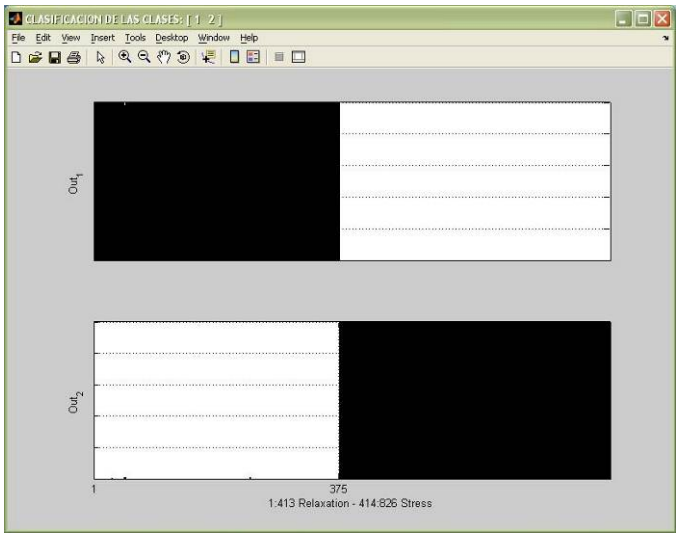


Fig. I.5.5: Clasificación usando todas las muestras de la red.

La figura nos muestra que la clasificación con esta arquitectura usando todas las muestras ha sido perfecta. Evaluamos la red con tres validaciones cruzadas, mediante las funciones "/datosrevisados/MLPCVconPCA" y "/datosrevisados/MLPCVevaluacionPCAsinoutliers", de la misma forma que antes:

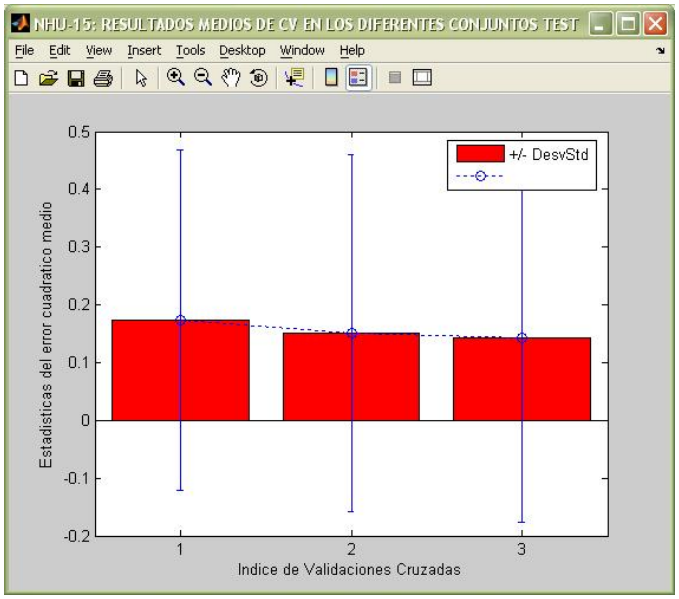


Fig. I.5.6: Error cuadrático medio en cada conjunto de prueba.

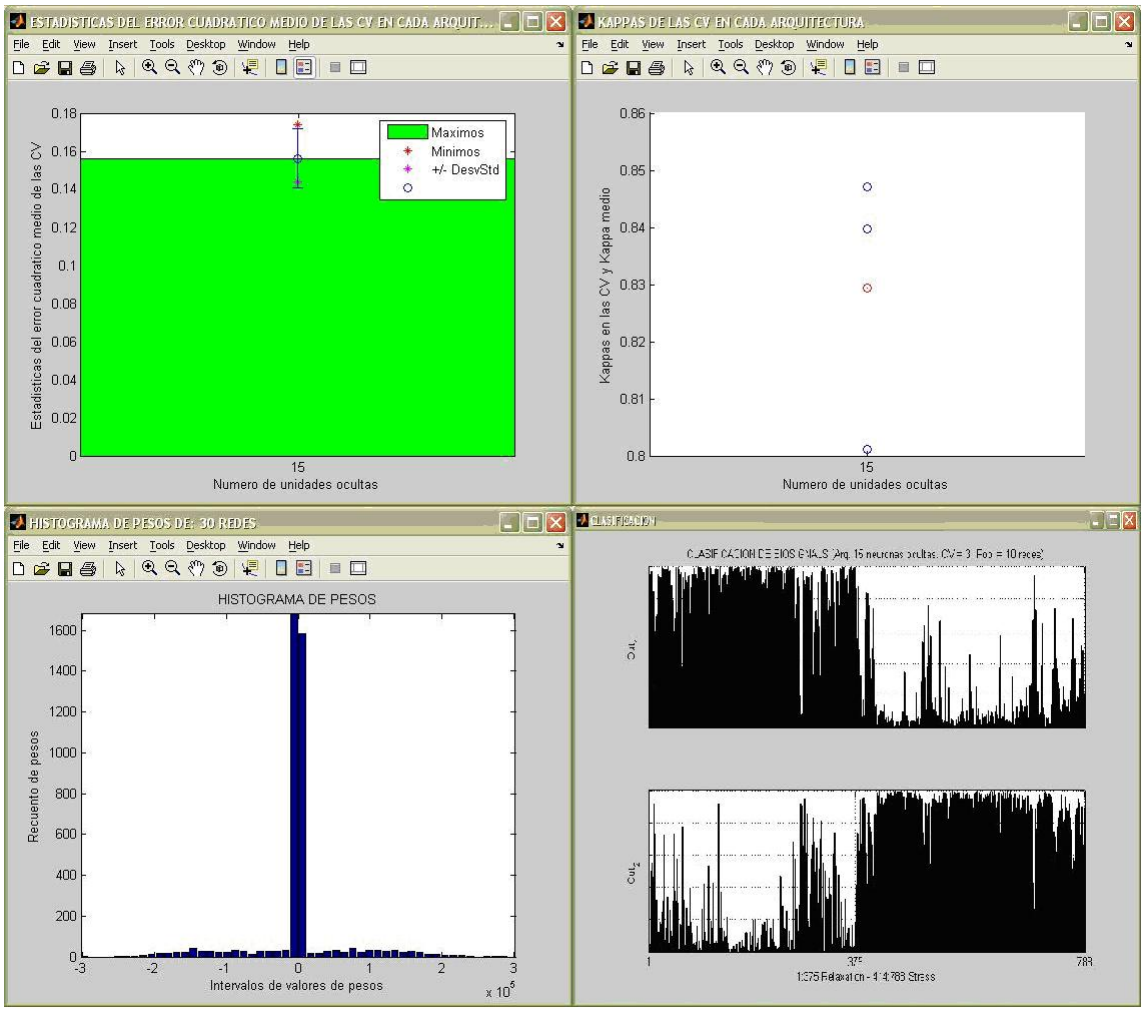


Fig. I.5.7: Estadísticas del error, kappa medio, histograma de pesos y clasificación.

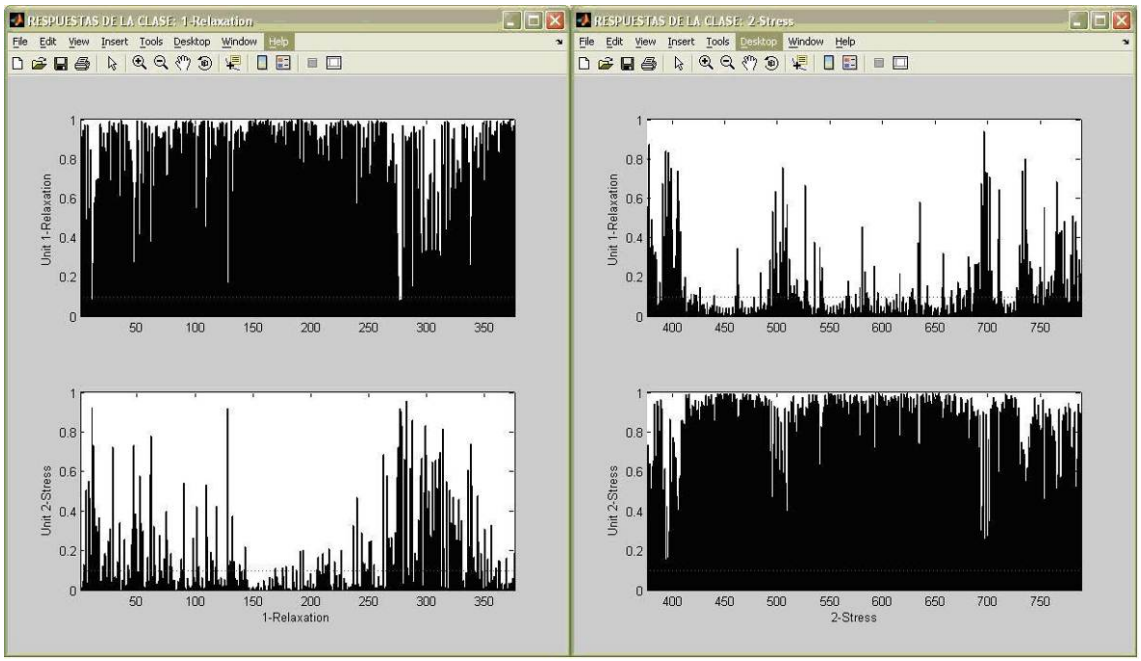


Fig. I.5.8: Respuestas de la clase 1.-Relaxation (izquierda) y 2.-Stress (derecha).

I.6.- Resultados finales

Finalmente, hemos podido comprobar cómo mediante la depuración de los datos se ha mejorado mucho la clasificación (kappa = 1 con todas las muestras, 0.8777 de media con una validación cruzada, 0.8421 con dos validaciones cruzadas, 0.8294 con tres validaciones cruzadas), sin necesidad de añadir más capas ocultas (de hecho, con dos capas ocultas, los resultados son peores).

I.7.- Conclusiones

En primer lugar cabe destacar, como hemos comentado anteriormente, que las vías de mejora para el detector de estados pasan por modificar la extracción de características, ya sea añadiendo variables nuevas o aplicando alguna técnica para que sean más fiables las actuales.

Por otra parte, los resultados finales están en la línea de los obtenidos en estudios anteriores, y son acordes en general con la literatura existente sobre el tema. La diferencia es que en algunos de esos estudios, los métodos usados para inducir estos estados emocionales en los usuarios son mucho más eficaces (estímulos audiovisuales, encuestas psicológicas, etc.), y de este modo los resultados mejoran, aunque también hay que decir que en buena parte de ellos se trata con varios individuos en lugar de uno, lo que aumenta considerablemente la dificultad.