Universidad de Alcalá
Escuela Politécnica Superior

INGENIERÍA TÉCNICA DE TELECOMUNICACIONES

Especialidad en Telemática

Trabajo Fin de Carrera

ANÁLISIS DE SEÑALES ELECTROENCEFALOGRÁFICAS CON MATLAB 7

Autora: Ángeles Trillo Ramos

Tutor: Don Luciano Boquete Vázquez

Año 2.017

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

INGENIERÍA TÉCNICA DE TELECOMUNICACIONES

Especialidad en Telemática

TRABAJO FIN DE CARRERA

ANÁLISIS DE SEÑALES ELECTROENCEFALOGRÁFICAS CON MATLAB 7

Autora: ÁNGELES TRILLO RAMOS

Director: D. Luciano Boquete Vázquez

TRIBUNAL:

Presidente: D. JOSÉ LUIS MARTÍN SÁNCHEZ

Vocal 1º: D. RAFAEL BAREA NAVARRO

Vocal 2º: D. LUCIANO BOQUETE VÁZQUEZ

FECHA:.....

Sin vosotros mi existencia no tiene sentido y con vosotros la eleva. Siempre en mi corazón y en mi cabeza. Rubén e Iván.

INDICE

I. RESUMEN

El presente estudio tiene como base el análisis de las respuestas de activación de fuentes neuronales denominadas ERPs - Event-Related Potentials registradas mendiante un sistema no invasivo denominado EEG- – electroencefalografía. Para tal fin se hace uso del software MATLAB 7.0 que a través de herramientas tales como TOOLBOX ICA y REDES NEURANALES se realizará un programa que procese y clasifique las señales de entrada existentes, dando como respuesta el error producido al categorizar un determinado registro EEG entre dos estados – pensamiento de acción o pensamiento de reposo.

II. ABSTRACT

The present study is based on the analysis of the activation responses of neuronal sources called ERPs - Event-Related Potentials, registered with a non-invasive system called EEG - electroencephalography. To achieve this goal, MATLAB 7.0 software is used which, through tools such as TOOLBOX ICA and NEURAN NETWORKS, it will carry out a program that processes and classifies the existing input signals, responding to the error produced by categorizing a particular EEG register between two states - thought of action or thought of rest.

III. PALABRAS CLAVE

EEG, ICA, LVQ, BSS, BCI

IV. RESUMEN EXTENDIDO

El principal objetivo de este estudio es la obtención de un sistema automático de adquisición, procesamiento e identificación de respuestas de activación de fuentes neuronales denominadas ERPs - Event-Related Potentials – Potenciales relacionados con eventos. Dichas señales son registradas a través de un sistema no invasivo llamado EEG – electroencefalografía.

Para ello se hace uso de redes neuronales artificiales, tales como el algoritmo ICA (Independent Component Analysis: Análisis de Componentes Independientes). A través de este algoritmo es posible extraer señales estadísticamente independientes, teniendo como origen de la mezcla distintas fuentes no conocidas, incluyendo ruido de fondo, es lo que se denomina "The Cocktail-Party Problem". Con ello se persigue categorizar la fuente a través de sus componentes independientes. En nuestro caso, dicha categorización se establece únicamente en dos, componentes independientes de un registro EEG de un pensamiento de ejecución de una acción determinada y componentes independientes de un registro EEG de pensamiento de reposo (inactividad sensorial y/o motora).

Para poder realizar la clasificación de una señal ERP cualquiera, se hará uso nuevamente una red neuronal denominada LVQ – Learning Vector Quantization – Aprendizaje de Cuantificación Vectorial. Para ello se realiza un aprendizaje previo de la red, introduciendo el resultado de las componentes independientes de cada uno de los registros identificados.

A través de la simulación de la red neuronal se extraen los errores producidos en la clasificación de una señal cualquiera no perteneciente al grupo de entrenamiento. Como conclusión final se indicarán los resultados obtenidos, valorando la ejecución del sistema así como la captación de las muestras de señales a analizar.

V. GLOSARIO DE ACRÓNIMOS Y ABREVIATURAS

- ✓ EEG – Electroencefalografía.
- ✓ ERP - Event-Related Potentials – Potenciales relacionados con eventos.
- ✓ BSS - Blind Source Separation - Separación Ciega de Fuentes.
- ✓ BCI - Brain Computer Interface – Interfaz Cerebro Ordenador.
- ✓ ICA - INDEPENDENT COMPONENT ANALYSIS algorithm – Algortimo de Análisis de Componentes Independientes.
- ✓ Infomax – extensión del Algoritmo ICA realizado por Bell and Sejnowski (1995).
- ✓ Red Neuronal - Modelo matemático inspirado en el comportamiento biológico de las neuronas y en cómo se organizan formando la estructura cerebral.
- ✓ Kohonen - algoritmo capaz de descubrir patrones de datos y clasificarlos en categorías.
- ✓ LVQ – Variante del modelo de Kohonen.

VI. MEMORIA

1. INTRODUCCIÓN

1.1. Planteamiento inicial

El reconocimiento de patrones realizado en este trabajo involucra diferentes etapas que a continuación detallamos:

1. Recopilación de muestras. El estudio se realizó por medio de dos colecciones de muestras EEG de dos individuos entre 30 y 35 años, completamente sanos, perfectamente entrenados en la realización de dos acciones claramente diferenciadas: pensamiento acción y pensamiento reposo. Dichos registros fueron realizados en el año 1999. Las muestras están diferenciadas, aunque no especifican el sujeto. El pensamiento de acción ejecutado es distinto en cada colección, siendo una de ellas un pensamiento de acción compleja y la otra un pensamiento de cierre del puño, siempre de una única extremidad.

2. Estudio de las herramientas necesarias para lograr extraer de las señales recogidas algún rasgo que las puedan diferenciar. En nuestro caso se basa en el Algoritmo ICA – Infomax[1].

3. Una vez extraídas las componentes independientes de las señales registradas, se introducen en una red neuronal para su aprendizaje y posterior clasificación de una señal de entrada cualquiera. Una red neuronal, tal y como su propio nombre indica, tiene como base la interconexión neuronal existente en el cerebro humano cuyas principales características son la robustez (diariamente se mueren múltiples neuronas sin perder la funcionalidad de la red), flexibilidad (se adapta a distintas situaciones) y por último puede manejar información llena de artefactos o producidos de forma inconsciente.

[1] *Variante algoritmo ICA - Anthony J. Bell and Terrence J. Sejnowski (1995)*

1.2. Objetivos

El principal objetivo del presente estudio es la implementación de un sistema interfaz cerebro-ordenador (Brain Computer Interface o BCI), analizando las señales de EEG de los usuarios del mismo. Dicho sistema posibilitaría la ejecución de determinadas acciones complejas, mediante un pensamiento de acción simple

En Julio del 2006 la revista "Nature" (Volumen 442, Número 7099 pp. 109-222) publicó dos estudios sobre los últimos avances en interfaces que podrían reemplazar o restaurar las funciones motoras en las personas con parálisis (Leigh R. Hochberg, 2006). Este estudio se basó en la inserción de una prótesis neuromotora, consistente en un sensor insertado en una zona específica del cerebro. En este caso se trataba de captar la actividad del conjunto neuronal registrada a través de una matriz de 96 microelectrodos implantada en la corteza motora primaria en un individuo con una lesión medular de tres años de antigüedad. Los electrodos del sensor grababan la información de esa zona cerebral para luego decodificarla y procesarla en un ordenador, de tal forma que la información neuronal se transformara en órdenes de movimiento para guiar a un ordenador o a una prótesis.

Nuestro procedimiento tiene el mismo principio pero eliminando el inconveniente de la inserción de una prótesis intracraneal. A modo de síntesis, la fig. 1 muestra el proceso general de un sistema BCI.

Figura 1 Diagrama de bloques que resume el funcionamiento del BCI en las fases de entrenamiento y explotación. (Coto, 2013)

1.3. Medios y datos utilizados

Debido a que las señales utilizadas son muestras de las señales captadas en los electrodos, a continuación se mostrarán los parámetros con los cuales han sido muestreados, así como las razones de la selección de los mismos.

Cuando se muestrea una señal, lo que se desea es una representación completa, es decir, mediante el proceso de interpolación conseguir que las muestras obtenidas den un aspecto similar a la señal muestreada. La selección de la frecuencia de muestreo para una señal, teniendo en cuenta que nuestra señal ha sido filtrada mediante un filtro paso bajo a 60 Hz, depende de cuán rápidamente cambie la señal con el tiempo. Debido a que la señal está claramente limitada, la reconstrucción de la señal original a partir de una secuencia de muestras uniformemente espaciadas, será posible siempre y cuando la frecuencia de muestreo sea superior a 2*fmax.

La frecuencia de muestreo utilizada es de 128 Hz. permaneciendo dentro de los límites establecidos en el teorema de muestreo descrito anteriormente. A partir de esta frecuencia en un intervalo de captura de 2 s. se obtienen 256 muestras a analizar.

Para ello los electrodos fueron adheridos mediante pasta conductora, fijados con colodión, un aislante, los cuales están formados por pequeños discos metálicos de 5 mm. de diámetro. Aplicados correctamente, las resistencias de contacto son muy bajas, por lo que no modifican en gran medida la señal pericraneal captada.

El sistema de posicionamiento de los electrodos utilizado es el denominado DIEZ-VEINTE, dando lugar a la ubicación de 21 electrodos, según el esquema de la Fig. 2:

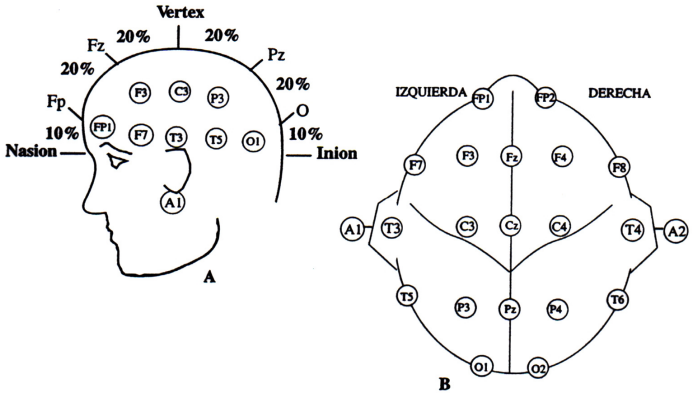


Figura 2 – Sistema Diez-Veinte. A Vista de perfil. B Vista Superior. Situación de los electrodos.

El montaje para proceder al registro de las señales EEG, parte de una serie de electrodos situados sobre la superficie del cuero cabelludo en situaciones precisas. Es necesario disponer de dos terminales. Por esto habrá que seleccionar cuáles de los electrodos deben ser la fuente de señal registrada en el electroencefalógrafo, dependiendo de número de canales disponibles y del propósito específico del registro a realizar, existiendo para tal fin registros monopolares y bipolares.

Para este estudio se selecciona el registro monopolar, el cual se basa en tomar la señal de cada uno de los electrodos independientemente de los demás. En esta situación el electrodo de registro se denomina electrodo activo y el segundo terminal de entrada al equipo se toma de un electrodo llamado de Referencia, pudiendo ser A1 o A2 (Fig. 3).

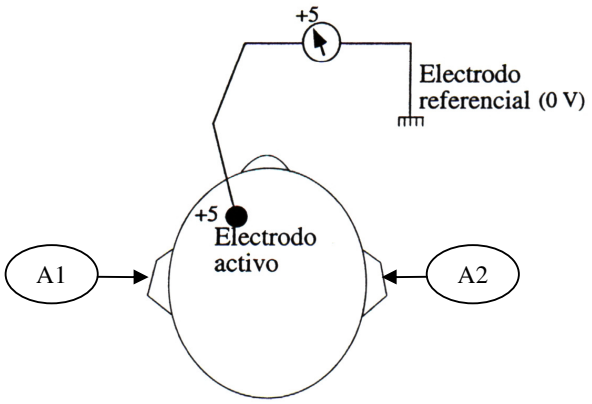


Figura 3 - Esquema eléctrico señal referencia sistema EEG

Teóricamente este electrodo debe estar situado a potencial cero, aunque esto en la práctica real nunca es posible. Por esta razón la referencia tomada es un electrodo situado en el lóbulo de la oreja.

El siguiente paso es la captación de la señal por los electrodos, para su amplificación. Un amplificador es un sistema con dos terminales de entrada y otros dos de salida. Si se introduce una señal en las entradas, tendremos a la salida otra señal igual pero mayor en intensidad.

Asimismo debemos considerar que aunque las señales cerebrales tienen un espectro de frecuencia relativamente estrecho, entre 0.2 y 100 Hz., existen otro tipo de señales, tanto biológicas como artificiales, que introducen artefactos en el registro EEG. En nuestro experimento la frecuencia de corte está en 60 Hz. teniendo en cuenta que el ruido producido por oscilaciones de las señales derivadas de los mecanismos eléctricos externos al experimento se encuentra por encima de esta frecuencia. Además también hemos de considerar otros factores inherentes al individuo objeto del estudio, como pueden ser la sudoración, movimientos cervicales, etc. los cuales producen alteraciones en las señales de medida.

Los medios utilizados en la adquisición y almacenamiento de las señales se realizaron gracias a la colaboración del Hospital Militar Gómez Ulla de Madrid. El software usado era propiedad del Hospital, empleado en diagnóstico de patologías neuronales así como trastornos de sueño.

2. ELECTROENCEFALOGRAFÍA – EEG

2.1. Historia

Fue una guerra lo que brindó la oportunidad de explorar el cerebro humano por vez primera. En 1870, Frisch y Hitzig, médicos militares del ejército prusiano, observaron que al estimular, mediante corriente galvánica, determinas áreas laterales de cerebros descubiertos de algunas de las bajas de la batalla de Sedán, se producía un movimiento en el lado opuesto del cerebro (Gross, 2007). Cinco años más tarde R. Caton confirmó que el cerebro es capaz de producir corrientes eléctricas. Fue en 1913 cuando Prawdwincz-Neminski registró lo que llamó "electroencefalograma" de un perro. Sin embargo hasta ese momento todos los experimentos realizados eran sobre cerebros descubiertos (Finger, 1994).

El verdadero avance dentro de este campo surgió en 1928 cuando un psiquiatra alemán, Hans Berger ideó un método para poder investigar la actividad eléctrica cerebral, descubriendo lo que denominó "ritmo de Berger". Años más tarde el mismo Berger pudo comprobar que dependiendo de la actividad mental del individuo este ritmo se alteraba en amplitud y frecuencia. Berger pudo medir potenciales relativos del orden de 50 a 100 microvoltios. Descubrió que estas ondas no eran del todo aleatorias ya que se percibía cierta periodicidad y regularidad. Observó que las ondas de menor frecuencia, del orden de 3 Hz. se producían durante el sueño, siendo de mayor frecuencia el resto del tiempo. Berger llegó a la conclusión de que la actividad del cerebro variaba dependiendo de la actividad del sujeto, e incluso afirmó que estas ondas podrían estar altamente afectadas en ciertas condiciones patológicas (Berger, 1929).

En 1934 otros investigadores, Adrian y Matthews, publicaron los hallazgos de Berger e identificaron ciertos ritmos dentro del registro EEG, por ejemplo, denominaron ritmo alpha a una oscilación regular de aproximadamente entre 10 y 12 Hz. desde el lóbulo occipital de la corteza craneal. Este ritmo alpha desaparecía cuando el sujeto desarrollaba cualquier tipo de atención o cuando enfocaba objetos dentro de su campo visual. Asimismo se descubrió un sistema de activación reticular, denominado RESPUESTA CEREBRAL SELECTIVA, por la cual ante

determinados impulsos se produce un estado de alerta del individuo, mientras otros son ignorados por completo.

Sin embargo a pesar de estos descubrimientos, este nuevo campo no tuvo demasiada relevancia por aquel entonces, hasta que se pudo confirmar las predicciones realizadas por Golla (1938) sobre las alteraciones de las oscilaciones rítmicas en las enfermedades. Comenzó a interesar entre los investigadores del EEG el estudio de la epilepsia y otras enfermedades mentales, poniéndose en relieve la complejidad del cerebro y la imposibilidad de aislar funciones simples, siendo estudiado como un órgano total.

2.2. Sistema Nervioso

El tejido nervioso presenta como una de sus funciones básicas, la capacidad de generar potenciales eléctricos que son la base de la excitabilidad del organismo.

No hay duda de que los sistemas nerviosos constituyen las estructuras organizadas con mayor complejidad. El sistema nervioso humano contiene entre 10^{10} y 10^{11} neuronas más un número similar de células satélites de soporte, no excitables. A lo largo del desarrollo, estas unidades se auto-organizan en formaciones interactuarte: los circuitos nerviosos que constituyen el sistemas nervioso.

La función de este sistema depende en gran medida de la actividad eléctrica de las células nerviosas excitables: las neuronas. La complejidad del sistema nervioso no descansa en un gran número de distintos tipo de señales, sino en el número y complejidad de las interconexiones con las que interactúan las neuronas. De hecho las neuronas tienen un repertorio reducido de señales eléctricas. Las neuronas a pesar de su gran número, operan con mínimas variaciones, usando sólo dos clases básicas de señales eléctricas: potenciales graduados e impulsos todo o nada.

La energía del estímulo que incide sobre las terminaciones receptoras especializadas de una neurona sensorial produce un potencial receptor, cambio de voltaje en la membrana de la célula receptora, que es graduado, es decir, modulado en amplitud, de acuerdo con la intensidad del estímulo, de manera que

los estímulos pequeños producirán pequeños potenciales receptores y los grandes estímulos producirán grandes potenciales receptores. Estos potenciales receptores persisten generalmente, con algo de atenuación, mientras se aplica el estímulo. Debido a que el curso temporal y la intensidad del potencial receptor están relacionados con el curso temporal y la intensidad del estímulo, puede decirse que el potencial receptor es un análogo eléctrico del estímulo. Este se propaga pasivamente sin regeneración propia a lo largo de las terminaciones sensoriales de la célula nerviosa, y por tanto, desaparece progresivamente al alejarse del punto de origen. Para que la información sea transmitida a lo largo de grandes distancias en el sistema nervioso central, esta debe convertirse en Potenciales de Acción puesto que estos son autorregenerativos, y de este modo se conducen sin pérdida de intensidad de la señal a lo largo de grandes distancias en las propagaciones de las neuronas denominadas axones. La cantidad de transmisor liberado y la amplitud de la respuesta postsináptica desencadenada son función de la frecuencia de los Potencias de Acción que llegan a los terminales de la neurona presináptica (Fig. 4).

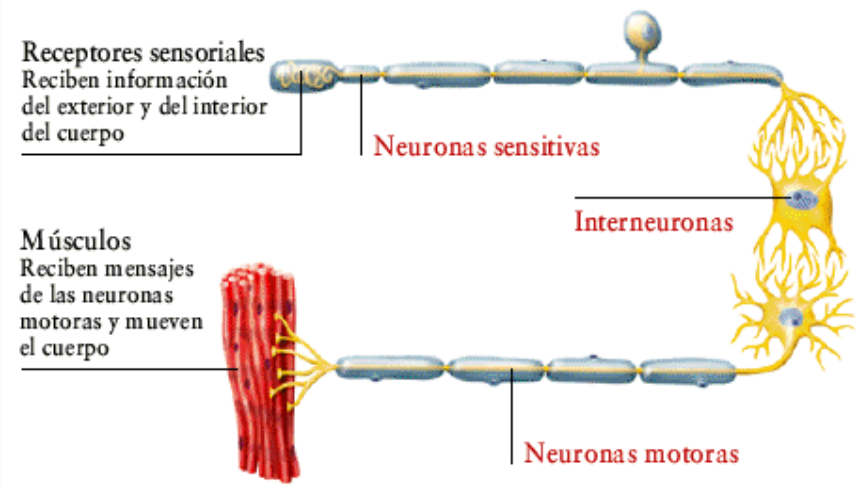


Figura 4 – Transmisión de la información desde un estímulo exterior hacia el receptor.

Dentro de ciertos límites, cuanto mayor sea la frecuencia de impulsos presinápticos, más rápida y abundante será la liberación de la sustancia transmisora y mayores serán los cambios postsinápticos del potencial. Al igual que los potenciales receptores, los potenciales postsinápticos también son analógicos, aunque no lineales y bastante distorsionados en relación al estímulo original.

Hemos de considerar que la generación de Potenciales de Acción depende en gran medida de un suceso sináptico excitador anterior que aumente la probabilidad de su iniciación. Asimismo existen otros estímulos inhibidores que dan un resultado opuesto. Es decir, la suma de ambos impulsos debe ser superior a un determinado umbral eléctrico asociado a la neurona sensorial que dé lugar a dicho potencial de Acción. Si dicho potencial se aproxima al potencial de reposo, siendo más negativo que el nivel de disparo, no se producirá corriente sináptica. Ante la estimulación de una fibra nerviosa puede ocurrir que no haya respuesta o que la respuesta sea máxima, todo o nada.

Asimismo el sistema nervioso actúa como "combinación de un todo", es decir, a nivel de una neurona aislada, ésta reacciona a la información que le llega procedente de varias señales sinápticas, de manera que producirá o dejará de producir impulsos propagados, potenciales de acción. Cada neurona integra las distintas señales sinápticas excitadoras e inhibidoras que llegan a ella. Toda actividad integradora de las motoneuronas se centra en el disparo de Potenciales de Acción, es decir excitación, o en la supresión del mismo. La información excitadora que no puede alcanzar el umbral, ya sea por ella misma o por sumación (o integración) con otras señales, se descarta (Fig. 5).

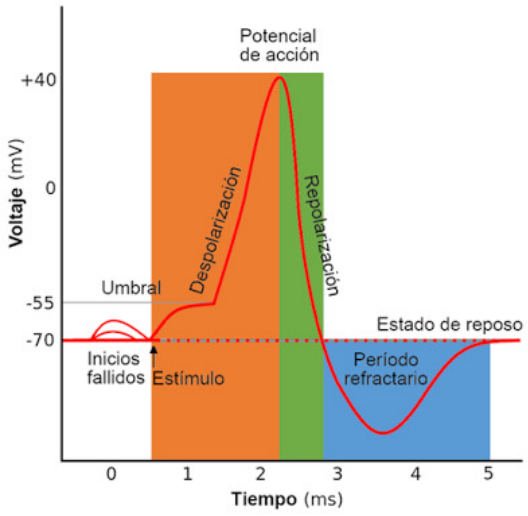


Figura 5 - Potencial de acción

Existen dos tipos de sumación de señales sinápticas, temporal y espacial. La temporal tiene lugar porque las corrientes sinápticas se originan a la vez en diferentes sinapsis y se propagan de modo aditivo hacia la zona iniciadora del

impulso. La sumación espacial, por otro lado no requiere la sumación de las corrientes sinápticas, y puede tener lugar aunque las corrientes individuales no se solapen, todo ello dependerá de las características eléctricas de las neuronas.

Por tanto hemos de tener en cuenta que en todo proceso eléctrico del sistema nervioso entran en juego numerosos factores, no pudiendo aislar en ningún momento procesos independientes ya que forman parte de un conjunto global que da lugar a la activación neuronal que promueve una acción o estado determinado.

2.3. Electroencefalograma

Un electroencefalograma o EEG, es un registro de la actividad eléctrica continua del cerebro medida por la diferencia de potencial eléctrico entre un electrodo colocado en un área específica del cuero cabelludo y un electrodo "neutral" o punto de referencia, ubicado en cualquier otro sitio del cuerpo, o la diferencia de potencial eléctrico entre pares de electrodos colocados sobre la cabeza. Los voltajes que se producen son muy débiles; en el adulto normal el valor máximo es de unos 300 microvoltios, y por lo tanto es necesario un equipo de registro extremadamente sensible. Los registros EEG representan la actividad combinada de un gran número de neuronas ubicada en la vecindad del electrodo de registro. La electroencefalografía es una herramienta útil para el diagnóstico y el control de patologías tales como la epilepsia, tumores y daño cerebrales, ya que existen patrones EEG característicos que pueden correlacionarse con ciertos niveles y tipos de actividad cerebral.

Según THE ORIGIN OF BIOPOTENTIALS (Jr, 1998), las frecuencias de las ondas cerebrales varían entre 0.5 y 100 Hz. y su valor está intrínsecamente relacionado con el grado de actividad de la corteza cerebral (Fig. 6).

TIPO DE ONDA y voltajes	FRECUENCIA	SITUACION MENTAL RELATIVA A LA QUE CORRESPONDE
DELTA 10-50 micro voltios	0,2 a 3,5 Hz	Estado hipnótico, hemisferio cerebral derecho en plena actividad, sueño profundo.
THETA 50-100 micro voltios	3,5 a 7,5 Hz	Estado de vigilia, equilibrio entre los hemisferios izquierdo y derecho, plenitud, armonía, meditacion
ALFA 100-150 micro voltios	7,5 a 13 Hz	Relajación, tranquilidad, creatividad inicio de actividad plena del hemisferio izquierdo y desconexión del hemisferio derecho, meditacion
BETA 150-200 micro voltios	13 a 28 Hz	Estado de alerta máxima, vigilante, miedo, es la situación normal cuando estamos despiertos, conduciendo, o trabajando en donde estamos en estado de alerta, ansiedad.
RAM-ALTA +200 micro voltios	+ de 28 Hz	Estado de stress y confusion.

Figura 6 Clasificación principal de señales cerebrales. Individuo adulto sano.

Por su frecuencia, las señales de EEG se clasifican en las siguientes bandas: ALFA, BETA, DELTA y GAMMA o RAM-ALTA.

Las ondas alfa son irregulares, con una frecuencia aproximada entre 8 y 13 Hz., registradas en la parte posterior de la cabeza. Las ondas alfa habitualmente se producen durante periodos de relajación, en la mayoría de las personas son notables únicamente cuando los ojos están cerrados. Aproximadamente dos tercios de la población tiene ondas alfa que son alteradas por la atención. Del tercio restante, la mitad casi no presenta ondas alfa y la otra mitad tiene onda alfa persistente que no son fácilmente alteradas por la atención. En la figura 7 se presentan dos registros de dos individuos jóvenes estudiantes con y sin ritmo alfa, a) y b) respectivamente.

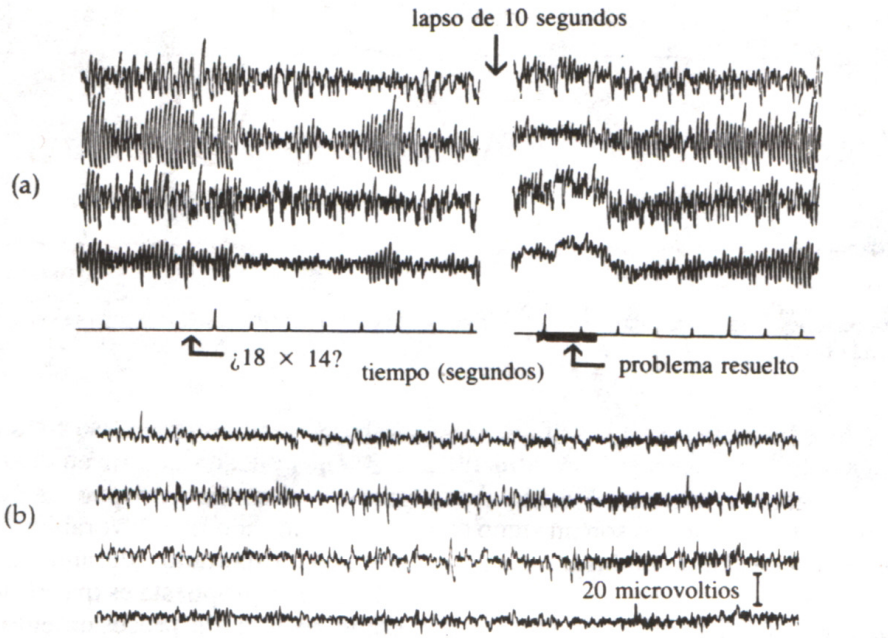


Figura 7 (a) Ejemplo de registros de las ondas cerebrales de un estudiante con notable ritmo alfa. Los electrodos fueron colocados en cuatro posiciones en la cabeza. Según este experimento cuando se le pidió al estudiante que multiplicará 18x14, el ritmo alfa desapareció para luego reanudarse una vez resuelto el problema (10 segundos del registro fueron emitidos en el presente esquema, para registrar las zonas divisorias de la acción). b) Ondas cerebrales de un estudiante con ausencia completa de ritmo alfa. De acuerdo con estudios recientes, las personas que casi carecen de ritmos alfa bajo cualquier condición piensan casi exclusivamente mediante imaginación visual, mientras que las personas con ritmo alfa persistente tienden a ser pensadores abstractos.

Las ondas beta son de menor amplitud que las ondas alfa, pero su frecuencia es mayor, y oscila entre 13 y 28 Hz . Las ondas beta aparecen en descargas y se asocian a la actividad mental y con la excitación. Las ondas GAMMA son ondas con frecuencias superiores a 28 Hz. y corresponden a estados de ansiedad y pánico.

Ondas de menor frecuencia que las alfa, denominadas ondas delta, se observan normalmente en lactantes o en adultos durante el sueño. Las ondas delta en adultos despiertos son un signo de alteración mental o de daño cerebral (Jr, 1998).

Por último están las ondas theta que se encuadran entre los 4 y los 7 Hz. siendo características dentro de la edad infantil en las regiones parietal y temporal, aunque también suceden en estados de estrés en algunos adultos, particularmente durante periodos de frustración o desencanto.

Una vez determinadas las frecuencias de las distintas señales EEG posibles, y considerando el grado de actividad cerebral a estudio, se considerarán los rangos de frecuencia entre 8 y 30 Hz. subdividiendo el rango beta en dos franjas de frecuencia I Y II, de 13 a 18 Hz. y de 18 a 30 Hz respectivamente. Este tipo de señales ser registran habitualmente en las regiones parietal y frontal. Las ondas Beta II, aparecen durante la activación intensa del sistema nervioso central y durante la tensión.

Como se puede apreciar, los diferentes ciclos evaluados de las señales a estudio se corresponden directamente con el estado de sujeto en el momento de su representación. Por tanto es de vital importancia poder encuadrar las señales dentro de estos patrones.

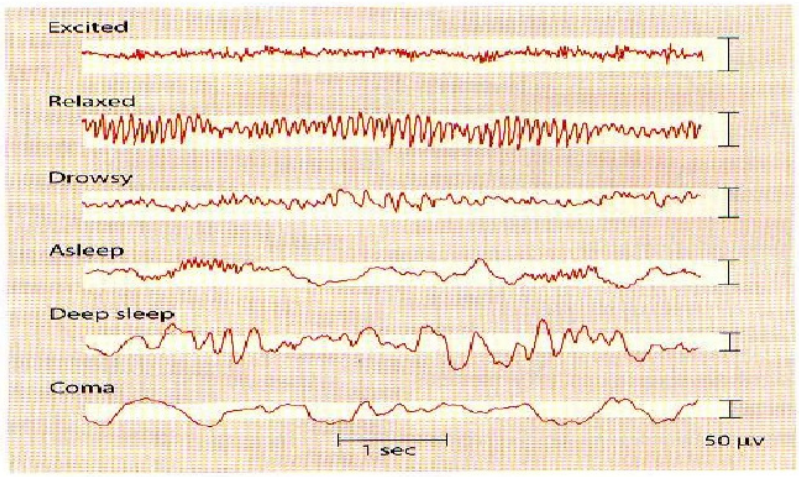


Figura 8 – Cambios del EEG en los distintos estados de conciencia

Es posible establecer el estado del sujeto en el momento de la observación, tal y como muestra la figura 8, aunque es un estado generalizado, es decir excitado, relajado, adormecido, durmiendo profundamente e incluso averiguar si padece algún daño cerebral o mental, como se puede comprobar en la Figura 8 – último registro y Figura 9, pero sin concretar su grado.

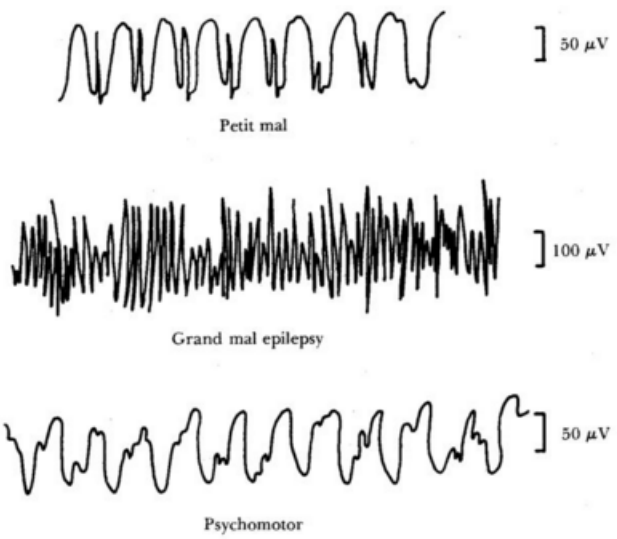


Figura 9 Representación de ondas de Registros EEG anormales en diferentes tipo de epilepsia. (A.C.Guyton, 1972)

En la actualidad se está avanzando más allá del terreno médico, llegando incluso a introducirse dentro del campo de la *percepción extrasensorial*. El próximo paso es poder encontrar un modelo que pueda procesar toda señal cerebral y corresponder exactamente con una acción o percibir de alguna forma lo que "pasa por nuestras mentes".

2.4. Imaginación mental dentro de un contexto de acción

Las observaciones registradas en nuestro estudio se basan en el concepto de realizar una acción sin ser ejecutada, es decir, utilizando para ello la imaginación.

En muchas ocasiones los medios técnicos utilizados no posibilitan una buena observación, siendo este el caso. Los sistemas telemétricos son utilizados para el registro de señales EEG en individuos en movimiento, evitando de este modo posibles errores ocasionados por el ruido producido por el movimiento del cableado a él conectado y con origen en el sistema de registro. Con ello se consigue registrar las señales emitidas por el cerebro en el momento de realizarse una acción determinada, mandando dicha señal a un sistema receptor el cual se encargará de su procesamiento y análisis posterior.

Sin embargo viendo la imposibilidad de utilizar este tipo de medida, nos apoyaremos en el estudio realizado por el investigador M. JEANNEROD (JEANNEROD, 1997). La idea general es que el acto de imaginar es una parte del fenómeno típico relativo a la intención y la preparación hacia un movimiento. De acuerdo con esta definición, las imágenes motoras están estrechamente relacionadas con las propiedades de la representación motora, es decir, se tiene la misma relación funcional para la imaginación que para la ejecución del movimiento, y por supuesto la misma causa.

La imaginación motora requiere una representación del cuerpo como el generador de las fuerzas que lo activan y no únicamente los efectos de esas fuerzas en un mundo externo.

La primera hipótesis es que la simulación del movimiento es lo mismo que realizarlo, teniendo en cuenta que la ejecución del mismo es bloqueada. Consecuentemente, el hecho de que la actividad muscular está sólo parcialmente bloqueada durante la simulación motora revela el delicado equilibrio entre las influencias de excitación e inhibición en los niveles de las motoneuronas, responsables de la propagación de las señales que originan el movimiento, lo cual sugiere que las motoneuronas están próximas a un umbral determinado que no es alcanzado.

El hecho de que la simulación mental del movimiento activa una salida motora, fue confirmado por un reciente estudio de la excitabilidad de la espina dorsal durante la imaginación motora. Un grupo de individuos fue instruido para realizar un experimento. Se basaba en realizar dos procesos, uno motor y otro mental de la misma acción. Se percibió que en procesos de simulación mental la amplitud de los reflejos monosinápticos espinales se incrementaba. Otros efectos normalmente no sujetos a control voluntario fueron observados, tales como la velocidad del corazón. Después de unos pocos segundos del ejercicio, la respiración y los latidos del corazón se incrementaban hasta un 50% por encima del valor de reposo. Igualmente en la condición mental este incremento era producido, aunque su valor estaba por encima del 32%. Esto revela que en una fracción amplia de este incremento en el principio del ejercicio debe ser debido al efecto de la preparación motora no por los cambios metabólicos producidos al ser ejecutada.

La preparación de la actividad muscular, juega un importante papel en la consecuencia del movimiento lo cual implica que las representaciones motoras para la actividad y para la imaginación son la misma esencia y por tanto representan lo mismo y que estos dos modos de representación son solamente distinguibles por las circunstancias que las genera.

Estableciendo de esta forma un patrón de acción se puede afirmar que la imaginación de dicho patrón nos ofrece las mismas características que las señales emitidas en su ejecución motora, pudiendo por tanto ser objeto del mismo estudio y por tanto afirmar que su análisis puede extrapolarse a esa situación de movimiento.

2.5. Resumen

Según lo visto hasta este punto se pueden establecer los siguientes criterios:

1. Toda señal cerebral viene caracterizada en amplitud y frecuencia, dependiendo de la actividad mental del individuo. Estas ondas no son del todo aleatorias percibiéndose cierta periodicidad y regularidad, pudiendo por tanto ser analizadas y tipificadas.

2. La complejidad del sistema nervioso tiene como base principal el número y composición de las interconexiones con las que interactúan las neuronas y no en los distintos tipos de señales. De hecho las neuronas tienen un repertorio reducido de señales eléctricas, usando sólo dos clases básicas de señales: potenciales graduados e impulsos todo o nada.

3. La información transmitida por una neurona sensorial se convierte en un Potencial de Acción autorregenerador, pudiendo de este modo conducir sin pérdida de intensidad de la señal a lo largo de grandes distancias en las propagaciones de las neuronas denominadas axones. Por tanto la información captada por los electrodos pericraneales es una atenuación leve de este tipo de señales, principales promotores de una actuación determinada.

4. El sistema nervioso actúa como "combinación de un todo", es decir, a nivel de una neurona aislada ésta reacciona a la información que le llega procedente de varias señales sinápticas, de manera que producirá o dejará de producir impulsos propagados, Potenciales de Acción. Cada neurona integra las distintas señales sinápticas excitadoras e inhibidoras que llegan a ella. Toda actividad integradora de las motoneuronas se centra en el disparo de Potenciales de Acción, es decir excitación, o en la supresión del mismo. La información excitadora que no puede alcanzar el umbral, ya sea por ella misma o por sumación con otras señales, se descarta. El resultado de toda actividad eléctrica es el conjunto de activaciones individuales que dependiendo de un determinado umbral darán lugar a la propagación de la señal de acción o bien a su inhibición.

5. El estudio se apoya en el concepto de la existencia de dos patrones diferenciados de ritmos electroencefalográficos ALFA y BETA. Las ondas ALFA, con rango de 8 a 13 Hz, son producidos durante periodo de relajación, estando presentes en el estado de reposo. Son ondas muy irregulares de baja frecuencia. Por otro lado está el ritmo BETA de menor amplitud pero de frecuencia mayor, que se relacionan con la actividad mental y con la excitación.

6. El sistema de imaginación motora, generador de señales EEG será considerado como lineal e invariante en el tiempo, pudiendo por tanto suponer que las señales de salida se obtienen como la convolución de un determinado número de señales de entrada, invariantes en tiempo y forma. Igualmente hemos de considerar la causalidad del sistema, si no existe estímulo no es posible generar potencial eléctrico en la salida.

7. Previamente la señal se filtra mediante un filtro paso bajo cuya frecuencia de corte se establece en 60 Hz. evitando por tanto cualquier ruido derivado de los aparatos eléctricos de medida. Seguidamente se realiza el muestreo de la señal registrada para su posterior procesamiento digital, para lo cual es tenido en cuenta el teorema de muestreo estableciendo una frecuencia de muestreo de 128 Hz, cayendo dentro de los márgenes establecidos en el mismo. Se obtienen por tanto 256 muestras de la señal de origen cuyo rango de tiempo se establece de 0 a 2 sg.

8. La simulación del movimiento es lo mismo que realizarlo, a nivel neuronal, teniendo en cuenta que la ejecución del mismo es bloqueado, en resumidas cuentas, se tiene la misma relación funcional para la imaginación que para la ejecución del movimiento, y por supuesto la misma causa. Los registros almacenados son originados por la imaginación de una determinada acción siendo igualmente válidos que si fueran realizados mediante movimientos motores.

3. SEPARACIÓN CIEGA DE SEÑALES

3.1. Introducción a la BSS

Normalmente la deconvolución se basa en el hecho de recuperar una señal de excitación, dando una respuesta de un operador lineal conocido, de esa entrada invariante en el tiempo. En nuestro caso la deconvolución se discute como un reto mayor de lo anterior, teniendo en cuenta que el operador lineal involucrado en la convolución de las fuentes no se conoce – hablamos de la SEPARACIÓN CIEGA DE SEÑALES – BSS (BLIND SEPARATION SIGNALS).

En una aplicación típica de procesamiento de señales, la base se establece en extraer información contenida en un conjunto de datos obtenidos experimentalmente designados por X{n}. Normalmente los elementos de estos datos se suelen interpretar como muestras de una secuencia de variables aleatorias fundamentales. Estas variables aleatorias son los instrumentos matemáticos destinados a representar los resultados de un determinado experimento, como es nuestro caso. Como cada suceso tiene asignado un valor de la variable aleatoria, podremos hablar de que dicho valor asignado es una probabilidad y su asignación se representa mediante la densidad de probabilidad. En resumen, el índice de la variable, n, toma exclusivamente un valor entero y está frecuentemente asociado con el tiempo. Así pues una secuencia discreta de variable aleatoria se referencia a menudo como series de tiempo aleatorio. Dentro del lenguaje de la teoría de probabilidad, el elemento n de la secuencia de los datos, se interpreta como una muestra de la variable aleatoria asociada. Asimismo para simplificar el estudio esta variable únicamente toma valores reales.

En la mayoría de los resultados importantes del proceso de señales, se establece que en series de tiempo aleatorio la señal es estacionaria, es decir, invariante con el tiempo. La suposición estacionaria más básica es que cada elemento aleatorio de la serie de tiempo aleatorio está gobernada por la misma función de densidad de probabilidad. Una serie de tiempo aleatorio de esta forma caracterizada se dice que es estacionaria de orden uno y puede ser interpretada como una secuencia de muestras de una variable aleatoria fundamental.

Dependiendo de la naturaleza del fenómeno representado, estas muestras podrían ser estocásticamente dependientes o independientes.

Esto indica que la suposición de la estacionariedad de primer orden no constituye una relación estadística entre los elementos de la serie de tiempo aleatorio, es decir, no establece si las muestras son independientes o no.

Así pues, la variable aleatoria fundamental simplemente podría ser referenciada como un generador de variable aleatoria para las series de tiempo aleatorio.

Para poder imponer un alto nivel de invarianza en el tiempo en una serie de tiempo aleatorio, es necesario introducir órdenes de más alto nivel de estacionariedad. Por ejemplo, la serie de tiempo aleatorio $\{X(n)\}$ se dice que es estacionaria de orden dos si la densidad de probabilidad de la unión del par de variables aleatorias $\{X(n_1),X(n_2)\}$ y $\{X(n_1+m),X(n_2+m)\}$ es la misma para toda selección de los enteros m, n_1 y n_2. Así pues, la naturaleza estadística de dos elementos cualquiera, de una serie de tiempo aleatoria de estacionalidad de orden dos, es dependiente únicamente de su diferencia de tiempo n_2-n_1 y no de sus valores específicos de n_1 y n_2, es decir, su frecuencia de aparición puede ser la misma pero no los valores a ella asociada. Una serie de tiempo aleatoria estacionaria de orden dos es también estacionaria de orden uno, aunque lo contrario necesariamente no es cierto. Se podría extender este concepto determinando que una serie de tiempo aleatorio es estacionaria de orden k si la función de densidad de la unión del conjunto $(X(n_1),X(n_2),....,X(n_k))$ y $(X(n_1+m),X(n_2+m),....,X(n_k+m))$ son lo mismo para toda selección de enteros $n_1,n_2,...,n_k$ y m.

Teniendo en cuenta un orden estadístico concreto es posible encuadrar la señal dentro de unas características esenciales para el planteamiento posterior en la ejecución del algoritmo que dé lugar a la separación de las muestras obtenidas.

He aquí la importancia de determinar la estacionariedad de las señales a estudio ya que de lo contrario no se podría encontrar ningún criterio óptimo para el desarrollo del algoritmo utilizado en la experimentación.

3.2. Deconvolución y separación ciega

En muchas aplicaciones, se trata de identificar la naturaleza básica de un fenómeno haciendo medidas empíricas de ese fenómeno. Debido a factores como la dinámica de la instrumentación utilizada, los efectos del entorno y otras consideraciones, estas medidas normalmente toman la forma de un convolución lineal invariante en el tiempo de una señal o señales.

Se asume que estas medidas son una muestra de una serie de tiempo aleatorio gobernado por una operación de convolución lineal invariante en el tiempo:

$$X\ (n) = \sum_{k \in K} f_k W\ (n - k)$$

Donde {X(n)} es la serie de tiempo de la muestra obtenida, {W(n)} es la señal estacionaria de interés a extraer, mientras que el número de secuencia f(k) designa la respuesta del impulso unidad de la distorsión producida por la operación de convolución. El índice establecido, K, designa al conjunto de enteros sobre los cuales la operación de convolución se ha definido. Este índice puede ser finito o infinito en tamaño, dependiendo de la naturaleza de la operación de convolución lineal.

.

En este desarrollo, el índice K establecido asociado con cualquier operación lineal de tiempo invariante está implicado en el resultado, aunque no es conocido. En cualquier caso, la operación lineal es invariante en el tiempo, por lo que sus coeficientes no depende de él. La serie de tiempo aleatoria es estacionaria y como consecuencia ninguna de sus propiedades estadísticas son afectadas por un desplazamiento sobre el eje de tiempos.

Ya que lo que se desea es poder reconstruir la señal de excitación a través de las muestras recogidas, estas dos señales deben estar relacionadas por una operación lineal invariante en el tiempo. Esto requiere que la deconvolución debe tomar la forma de una operación lineal invariante en el tiempo, tal y como se muestra a continuación:

$$Y(n) = \sum_k g_k X(n - k)$$

Donde $\{g_k\}$ es la respuesta de impulso unitario del operador de deconvolución. La unión del conjunto de índices de este operador de deconvolución es generalmente diferente de la convolución.

Por ejemplo, si el conjunto de índices del operador de convolución es finito y contiene más de un entero entonces el conjunto de índices del operador de deconvolución podría contener un número infinito de enteros. Un operador ideal de deconvolución está definido como aquel cuya repuesta de serie de tiempos aleatorios deconvolucionados es una escala y una imagen retardada de la señal de excitación. La fig. 10 representa esquemáticamente el resultado de la operación, tal que W(n) sería el origen e Y(n) el resultado obtenido, después del proceso de deconvolución de las muestras X(n) existentes.

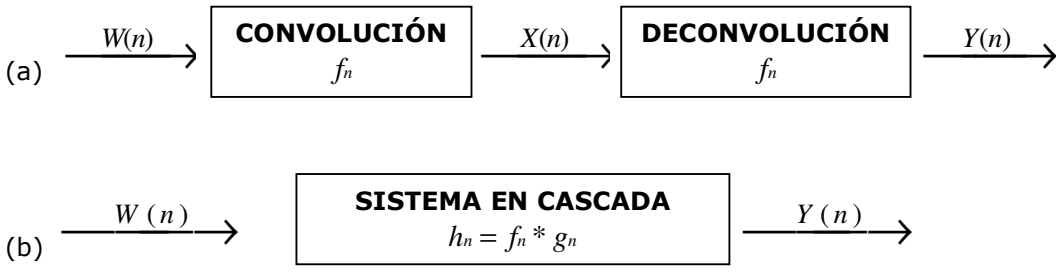


Figura 10 (a) Operación Convolución-deconvolución (b) Sistema equivalente.

En otras palabras, en nuestro sistema X(n) está relacionado con la matriz de datos procedentes de los electrodos pericraneales y lo que intentamos averiguar es Y(n), las señales resultantes de la deconvolución.

3.3. Momentos

En principio se podría establecer que una serie de tiempo aleatoria se puede generar por una secuencia de muestras estadísticamente independientes o dependientes de una variable aleatoria fundamental, designada como X.

La función de densidad de probabilidad $f_x(x)$ que gobierna esta variable aleatoria generada X, puede ser interpretada como una descripción matemática de cómo la unidad de masa se distribuye a lo largo del eje x. Si esta masa se distribuye de una forma continua entonces X se denomina variable aleatoria continua y $f_x(x)$ es una función continua de x.

Por otro lado, si la unidad de masa está localizada en un número finito o computacionalmente finito de puntos en el eje x, X se dice que es una variable aleatoria discreta y $f_x(x)$ está compuesta de suma de funciones Delta de Dirac ponderadas. Cuando la masa está distribuida tanto en forma continua como discreta, la variable aleatoria es una mezcla y la función de densidad de probabilidad asociada contiene tanto componentes continuas como delta de Dirac.

Cualquiera que sea la naturaleza de la variable aleatoria, es posible resumir la manera con la que la masa unitaria se distribuye utilizando un conjunto de parámetros discretos llamados momentos. En particular el momento de orden nth de la variable aleatoria X se expresa del siguiente modo:

$$E\{X^n\} = \int_{-\infty}^{\infty} X^n f_x(x)\,dx \qquad \text{para n = 1,2,3,.....}$$

Si el momento nth existe también existirán los anteriores. El momento de primer orden es el más importante y se denomina valor medio y corresponde con el centro de gravedad de la distribución de masa unitaria.

El momento de segundo orden es la varianza, el cual proporciona una medida de cómo se dispersa la masa en las cercanías del centro de gravedad.

El momento de tercer orden es usado para medir la simetría de la función de densidad sobre el valor medio. Por ejemplo si la simetría tiene valor cero indica que la función es simétrica sobre el eje del valor medio.

El cuarto es a menudo usado para medir el exceso o llanura de la función de densidad de probabilidad sobre su media y se denomina KURTOSIS.

En nuestro caso, el registro EEG contiene información de cambios visibles en los potenciales eléctricos de cerebro obtenidos desde una serie de electrodos. Los datos incluyen las características de las ondas con acompañamiento de variaciones en amplitud, frecuencia, fase,.... En el cómputo de las distribuciones en amplitud del EEG, por ejemplo sucesivas amplitudes EEG, deben ser medidas y ordenadas dentro de unas clases de amplitud específicas. Los histogramas en amplitud que resultan de estos procesos son a menudo simétricos, esencialmente distribuciones gaussianas.

Las principales características de la distribución gaussiana son simplemente especificaciones de sus medias y desviaciones estándar. Con respecto a los momentos de control más altos de la distribución, tales como la asimetría y la kurtosis, éstos son igual a cero. Sin embargo, en una distribución no gaussiana las medidas de la asimetría y kurtosis asumen valores no cero y pueden ser usadas para caracterizar la distribución particular de la amplitud. Las cuatro medidas estadísticas usadas para caracterizar un histograma de amplitudes EEG incluye la media, la amplitud estándar, la asimetría y la kurtosis.

Ya que la suma de los potenciales EEG es normalmente del orden de pocos microvoltios cuando el análisis es demasiado corto, la media es esencialmente constante, aunque de pequeño valor. Cualquier elevación en valores de la media, son indicativos de cambios en potencial que son de origen técnico, tales como corrientes del amplificador.

La varianza de la distribución de las amplitudes EEG está directamente relacionada con la potencia total de EEG. Por ejemplo, un EEG plano proporcionaría unos valores bajos de varianza, mientras que amplias oscilaciones de EEG producirían altos valores de varianza.

El grado de desviación desde la simetría de una distribución normal o gaussiana es medido por la asimetría. Este tercer momento central del histograma de amplitudes tiene un valor cero cuando la distribución es completamente simétrica y asume algún valor no cero cuando las ondas EEG tiene asimetrías con respecto a la línea base, característico en algunos patrones de características de sueño, ritmos musculares,. En general un valor no cero en el índice de asimetría refleja la presencia de eventos monofásicos en las ondas.

Otro de los momentos estadísticos más importantes en la caracterización de las señales aleatorias, es el cuarto momento, denominado Kurtosis, el cual nos da una idea de la variabilidad de la señal. Una Kurtosis negativa nos indica un alejamiento mayor de los valores con respecto a la media; en cambio una Kurtosis positiva no implica un mayor número de datos próximo a la media. Un valor cero implica una distribución tipo normal, distribución normal de Gauss (Fig.11).

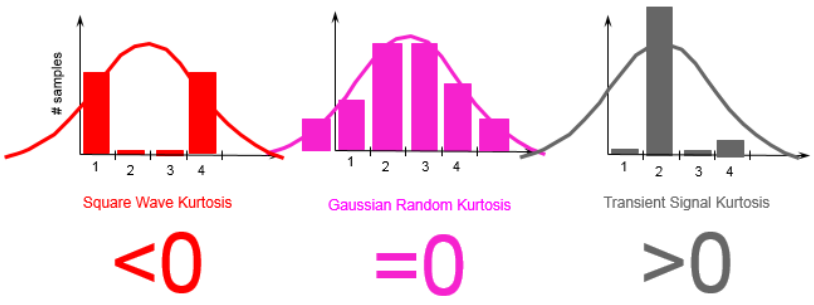


Figura 11 Función de distribución según el valor de la Kurtosis

El registro EEG ha sido estudiado extensamente usando procesos de señales, la mayoría de los cuales están basados en la suposición de que EEG es un proceso lineal gaussiano. Aunque el análisis lineal es computacionalmente eficiente y útil, ya que sólo utiliza información retenida en la función de autocorrelación, hasta el cumulante de segundo orden, la información adicional almacenada en cumulantes de más alto orden es a menudo ignorado por el análisis lineal de EEG.

Existen evidencias de que la distribución de amplitud de EEG a menudo se desvía del comportamiento gaussiano. Ha sido mostrado, por ejemplo, que el EEG humano involucrado en un proceso aritmético mental muestra comportamiento no gaussiano. En suma, el grado de desviación del comportamiento gaussiano de EEG

se ha comprobado que depende de estado de conducta. En el estado del sueño se muestran menos señales gaussianas que en acción.

3.4. Maximización de la información – principio de Infomax

Tanto la separación como la deconvolución ciega están relacionadas con el aprendizaje no supervisado. En la separación ciega un conjunto de fuentes, $s_1(t)...s_n(t)$ es mezclado linealmente en conjunto mediante una matriz A. No se conoce nada sobre las fuentes, o el proceso de mezcla. Todo lo que se recibe son las N superposiciones de ellas, $x_1(t)....x_n(t)$. La matriz W es una permutación de la inversa de la matriz desconocida A.

En la deconvolución ciega, un señal desconocida s(t) es convolucionada mediante un filtro desconocido, $a_1...a_L$, dando una señal corrupta $x(t)=a(t)*s(t)$ donde a(t) es la respuesta impulsiva del filtro. La tarea es reconstruir s(t) por deconvolución de x(t) con un filtro resultado del aprendizaje $w_1,....,w_L$ que realiza la operación inversa del filtro.

Ambos problemas tienen su dificultad. En el caso de la separación ciega, la aproximación es generalmente asumir que las fuentes S, son estadísticamente independientes y no-gaussianas, entonces el problema de aprendizaje W se convierte en un problema de análisis de componentes independientes. La condición de que las señales sean no gaussianas es debido a la interpretación de su independencia, ya que dicha medida es realizada mediante sus momentos y cumulantes cruzados, y cuando dos variables aleatorias son independientes, sus cumulantes cruzados de cualquier orden se anulan. Como toda fuente gaussiana sus cumulantes de orden superior a dos son cero, da lugar a la incertidumbre de análisis de independencia.

El problema aparece cuando, en el caso de deconvolución ciega, la aproximación es a menudo asumir que la señal original s(t) consiste en símbolos independientes, entonces la deconvolución llega a ser un problema de trasladar desde x(t) cualquier dependencia estadística a través del tiempo, introducido por el filtro corrupto A. Este proceso a menudo se denomina preblanqueo de x(t).

Técnicas de separación ciega pueden ser utilizadas dentro de cualquier dominio donde un array de N receptores, en nuestro caso electrodos, reciben N señales de entrada mezcladas linealmente.

La aproximación que se toma de estos problemas es una generalización del principio infomax de Linsker para unidades no lineales con entradas distribuidas arbitrariamente incorruptas por cualquier fuente de ruido. El principio es descrito por Laughlin (1981):

"Cuando las entradas x son filtradas mediante una función sigmoidal g(x) para dar una salida y, es posible obtener la máxima información transmitida de las entradas a las salidas cuando la parte de la pendiente de la sigmoidal está alineada totalmente con la parte de más alta densidad de las entradas f(x)".

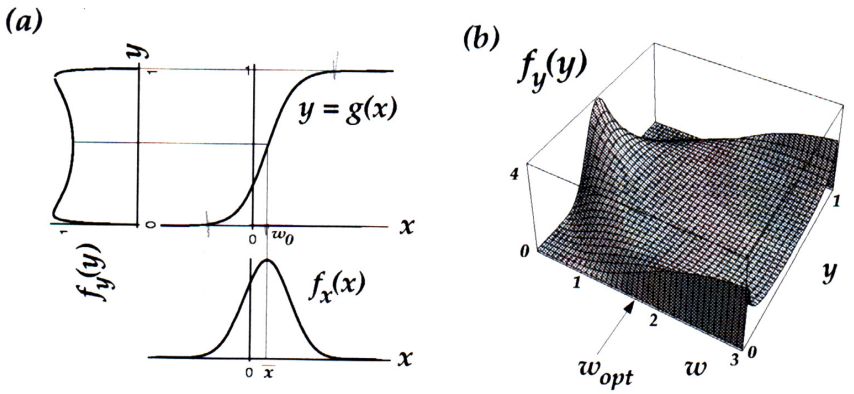


Figura 12 (a) y (b) Ilustran la característica de la alineación de la función de transferencia sigmoidal para la obtención de la información óptima.

La generalización de este hecho es un sistema que maximizando la transferencia de información, reduce la redundancia entre las unidades en la capa de salida, este sistema es también llamado ICA – ANALISIS DE COMPONENTES INDEPENDIENTES.

El problema básico es cómo se puede maximizar la información mutua que la salida Y contiene sobre la entrada X. Esto mismo se define del siguiente modo:

$$I(Y,X) = H(Y) - H(Y \mid X)$$

donde H(Y) es la entropía de la salida, mientras que H(Y|X) es cualquier otra entropía de la salida que no proviene de la entrada. Definimos entropía como el grado de desorden de un sistema dado, por tanto, en el supuesto de no tener ruido, lo cual es bastante extraño, el planteamiento entre X e Y es determinista, es decir, todas las probabilidades son cero excepto un valor que será uno, y por tanto el valor de la entropía H(Y|X) tiene el valor más bajo posible, y diverge a $-\infty$. Esta divergencia es una de las consecuencias de la generalización de la información teórica para variables continuas. H(y) es realmente la entropía diferencial de Y con respecto a alguna referencia, tales como el nivel de ruido o la precisión de la discretización de las variables en X e Y. En este caso es considerado únicamente el gradiente de la cuantificación de la información teórica con respecto del parámetro w de la red utilizada en el algoritmo de separación.

$$\frac{\partial}{\partial w} I(X,Y) = \frac{\partial}{\partial w} H(Y)$$

Hemos de tener en cuenta que H(Y|X) no depende de w y por tanto no toma parte de la ecuación diferencial anterior. Se puede comprobar esto mismo considerando un sistema que permite valores infinitos: Y=G(X)+N, donde G es alguna transformación invertible y N es el ruido sumado a las salidas. En este caso H(Y|X) = H(N). Cualquiera que sea el nivel de este ruido, la maximización de la información mutua, I(Y,X), es equivalente a la maximización de la entropía de salida, H(Y), ya que $\frac{\partial}{\partial w} H(N) = 0$.

Considerando un vector de entrada X, una matriz de pesos W, un vector de inicio W_o y un vector de salida $Yx=g(Wx+W_o)$, la función de densidad de probabilidad multivariable Y puede ser escrita de la siguiente forma:

$$f_y(y) = \frac{f_x(x)}{|J|}$$

donde |J| es el valor absoluto del Jacobiano de la transformación. El Jacobiano es el determinante de la matriz de las derivadas parciales:

$$J = \det\begin{bmatrix} \frac{\partial y1}{\partial x1} & \cdots & \frac{\partial y1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial y_n}{\partial x1} & \cdots & \frac{\partial y1}{\partial x_n} \end{bmatrix}$$

para unidades sigmoidales, y=g(u), u = Wx+w_o, siendo g la función lógica: $g(u) = (1 + e^{-u})^{-1}$, el resultado del aprendizaje es el siguiente:

$$\Delta W \propto \left[W^T\right]^{-1} + (1 - 2y)x^T$$
$$\Delta w_o \propto (1 - 2y)$$

excepto que ahora x, y, w_o y 1 son vectores (1 – vector de unos), W es una matriz.

Así pues considerando en nuestro sistema y1 e y2 salidas, canales, la entropía de la unión puede ser considerada como:

H(y1,y2) =H(y1)+H(y2)-I(y1,y2)

Por tanto maximizar la entropía de la unión consiste en maximizar las entropías individuales y minimizar la información mutua. Cuando esta información mutua es cero podremos asegurar que las dos variables son estadísticamente independientes y la función de densidad de probabilidad puede ser factorizada:

$$f_{y1y2}(y1, y2) = f_{y1}(y1) f_{y2}(y2)$$

Este proceso es conocido como código de aprendizaje factorial (Barlow, 1989), minimización predecible (Schmidhuber, 1992), INDEPENDENT COMPONENT ANALYSIS (Comon, 1994) y reducción redundante (Barlow, Redundancy reduction revisited, 2000).

3.5. Resumen

Como se puede comprobar en el presente capítulo se ha intentado poner en relieve el problema de la separación y la deconvolución ciega.

La importancia de este tema estriba en poder establecer los criterios por los cuales se basa el algoritmo utilizado en nuestro estudio. Como más adelante se podrá comprobar esté algoritmo, desarrollado por Bell & Sejnowski establece su principio en el Análisis de Componentes Independiente –ICA–. Asimismo previamente se ha realizado un proceso de preblanqueo de la señales registradas obteniendo de esta forma una aproximación mayor de las fuentes a separar.

Toda separación de señales desconocidas debe tener las siguientes consideraciones para poder de esta forma ser procesadas y estudias posteriormente:

1. La interpretación de los datos obtenidos en las distintas muestras son consideradas como una secuencia de variables aleatorias fundamentales. Ya que el índice de la variable, n, está asociado con el tiempo, la secuencia discreta de la variable aleatoria es denominará series de tiempo aleatorio.

2. Los resultados más óptimos en diversos estudios están basados en la suposición de que en series de tiempo aleatorio la señal es estacionaria y por tanto invariante en el tiempo.

3. La siguiente hipótesis es asumir que las muestras de la serie de tiempo aleatorio se procesan a través de una operación de convolución lineal invariante en el tiempo

4. Cualquiera que sea la naturaleza de la variable aleatoria, es posible resumir la manera con la que la masa unitaria, función de densidad de probabilidad, se distribuye utilizando un conjunto de parámetros discretos llamados momentos: la media, la varianza, la simetría y la kurtosis.

5. El registro EEG incluye las características de las ondas con acompañamiento de variaciones en amplitud, frecuencia y fase. Los histogramas en amplitud son a menudo simétricas, esencialmente distribuciones gaussianas. Las cuatro medidas estadísticas usadas para caracterizar una histograma de amplitudes EEG incluye la media, la amplitud estándar, la asimetría y la kurtosis.

6. El algoritmo utilizado en el ANÁLISIS DE COMPONENTES INDEPENDENTES está basado en la maximización de la entropía de la unión que consiste en maximizar las entropías individuales y minimizar la información mutua. Cuando esta información mutua es cero es posible asegurar que las dos variables son estadísticamente independientes.

7. En la separación ciega se asume que las fuentes S, son estadísticamente independientes y no-gaussianas. Entonces el problema de obtención de la matriz W, la cual realiza la operación inversa de la convolución, se convierte en un problema de análisis de componentes independientes – ICA.

8. Esta suposición puede ser corroborada utilizando el cuarto orden estadístico, en el cual se determina qué señales son **SUPER-GAUSSIANAS**, y por tanto independientes, siendo aquellas cuya KURTOSIS sea mayor que cero. Esta verificación se realiza una vez obtenidas las componentes "supuestamente" independientes.

4. ICA – INDEPENDENT COMPONENT ANALISYS

4.1. Introducción

ICA es un procesamiento de señales cuya definición más simple puede describirse como un método para separar N entradas estadísticamente independientes que han sido mezcladas linealmente en N canales de salida, sin profundizar ni conocer sus distribuciones o dinamismo. También es denominado como un problema de SEPARACIÓN CIEGA, descrito en el capítulo 3.

Para resumir lo anteriormente descrito se muestra la siguiente ilustración fig. 13:

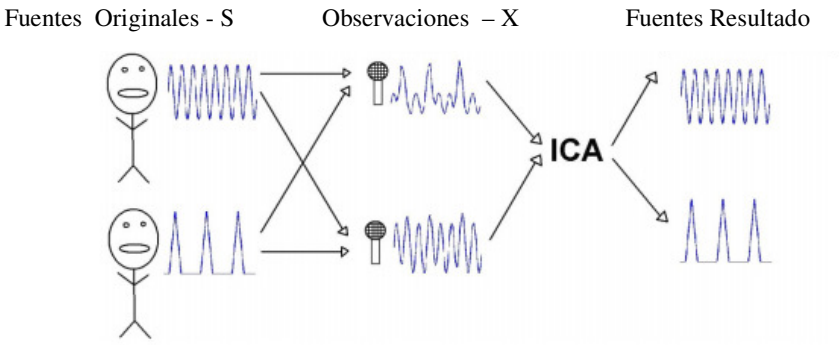


Figura 13 Síntesis procedimiento ICA

Considerando el vector de una variable aleatoria de observación X, cuyos *m* elementos son las mezclas de *m* elementos independiente de un vector de una variables aleatoria S dada por X=AS el objetivo principal de ICA es encontrar la variable $W = A^{-1}$ tal que Y = WX siendo Y una aproximación de S. Dicha aproximación no suele coincidir en los valores de amplitud de la señal pero sí en su forma.

Para poder hacer uso de ICA, se deben asumir ciertas propiedades sobre la naturaleza de las fuentes así como del proceso de mezcla.

El primer planteamiento es considerar que el proceso de mezcla es lineal o no lineal.

En el caso de la no linealidad, al considerar la existencia de un proceso de transformación no lineal, la extracción de las fuentes llega a ser complicada si no se aporta ningún tipo de información adicional. Por tanto viendo la dificultad de este problema se opta por asumir un modelo de mezcla lineal (Pedro A. Carrión, 2007).

Para poder aplicar el proceso ICA las fuentes deben cumplir al menos estas premisas:

1. Las fuentes son estadísticamente independientes. Es la condición más importante, sin la cual el proceso no tendría sentido. El concepto de independencia estadística se establece definiendo que dos variables aleatorias son independientes si y sólo si la información que aporta una no revela información sobre el valor de la otra, y viceversa (A. Papoulis, 1991).
2. El número de mezclas debe ser el mismo que el número de fuente y estas mezclas deben ser linealmente independientes, en otras palabras, ninguna de las mezclas es una combinación lineal de alguna de las restantes.
3. El modelo debe estar exento de ruido.
4. Se asume que los datos están centrados, por lo que su media es cero.

El problema de la separación de fuentes ha sido estudiado extensamente, dando lugar numerosos algoritmos utilizados según la naturaleza de las señales mezcladas. El problema de la separación ciega de fuentes es más difícil debido principalmente al desconocimiento de las señales que han sido mezcladas, ya que no es posible diseñar un preprocesamiento apropiado para separarlas óptimamente.

A lo largo de los años, el estudio de ICA se ha basado en dos vertientes muy diferencias: sistemas supervisados o no supervisados. Por lo general las redes neuronales de aprendizaje no supervisado no requieren de influencia externa para ir ajustando los pesos de las conexiones entre las neuronas, en otras palabras la red no recibe información que verifique si la salida, ante una determina entrada, es correcta o no. Por el contrario, en un sistema supervisado, a la red se le proporciona un conjunto de ejemplos del comportamiento de la propia red.

Así pues en el origen del algoritmo de separación ICA, los investigadores Jeanny Herault y Christian Jutten presentaron en el congreso celebrado en abril de **1986** en Snowbird Utah, un modelo de red neuronal y un algoritmo de aprendizaje supervisado basado en la versión del algoritmo de Hebb. El denominado aprendizaje Hebbiano implica la modificación de los pesos de la matriz de mezcla W, proporcional al producto de una entrada x_j y de una salida y_i de la neurona.

La única suposición impuesta por Herault y Jutten fue la independencia de estas señales. Mediante este algoritmo si se asumía que las señales fuentes eran GAUSSIANAS, entonces se demostraba que el problema no tenía una solución general. Los mejores resultado alcanzados por la red de Herault-Jutten se obtuvieron cuando las señales fuentes eran SUBGAUSSIANAS, es decir, cuando la Kurtosis es menor que la de una distribución Gaussiana, menor que 0. Sin embargo el resultado más óptimo fue alcanzado con apenas dos fuentes.

Años más tarde, en 1994 dentro del campo de redes neuronales se dio un paso desde algoritmos de aprendizaje supervisado a no supervisado. Empezó a surgir una clara competición entre diversos investigadores para conseguir el algoritmo que mejor y más rápido funcionara. Amari (Amari, 1998) propuso una método adaptativo de ajuste del ratio de aprendizaje. Sostuvo que el algoritmo podía ser mejorado si se usaba el gradiente natural, que multiplica el gradiente de la matriz de pesos W por una matriz positiva definida W^T, y además se aumentaba la velocidad de convergencia mediante la eliminación de la matriz inversa.

Sin embargo el algoritmo original ICA con sigmoidal no lineal era únicamente posible para fuentes SUPERGAUSSIANAS, Kurtosis mayor que cero. Te-Won Lee determinó que la clave para generalizar el algoritmo para cualquier fuente no-gaussiana, era estimar los momentos de las señales fuentes e inicializar el algoritmo adecuadamente, por tanto es importante saber a priori sobre qué tipo de señales se va a aplicar el proceso de separación.

En nuestro caso las señales almacenadas provienen de potenciales pericraneales humanos y debido a que tanto el cerebro como el pericráneo son muy buenos conductores y que se obtiene buenas aproximaciones de las señales fuentes, ICA es un proceso ideal para su separación, teniendo en cuenta que los

registro EEG son diferentes mezclas lineales tanto de señales cerebrales como de artefactos inherentes a la propia generación de las fuentes. Una extensión de este algoritmo proporciona un método más adecuado para poder separar estos artefactos, que incluyen fuentes Subgaussianas tales como ruido de 50 - 60 Hz, de las señales cerebrales que son generalmente SUPERGAUSSIANAS, de frecuencias, en estados de activación normal, inferiores a 40 Hz.

Véase la siguiente ilustración donde se puede comprobar que la existencia de ruido puede alterar el mapa de frecuencias de la señal original (Fig. 14).

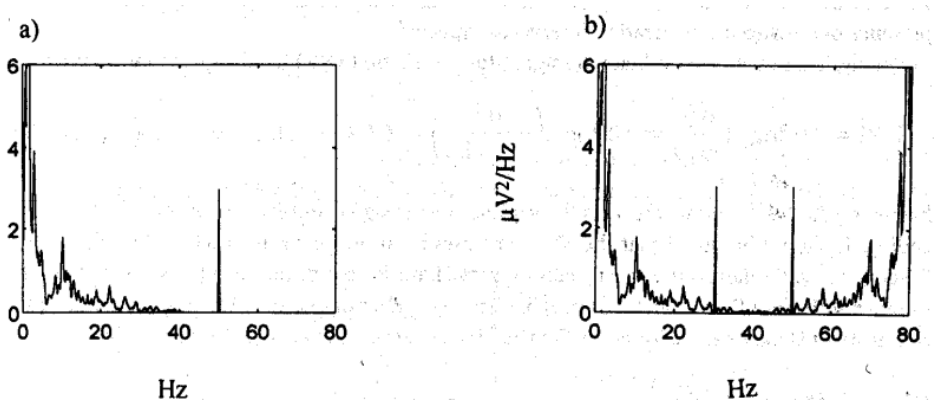


Figura 14. Potencia espectral de una señal de EEG (originalmente limitada en banda a 40 Hz). La presencia de ruido de 50 Hz (a) causa un error de aliasing en la componente de 30 Hz si muestreamos la señal a 80 Hz (b).

4.2. ICA y EEG

Generalmente todos los estudios están basados en la separación de señales mezcladas observadas desde un conjunto de sensores.

A partir de la base introducida por Herault and Jutten en 1986, han surgido diversas variaciones, tales como la dada por Comon (Comon, 1994) en la cual elaboró el concepto del análisis de componentes independientes y propuso funciones para aproximar la minimización de la información mutua entre el conjunto de sensores.

Paralelamente a los estudios de separación ciega de fuentes, una serie de modelos de aprendizaje no supervisado basado en información teórica fueron propuestos por Linsker (R., 1989). El acierto más importante se obtuvo al maximizar la información mutua entre las entradas y las salidas de la red neuronal. Cada neurona debe codificar rasgos independientes estadísticamente tanto como les fuera posible de las otras neuronas sobre un conjunto natural de entradas. Nadal y Parga (Lee, 1998) mostraron que en casos de bajo nivel de ruido, el máximo de información mutua entre las entradas y las salidas de la red neuronal implicaba que la distribución de la salida fuera factorial.

El modelo de aprendizaje para la separación ciega propuesta por Bell and Sejnowski (Sejnowski, 1995) era posible para fuentes SUPERGAUSSIANAS. Makeig et al. (1996) aplicó este algoritmo a datos EEG y ERP mostrando que podría extraer activaciones EEG y artefactos aislados. Jung et al. (1997) mostró que la extensión del algoritmo puede descomponer linealmente artefactos EEG tales como el ruido base, movimiento ocular y el ruido cardiaco, en componentes independientes con distribuciones sub-gaussianas y super-gaussianas.

Se pudo comprobar que las ondas ERP representan una suma de solapamientos, discretos y limitados en el tiempo, de eventos cuyas amplitudes están modeladas por una atención selectiva, sin afectar su curso a través del tiempo.

4.3. Separación ciega de respuestas cerebrales en componentes independientes

Los datos recogidos en promedios de potenciales relativos – ERP desde el pericráneo humano revela la actividad electroencefalográfica - EEG que está indudablemente relacionado en tiempo y fase con eventos experimentales. ICA descompone datos ERP en un número de componentes igual al número de sensores. Los componentes derivados tienen distinta, aunque no necesariamente ortogonal, proyección pericraneal. A diferencia de otros métodos, como el análisis de componentes principales que separa correlaciones de segundo orden, es decir, considerando la varianza de las señales, ICA también minimiza los más altos niveles.

A pesar de la distancia entre el cráneo y el cerebro y sus diferentes resistividades, los datos recogidos electroencefalográficamente desde cualquier punto en un pericráneo humano incluyen actividad generada en el interior de la extensa área del cerebro. Este "enmarañamiento" de datos EEG por volumen de conducción no involucra retrasos en el tiempo considerables. El algoritmo ICA separa el problema de la identificación de las fuentes de la localización de las mismas.

Así pues el problema de determinar las fuentes eléctricas cerebrales desde patrones de potencial registrados en la superficie del pericráneo es matemáticamente indeterminado. Para maximizar la entropía de la unión de un conjunto de canales de salida derivado de señales de entrada mediante un filtrado lineal sin retrasos de tiempo, el algoritmo procura derivar las fuentes independientes desde señales EEG correlacionadas en el más alto nivel estadístico sin tener en cuenta su localización o configuración de las fuentes generadoras.

ICA es un instrumento ideal para poder determinar, de forma matemática, no visual, las señales que son estadísticamente independientes sin tener en cuenta ni su localización ni las fuentes previas que generan dicha señal independiente y que pueden y de hecho identifican una situación concreta. No profundiza en la generación de dichas señales aisladas, sino que se concentra en alcanzar un resultado que pueda identificar el experimento realizado. Actualmente es difícil

averiguar qué conjunto de impulsos generan un determinado evento, ni la modulación o proceso, lineal o no, que sufre para ser propagado a través de la red cerebral. El mecanismo eléctrico cerebral se contempla como una "caja negra" donde se desconoce la entrada y su salida es procesada para poder averiguar qué señales caracterizan una situación singular.

Gracias a estos algoritmos es posible identificar estados físicos y mentales de un individuo. Aunque determinar un prototipo de señales ante procesos o estados precisos, tales como sueño, daños cerebrales o enfermedades psíquicas, es hoy en día practicable, la generalización de eventos cotidianos aún no es posible, debido principalmente al desconocimiento del sistema que tratamos, la tipificación de diversos estados aún no puede ser establecida. Cada red cerebral está gobernada por distintos niveles de amplitud y fase, modulados y propagados según la evolución del individuo teniendo como parámetros importantes la edad y la salud física y/o mental entre otros.

4.4. Algoritmo Runica()

El presente documento tiene su base en el algoritmo descrito por Bell and Sejnowski para el "análisis de componentes independientes" apoyado en la maximización de la información mutua entre las entradas y las salidas de una red neuronal - Infomax. Actualmente Scott Makeig, Tzyy-Ping Jung, Martin McKeown, Te-Won Lee, Dara Ghahremani y otros investigadores en el laboratorio de Terry Sejnowski, CNL, en el instituto Salk, La Jolla (Swartz Center for Computational Neuroscience, 2001), están explorando nuevas aplicaciones de ICA para procesamiento de señales biomédicas.

En el rango de frecuencias del registro EEG, la mezcla de los campos cerebrales en los electrodos pericraneales es básicamente lineal. El cráneo atenúa los potenciales cerebrales fuertemente, y los modifica espacialmente, aunque no afecta la relación lineal de los potenciales del cerebro en el pericráneo.

El algoritmo de Bell & Sejnowski (1995) resuelve el problema de las componentes independientes considerando las suposiciones indicadas anteriormente (*pto. 4.1 Introducción*):

- ✓ Las fuentes (fi(t), i = 1,2,...,p) son desconocidas y estadísticamente independientes.
- ✓ El número de sensores (p) es igual al número de fuentes.
- ✓ Se modela la influencia del medio material de manera lineal mediante una matriz de mezcla A

La principal ampliación del algoritmo RUNICA, del paquete MATLAB 7 basado en la modificación de Makeig et al. utilizado en nuestra experimentación es añadir el término "gradiente natural" W'*W – ("WTW") que elimina la matriz inversa durante el entrenamiento, lo cual reduce ampliamente la carga computacional.

Multiplicar el gradiente por W'*W no cambia el mínimo (ya que W'*W se define positivo) aunque altera el gradiente a $\frac{dW}{dt} = (I + (1 - 2y)u')W$ para que la convergencia continúe con la misma velocidad sin considerar el condicionante de la matriz mezclada. Esto se denomina equivarianza o gradiente natural.

Previo al entrenamiento de nuestra red, es posible realizar algún preproceso en las señales de entrada que mejore y simplifique la ejecución del algoritmo. Estos preprocesos son el blanqueo y el centrado de datos.

El centrado de datos es inherente al proceso y se ejecuta si o si, eliminando el valor medio de las muestras. Sin embargo el blanqueo puede ser seleccionado o no en la función.

Con el blanqueo de las muestras se consigue una nueva matriz origen, cuyas componentes son incorreladas y sus varianzas son iguales a la unidad. Básicamente lo que conseguimos con el blanqueo es reducir la carga computacional del algoritmo.

Runica() ejecuta el análisis de componentes independientes descomponiendo datos fisiológicos usando el algoritmo ICA de Bell & Sejnowski (1995) con el

gradiente natural introducido por Amari, Cichocki & Yang – *algoritmo extendido de ICA* de Lee, Girolami & Sejnowski.

Existen dos forma de ejecutar la función runica:

Una simple:

[weights,sphere] = runica(data);

Y otra más completa:

[*weights,sphere,activations,bias,signs,lrates]= runica(data,'Key1',Value1',...);*

La entrada principal lo forma una matriz de datos, *data*, especificada del siguiente modo:

data (chans,frames*epochs), donde *chans* identifica los canales de entrada introducidos, es decir, los electrodos a analizar, *frames* número de muestras y *epochs* las distintas épocas introducidas, es decir, existe la posibilidad de poder introducir dos eventos al mismo tiempo.

Asimismo encontramos una serie de parámetros de entrada adicionales que, dependiendo del análisis a realizar, serán considerados o no:

- 'ncomps'=[N] número de componentes ICA que se desea extraer, por defecto es el número de canales.
- 'pca'=[N] descomposición en PCA, componentes principales. Valor por defecto, 0, no activo.
- 'sphering'= Valores 'on'/'off', por defecto está activo. Utilizado para realizar un preblanqueo de los datos
- 'weights' = [W] Inicialización de la matriz de pesos, por defecto es la matriz unidad.
- 'lrate'= [rate] Velocidad de aprendizaje de la red utilizada, inicialmente la velocidad será menor que la unidad y por defecto variará heurísticamente

- 'block'= [N] Tamaño de los bloques de datos, si no se especifica nada, se considera un único conjunto de datos
- 'anneal' = control de la velocidad de convergencia entre 0 y 1
- 'annealdeg'= [N] grado de cambio de pesos para el control, por defecto 70.
- 'stop'= [f] tasa de parada del entrenamiento. Por defecto el entrenamiento finalizará cuando la variación de los pesos sea menor que 10^{-6}.
- 'maxsteps'= [N] número máximo de pasos de entrenamiento, en el supuesto de no alcanzarse el valor anterior, el proceso finaliza pasados 512 pasos por defecto.
- 'bias' = ['on'/'off'] ajuste de los pesos, por defecto está activo.
- 'momentum'= [0<f<1] momento de entrenamiento por defecto será cero.
- 'extended'= [N] considera como función de transferencia la tangente hiperbólica cada N bloques de entrenamiento. Si N < 0, fija el número de Sub-Gaussianas a –N, siendo más rápido que si N>0. Por defecto esté parámetro no está activo.

Los resultados obtenidos se identifican según los siguientes parámetros de salida:

- 'weights'= Matriz de pesos ICA (componentes canales) [RO]
- 'sphere'= matriz de datos preblanqueados (canales,canales) = spher(data)
- 'activations'= activación en el tiempo de las componentes de salida (número de componentes,tramos*épocas)

El algoritmo se basa en maximizar la información mutua tal y como se explica en el capítulo 3. *(Pto. 3.4 Maximización De La Información)*.

Como ejemplo del uso realizado del algoritmo runica() se describe la ejecución del fichero de muestras TTH99X.ASC – fichero de reposo y acción.

Previamente se pasan los registros a través de un filtro paso bajo de 40 Hz. Eliminando cualquier otra señal fuera del rango de estudio.

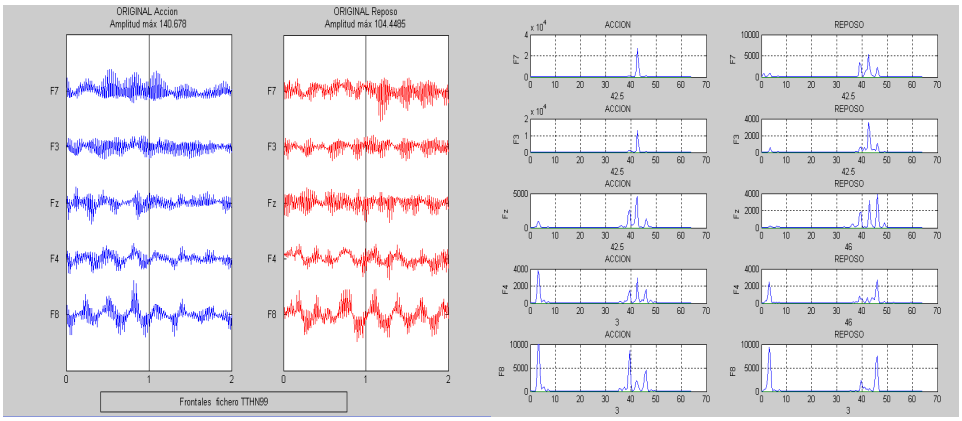


Figura 15 Representación gráfica en amplitud y frecuencia señales muestra

Una vez aplicado el filtro paso bajo de 40 Hz, dando como resultado:

.

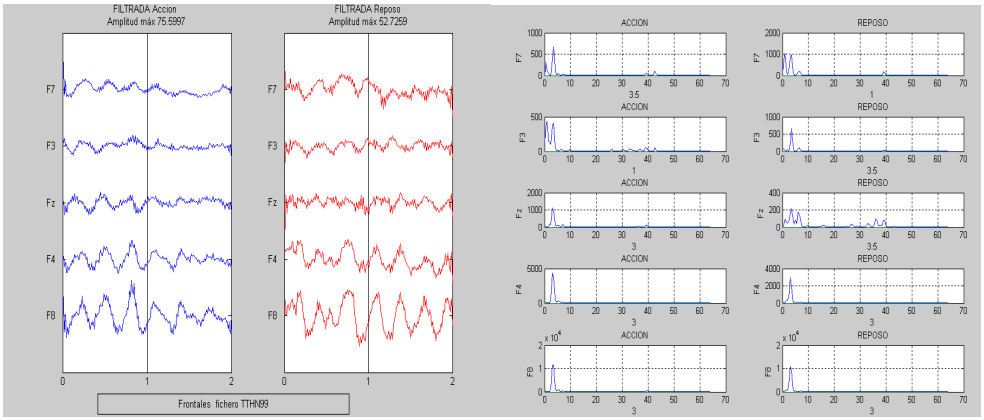


Figura 16 Representación gráfica muestras filtradas <40Hz. en amplitud y frecuencia señales muestra

Ejecutando el algoritmo runica(), aparecerá lo siguiente en el Matlab.:

```
5 fila 256 columna5 fila 256 columna
Input data size [5,256] = 5 channels, 256 frames.
Finding 5 ICA components using logistic ICA.
Initial learning rate will be 0.00932002, block size 9.
Learning rate will be multiplied by 0.9 whenever angledelta >= 60 deg.
Training will end when wchange < 1e-006 or after 512 steps.
Online bias adjustment will be used.
Removing mean of each channel ...
Final training data range: -28.5957 to 47.004
Computing the sphering matrix...
Starting weights are the identity matrix ...
Sphering the data ...

 Beginning ICA training ...
 step 1 - lrate 0.009320, wchange 0.207912
 step 2 - lrate 0.009320, wchange 0.076848
 step 3 - lrate 0.009320, wchange 0.095020, angledelta 91.3 deg
 step 4 - lrate 0.008388, wchange 0.122815, angledelta 109.4 deg
 step 5 - lrate 0.007549, wchange 0.063575, angledelta 114.0 deg
 step 6 - lrate 0.006794, wchange 0.063867, angledelta 105.9 deg
 step 7 - lrate 0.006115, wchange 0.073290, angledelta 120.6 deg
 step 8 - lrate 0.005503, wchange 0.044842, angledelta 131.5 deg
 step 9 - lrate 0.004953, wchange 0.028349, angledelta 140.1 deg
 step 10 - lrate 0.004458, wchange 0.023468, angledelta 127.6 deg
 step 11 - lrate 0.004012, wchange 0.017847, angledelta 115.5 deg
.
.
.
.
.
step 45 - lrate 0.000189, wchange 0.000006, angledelta 79.2 deg
step 46 - lrate 0.000170, wchange 0.000002, angledelta 61.6 deg
step 47 - lrate 0.000153, wchange 0.000002, angledelta 70.0 deg
step 48 - lrate 0.000138, wchange 0.000002, angledelta 62.2 deg
step 49 - lrate 0.000124, wchange 0.000003, angledelta 75.2 deg
step 50 - lrate 0.000112, wchange 0.000002, angledelta 92.7 deg
step 51 - lrate 0.000100, wchange 0.000001, angledelta 79.7 deg
step 52 - lrate 0.000090, wchange 0.000001, angledelta 81.6 deg
step 53 - lrate 0.000081, wchange 0.000001, angledelta 52.1 deg
Inverting negative activations: 1 -2 -3 -4 5
Sorting components in descending order of mean projected variance ...
1 2 3 4 5
Permuting the activation wave forms ...
```

Como se puede apreciar en la ejecución del algoritmo, los cambios de w se producen dependiendo del parámetro *angledelta.* Si se incrementa en la posición actual con respecto a la anterior *wchange* aumenta, lo que implica que la velocidad

de aprendizaje es elevada, disminuyendo para la siguiente iteración. Así hasta que la diferencia del valor de pesos sea inferior a 0.000001.

Runica devuelve dos valores: la matriz de blanqueo y la matriz de pesos. El resultado final son 5 componentes independientes representadas en orden decreciente de la varianza EEG. Por tanto la última posición nos indica la componente con mayor número de datos neuronales y/o artefactos. El número de componente en ningún momento nos está indicando que equivale a la señal recogida en el electrodo introducido en la misma posición (Fig. 17).

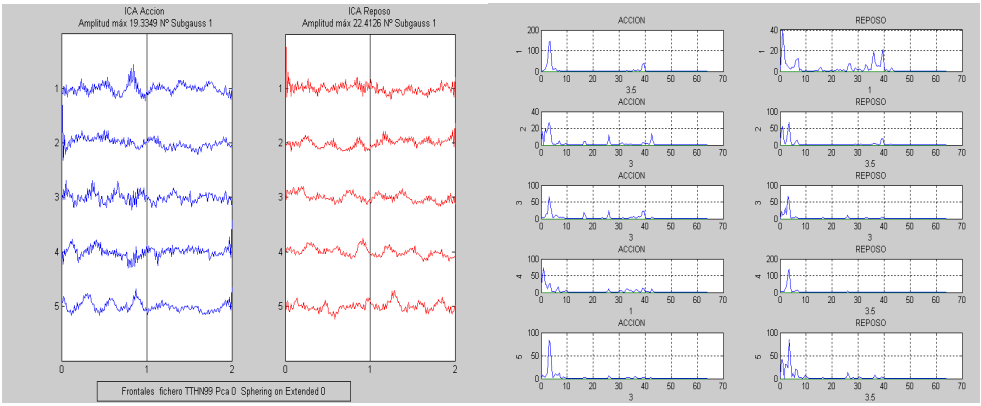


Figura 17 Resultado runica() en amplitud y frecuencia.

5. REDES NEURONALES

5.1. Introducción a las redes neuronales

En este capítulo se describe superficialmente la evolución del aprendizaje automático mediante redes neuronales artificiales dentro del campo del análisis predictivo.

La teoría de redes neuronales se empezó a desarrollar en la primera mitad del siglo XX. Sin embargo tuvo su mayor avance con la introducción de modernas computadores las cuales favorecieron el crecimiento y desarrollo de redes más complejas y sofisticadas.

La principal característica de las redes neuronales es su transportabilidad a distintos campos de la ciencia, pudiendo ser utilizadas en áreas de muy diversa índole: análisis de señales biomédicas, de imágenes, economía, etc. Por tanto la evolución de esta importante herramienta de análisis se ha visto incrementada a lo largo de los años.

Existen distintas arquitecturas que clasifican las redes dependiendo del número de elementos básicos utilizados, neuronas, y de su estructura dentro del sistema.

En presente capítulo se realizará una breve introducción a estos sistemas así como un desarrollo más extenso de la red utilizada en la experimentación.

5.2. Clasificación según el número de entradas

5.2.1. Neurona de una sola entrada

Neurona se define como la unidad de procesamiento de la información que ante un vector de entrada proporciona una única salida.

En este caso, dado un único vector de entrada p, éste se transmite multiplicándose por un vector de pesos w, de igual dimensión al dato de entrada. Dicho producto es la entrada de una función de transferencia F cuya salida es el escalar a, siendo el resultado al que llega la red (fig. 18).

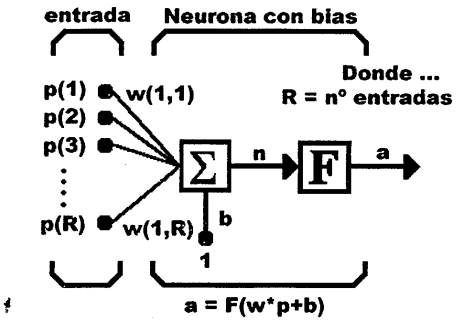


Figura 18 Esquema sistema de red neuronal de un único vector de entrada y una sólo neurona

El parámetro b se denomina bias y es utilizado para reajustar, sumando un valor constante, la función de transferencia.

5.2.2. Neurona de entrada múltiple

Este tipo de arquitectura es la más utiliza ya que integra n vectores de entrada hacia un única neurona.

Cada entrada individual p(1),p(2),...,p(R), se multiplica por los elementos w(1,1),w(1,2),...,w(1,R), siendo dicho producto vectorial la entrada al sumador, w*p, el producto del vector fila w y el vector columna p.

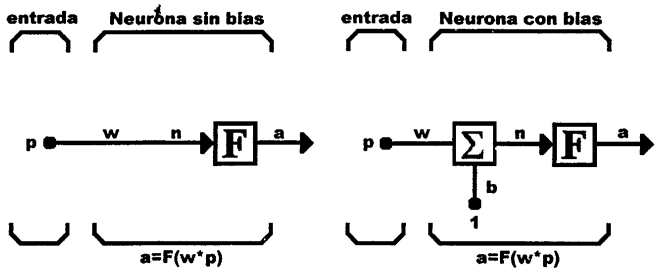


Figura 19 Esquema sistema de red neuronal con varios vectores de entrada y una sólo neurona con y sin parámetro bias.

5.3. Función de transferencia - F

La función de transferencia de una neurona puede variar dependiendo del tipo de aplicación que se desea emplear, hardlim (salida 0 ó 1), purelin (relación lineal entre la entrada y la salida) y logsig (entrada con valores $\pm \infty$ y salida entre 0 y 1). Todas ellas pueden optar por introducir un parámetro adicional, *bias*.

El bloque genérico F se sustituirá por uno específico según el tipo de función utilizada (fig.20):

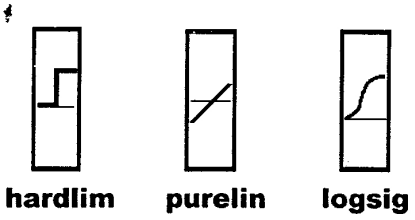


Figura 20 Distintos tipos de funciones de transferencia usadas en redes neuronales

La función de transferencia sigmoidal (denominada logsig), usada en el algoritmo de entrenamiento para el cálculo de componentes independientes tiene la forma de la Fig. 21.

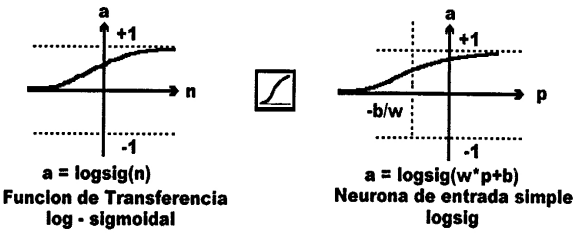


Figura 21 Función de transferencia usada en algoritmo ICA

Resumiendo, la arquitectura de una red neurona de una sola o múltiples entradas, es el resultado de pasar a través de una función de transferencia dada el producto de los vectores entrada por una matriz de pesos establecida, pudiendo añadir un parámetro adicional sumado a dicho producto.

5.4. Clasificación según el número de capas

5.4.1. Red monocapa

Generalmente las neuronas de una red se organizan en capas. Una red monocapa, como su nombre indica, sólo tiene una capa, en la que todas las neuronas reciben el estímulo de entrada y también todas las neuronas, contribuyen al valor de salida.

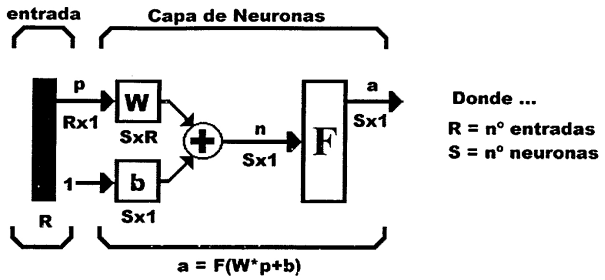


Figura 22 Red Monocapa

Es la arquitectura más simple de una red neuronal y en ellas se basa el PERCEPTON (red monocapa, con neuronas con función de transferencia lineal), creado por Rosemblatt el cual dio origen a la utilización de las redes para predecir procesos dependiendo del tipo de entrada introducido.

5.4.2. Red multicapa

La redes complejas organizan sus neuronas en capas, donde la transferencia de información se realiza desde las neuronas de las capas precedentes a las neuronas de las capas posteriores. No existe intercambio de información entre neuronas de una misma capa (Fig. 23).

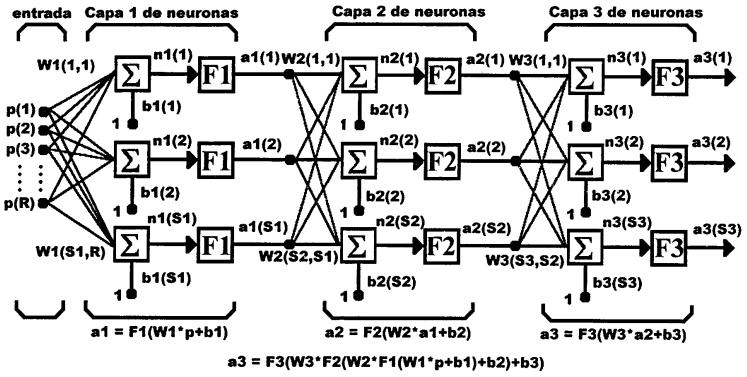


Figura 23 Red Neuronal multicapa

Las redes multicapa son muy potentes y seleccionando adecuadamente el número de capas, así como la funciones de transferencia de las neuronas, pueden llegar a aproximar sin cometer apenas errores, cualquier función matemática.

Cada capa puede realizar diferentes funciones, siendo la capa de salida la que dé el resultado final y el resto, denominadas capas ocultas, quienes realizarán procesos intermedios necesarios para obtener el resultado deseado.

5.5. Entrenamiento batching o procesamiento por lotes

El tipo de redes que a continuación se describen es el implementado tanto en la red ICA como en la clasificación, utilizadas en la experimentación.

El proceso de entrenamiento consiste en introducir a la red **un vector de entrada**, *calcular el vector de salida* y aplicar el algoritmo concreto para el cálculo del incremento de los pesos en la función del error cometido.

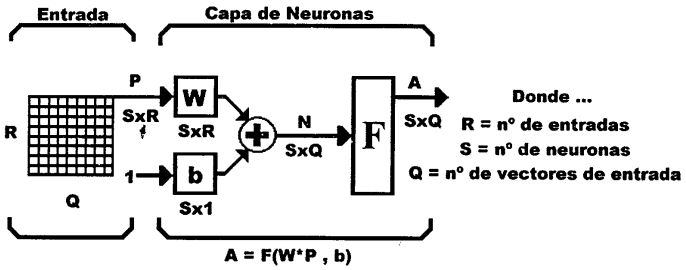


Figura 24 Red Neuronal batching

Como se puede apreciar el conjunto de datos es una matriz de Q vectores de entradas de longitud R que se aplican simultáneamente a la red. A la excitación de Q vectores de entrada, la red responde con Q vectores de salida que son los componentes de la matriz A. En nuestro caso se trata de la matriz formada por Q canales de señales de electrodos de tamaño R muestras, utilizadas simultáneamente en el entrenamiento de la red ICA, asimismo la red de clasificación tendrá una matriz de Q componentes independientes con R muestras.

5.6. Redes lineales

Este tipo de redes se diferencian del Percepton en su salida, ya que su valor puede ser distinto de 0 ó 1.

Dentro de la clasificación de datos, esta red tiene vital importancia ya que será la que encuadre las distintas neuronas en un determinado tipo de dato de salida, es decir, concretará la clase, acción o reposo, a la que pertenece la neurona ganadora.

A continuación se presenta la arquitectura básica de este tipo de redes, cuya función de transferencia utilizada es la denominada purelin o lineal.

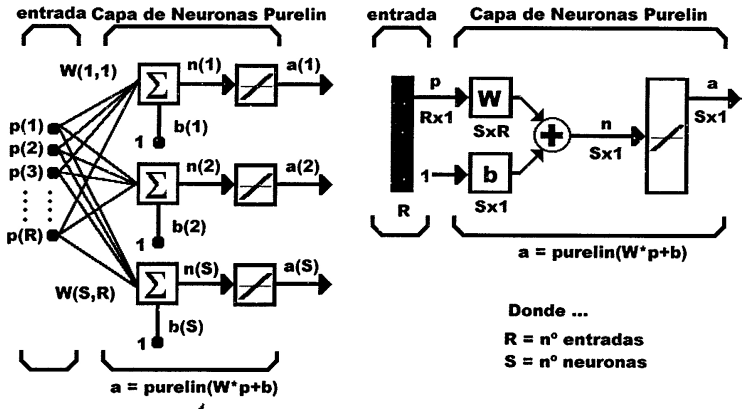


Figura 25 Esquematización Redes lineales

Introducido un vector de entrada determinado, una vez hallada la neurona ganadora, esta determinará a que clase pertenece dependiendo de los valores resultado de las distintas capas lineales.

5.7. Redes Learning Vector Quantization – LVQ

En el proceso desarrollado para la clasificación de los ficheros a estudio, se utiliza un tipo de red Backpropagation denominada LVQ, o cuantificación del vector de aprendizaje.

Esta red es la combinación de una capa de neuronas competitivas y una red neuronal lineal entrenada mediante el algoritmo Backpropagation. (Fig. 26)

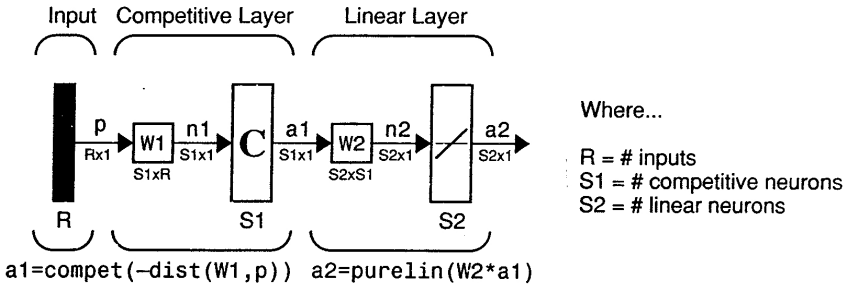


Figura 26 Estructura de una red LVQ

Existen por tanto dos capas bien diferenciadas, la capa competitiva que será la encargada de especializar las distintas neuronas hacia un valor preciso, y la capa lineal que realizará una clasificación de los resultados.

El entrenamiento de las capas competitivas se realiza en modo supervisado, es decir, previamente se entrenará la red con pares de vectores de entrada-salida conocidos, es decir, asociamos un vector de entrada a una clasificación dada de salida.

Para cada patrón de entrenamiento introducido en la red, se determina el vector de referencia que está más cerca de él. La neurona de salida correspondiente también se denomina neurona ganadora. Los pesos de las conexiones a esta neurona - y sólo esta neurona – se irán adaptando en el entrenamiento. La dirección de la adaptación depende de si la clase del patrón de entrenamiento y la clase asignada al vector de referencia coinciden o no. Si coinciden, el vector de referencia se mueve más cerca del patrón de entrenamiento, de lo contrario se mueve más lejos. Este movimiento del vector de referencia se controla mediante un parámetro denominado tasa de aprendizaje. Indica como una fracción de la distancia al patrón de entrenamiento hasta qué punto se mueve el vector de referencia. Por lo general, la tasa de aprendizaje disminuye con el transcurso del tiempo, por lo que los cambios iniciales son mayores que los cambios realizados en épocas posteriores del proceso de formación. El aprendizaje puede terminar cuando las posiciones de los vectores de referencia apenas cambian, siempre y cuando esto se produzca antes de finalizar el número de épocas o entrenamiento máximos seleccionados.

Un punto importante a tener en cuenta es que las clases que la capa competitiva encuentra, están directa y únicamente relacionadas con la distancia entre los vectores de entrada, sin incorporar ninguna característica adicional.

Los parámetros de entrada de este tipo de redes están formados principalmente por un vector de entrada y un vector de objetivos asociado a ésta.

Es importante que los vectores de entrada/objetivo estén distribuidos exactamente para poder inicializar correctamente los pesos de la segunda capa. La función de inicialización de los pesos debe conocer las distribuciones relativas de las

clases. Es decir, si el 68% de los vectores de entrada pertenecen a la clase 1, entonces la red deberá especializar 68% de las neuronas de la capa competitiva para encontrar esta clase.

El algoritmo utilizado en este tipo de clases, llamado de KOHONEN, se basa en encontrar aquella neurona competitiva cuyos pesos y forma se aproximen al vector de entrada, siendo la neurona ganadora cuya salida será 1. El resto de neuronas tomarán valor 0. Los pesos de la neurona ganadora serán entonces actualizados con el algoritmo de Kohonen de tal forma que modifique el vector de pesos lo más cerca posible al vector de entrada. El resto quedará con los pesos anteriores. Este proceso se realizará reiteradamente hasta alcanzar el total de épocas a ejecutar, aunque no converja hacia el valor deseado. Así pues es importante determinar el número de bucles a realizar por el algoritmo para poder encontrar la solución correcta.

Una vez entrenado se obtendrá una matriz de pesos que será la que determine, dado un vector de entrada cualquiera, a que clase pertenece.

6. DESCRIPCIÓN EXPERIMENTAL

6.1. Procesamiento de señales – Aplicaciones - Análisis de señales EEG y Red Neuronal

Para el análisis de las señales de muestreo se han generado dos funciones bien diferenciadas.

- Análisis de Señales EEG – Selección, carga y extracción de componentes independientes de las muestras experimentales.
- LVQ – Del resultado de la primera aplicación, alimentamos la red neuronal definida en está aplicación.

Como resultado de ambas aplicaciones se extrae el ERROR resultante de la simulación realizada en la red neuronal.

A través de estas dos aplicaciones y la selección de los distintos parámetros disponibles, se ha hecho posible la realización de una serie de pruebas cuyo resultado se muestra más adelante.

6.2. Clasificación de señales - Pruebas y Resultados

Cada una de las pruebas realizadas en el presente experimento se basa en un método de selección y almacenamiento de datos recogidos desde el pericráneo humano de un individuo en concreto. Para ello se realiza el proceso que a continuación se muestra esquemáticamente (Fig. 27):

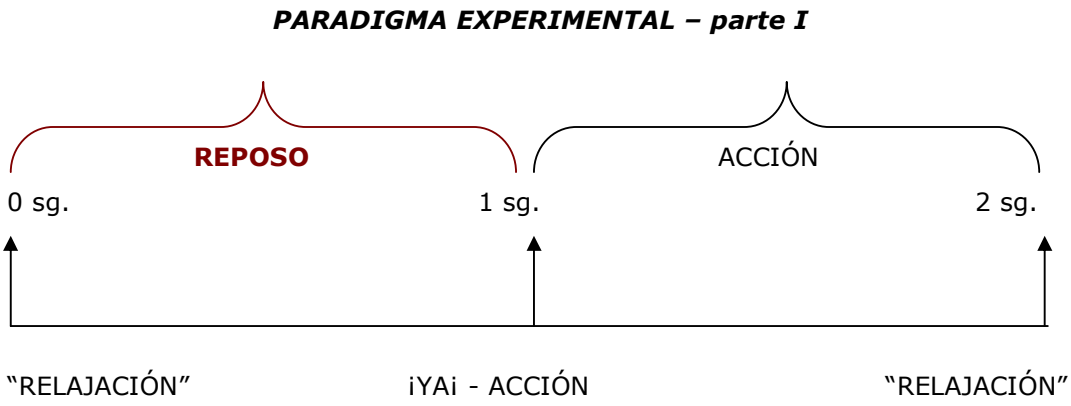
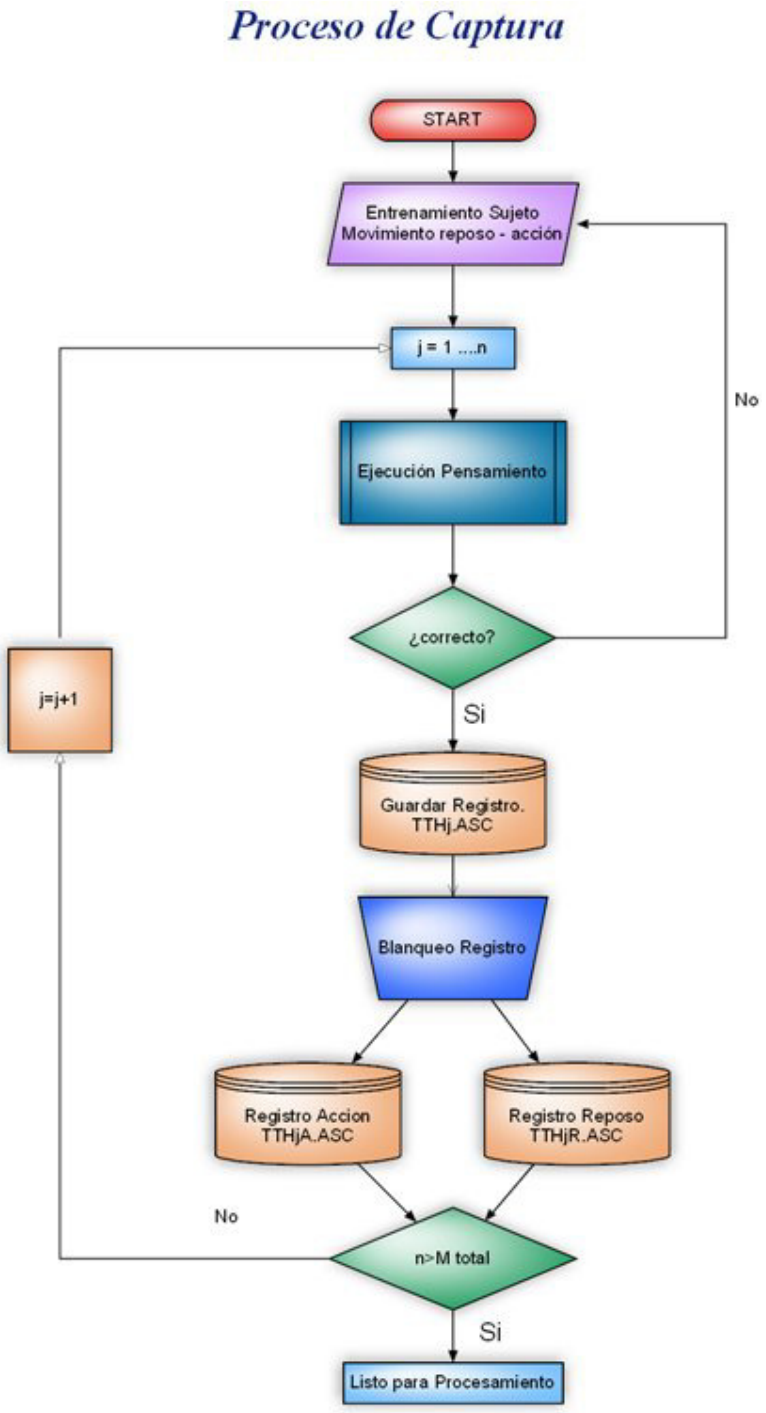


Figura 27 Representación gráfica de la metodología utilizada para la extracción de las muestras del experimento.
Parte I – Paso de situación de reposo a acción – Parte II – selección de datos almacenados.

El proceso de captura se puede resumir en el siguiente diagrama:



Inicialmente el sujeto es sugestionado para permanecer en estado de reposo absoluto. Posteriormente se le ordena que "imagine" reiterativamente una acción concreta. Esta acción conlleva numerosas acciones simples, que se registran simultáneamente en el proceso. Pasado un periodo de tiempo no determinado se insta al sujeto que retorne al estado de relajación, iniciando nuevamente el experimento. Seguidamente se realiza el preblanqueo de las señales recogidas, tomando el registro completo y seleccionando los tramos de registro EEG que alcance ambos estados, tal y como se muestra en el esquema anterior, considerando niveles y frecuencia de señal tipificadas en tal situación, almacenado 2 sg. de cada uno de los estados en un sendos ficheros denominados, acción y reposo.

En todos los casos se filtran las señales de entrada por un filtro paso bajo de 40 Hz, eliminando cualquier señal superior a esa frecuencia, al estar fuera del rango experimental.

El proceso ICA se ejecuta activando el preblanqueo sobre estas señales, además de considerar distintos grupos de electrodos.

El rango de los datos introducidos en la función rúnica() es el total de muestras, 256 – 2 segundos. Para eliminar el tiempo computacional del procesamiento de datos, se seleccionan diversos canales origen, considerando por tanto en cada experimento distintos grupos de fuentes, es decir, distintos grupos de electrodos. Se tendrá en cuenta el mapa de actividad cerebral conocido hasta el momento (fig. 28), poniendo en énfasis las zonas con mayor probabilidad de obtener la clasificación esperada. zona frontal y/o central, zona cerebral donde se ha observado mayor actividad neuronal en acciones motoras – (fig. 29):

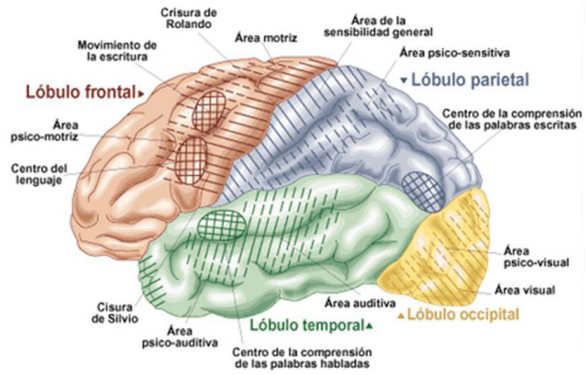
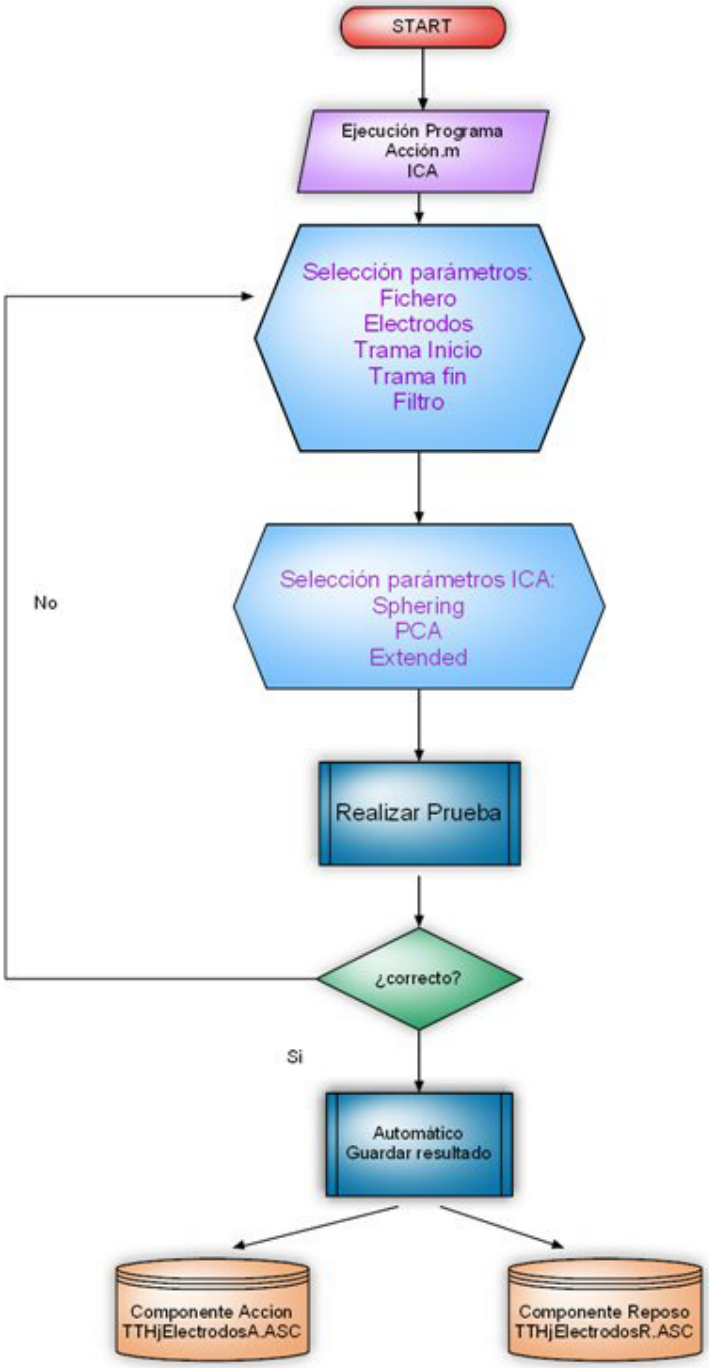


Figura 28 Mapa actividad cerebral

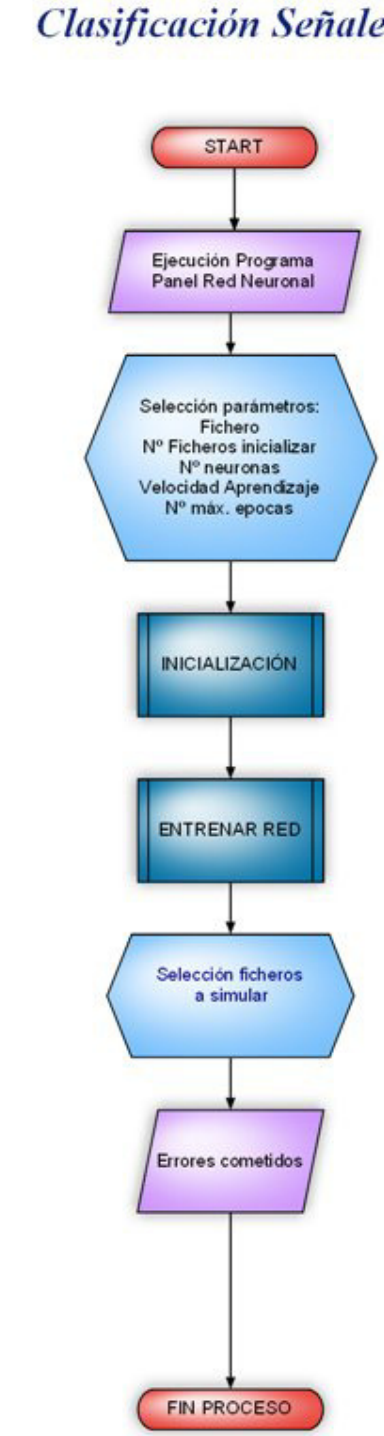
Se ha de tener en cuenta que el experimento seleccionado posee numerosos condicionantes debido a que la acción a realizar es compleja, y conlleva otras acciones simples que se escapan al análisis. Asimismo se debe considerar la percepción del propio sujeto de la situación, ya que en sesiones sucesivas los resultados pueden ser similares, pero si transcurren días o simplemente horas, esa misma situación puede variar, no teniendo ninguna validez.

Una vez recogidas las muestras, dichos ficheros son procesados a través del programa Acción.m – ejecutado bajo en entorno Matlab 7.0 según el siguiente diagrama:

Procesamiento Señales



Conseguidas las componentes independientes de ambas acciones, tanto de reposo como de la acción a estudio, éstas son pasadas a una red neuronal para su entrenamiento y posterior clasificación, tal como figura en el siguiente diagrama:



El resultado final será los errores cometidos tanto parciales como generales al clasificar el resto de señales, no involucradas en el entrenamiento, obtenidas en la separación de componentes. De este modo se podrá valorar la viabilidad del sistema.

6.2.1. Colección muestras nº 1 – Acción Compleja

El primer grupo de datos recogidos para el análisis proceden de una experimentación compleja. El sujeto debe imaginar las acciones que debe realizar para dirigirse hacia una puerta y abrirla, recogiendo los resultados tal y como se mostró anteriormente.

ELECTRODOS	FICHEROS		PARÁMETROS RED			ERRORES %		
	Entrenados	Simulados	Velocidad	Épocas	Neuronas	Acción	Reposo	Total
TODOS	100	23	0.08	1000	50	0	17	9
TODOS	23	100	0.08	1000	50	51	0	26
TODOS	10	113	0.08	2000	5	70	3	37
FRONTALES	100	23	0.08	1000	50	0	0	0
FRONTALES	10	113	1	255	2	43	18	31
FRONTALES	10	113	0.08	1000	5	50	1	26
FRONTALES	10	113	0.08	2000	5	41	15	28
FRONTALES	23	100	1	255	2	0	100	50
FRONTALES	23	100	0.08	1000	50	0	18	9
FRONT+CENTRALES	100	23	0.08	1000	50	0	4	2
FRONT+CENTRALES	10	113	1	255	50	0	100	50
FRONT+CENTRALES	23	100	0.08	1000	50	51	29	40
FRONT+CENTRALES	50	63	0.08	1000	50	15	3	9
FRONT+CENTRALES	75	48	0.08	1000	50	0	0	0

Tabla 1 Resultado más significativo experimento Acción Compleja-Reposo

Los mejores resultados presentados en la tabla anterior muestran los siguientes mapas de ubicación de los vectores de entrada de inicialización de la red neuronal:

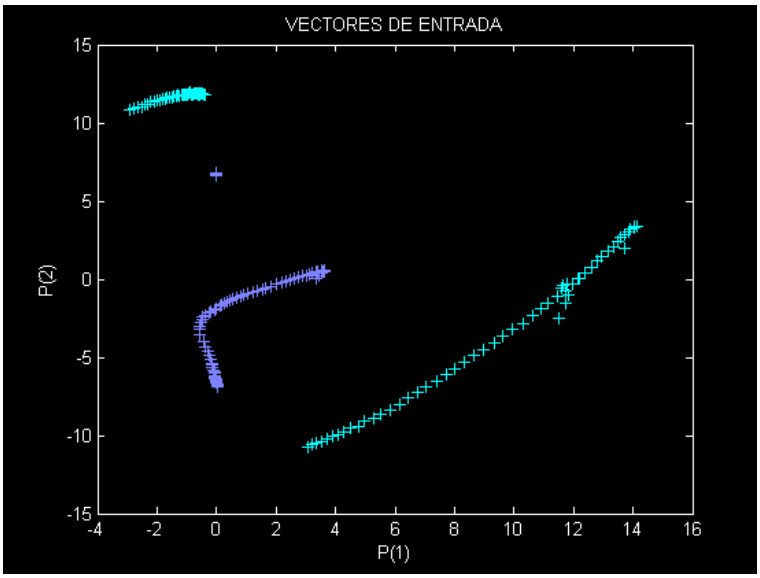


Figura 29 Vectores de entrada proceso sensores frontales – Azul claro – Pensamiento Acción – Azul Oscuro – Pensamiento Reposo

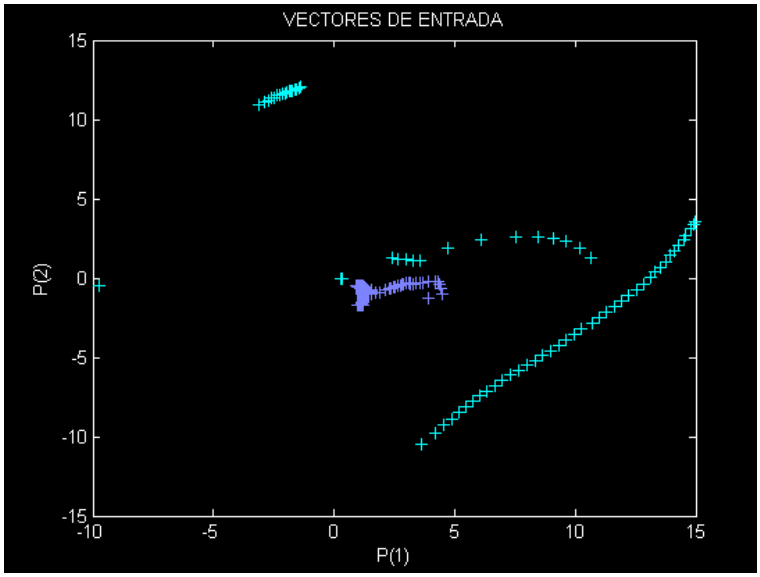


Figura 30 Vectores de entrada proceso sensores frontales+centrales – Azul claro – Pensamiento Acción – Azul Oscuro – Pensamiento Reposo

Conclusiones:

La velocidad de aprendizaje está íntimamente relacionada con el tipo de datos involucrados en el proceso. Teniendo en cuenta la distancia de separación entre las distintas pruebas, ésta deberá ajustarse de tal forma que la aproximación de las neuronas sea mediante pequeños "saltos".

El número de épocas de entrenamiento debe ser tal que una vez alcanzado un valor óptimo de aproximación un paso más no nos aleje de él. Es decir, aumentar el número de épocas no implica una mejora en el resultado, e incluso puede alejarnos aún más de él, ya que aunque las neuronas converjan hacia un valor determinado, al tener datos de estas dimensiones, 100 ó 200 vectores, puede suceder que algunas de estas neuronas comiencen a aproximarse hacia otros valores no alcanzando el valor deseado al finalizar el bucle. Esto no ocurre en procesos donde se tenga un número reducido y diferenciado de datos a clasificar, pudiendo especializar una neurona para cada uno ellos, dando lugar una convergencia completa, una neurona para cada uno de los vectores de entrada.

Como se puede apreciar en el mapa de entrada de la red neuronal del grupo de electrodos Frontales+Centrales, los vectores relacionados con el pensamiento de reposo está claramente ubicados, en cambio los de acción, aunque ciertamente tienen zonas diferencias, éstas distan mucho entre ellas, aunque no confluyen en el mismo espacio del registro de reposo.

De los resultados obtenidos es posible concluir que la clasificación es óptima tanto son seleccionados canales frontales, frontales+centrales e incluso añadiendo todos los canales muestreados. Sin embargo hemos de considerar la carga computacional al seleccionar unos u otros

ELECTRODOS	FICHEROS		PARÁMETROS RED			ERRORES %		
	Entrenados	Simulados	Velocidad	Épocas	Neuronas	Acción	Reposo	Total
FRONT+CENTRALES	75	48	0.08	1000	50	0	0	0
FRONTALES	100	23	0.08	1000	50	0	0	0
FRONT+CENTRALES	100	23	0.08	1000	50	0	4	2
FRONTALES	23	100	0.08	1000	50	0	18	9
FRONT+CENTRALES	50	63	0.08	1000	50	15	3	9
TODOS	100	23	0.08	1000	50	0	17	9
FRONTALES	10	113	0.08	1000	5	50	1	26
TODOS	23	100	0.08	1000	50	51	0	26
FRONTALES	10	113	0.08	2000	5	41	15	28
FRONTALES	10	113	1	255	2	43	18	31
TODOS	10	113	0.08	2000	5	70	3	37
FRONT+CENTRALES	23	100	0.08	1000	50	51	29	40
FRONT+CENTRALES	10	113	1	255	50	0	100	50
FRONTALES	23	100	1	255	2	0	100	50

Tabla 2 Orden de menor a mayor Error resultado más significativo experimento Acción Compleja-Reposo

Analizando la tabla 2 se puede comprobar que los mejores resultados son los obtenidos analizando los sensores frontales y/o frontales+centrales. También

hemos considerar la importancia en la selección de los parámetro Velocidad y Épocas de entrenamiento y Neuronas.

6.2.2. Colección Muestras nº 2 – Acción Puño

Para verificar el resultado obtenido anteriormente se opta por modificar la situación de acción en el sujeto, centrándonos en un movimiento simple de la mano izquierda: cerrar el puño. Se evita de esta forma introducir complementos adicionales a la acción. El área que controlan tanto la mano como los dedos es la más amplia, por tanto tal movimiento tendrá un espacio mayor de actividad que en el resto de acciones, así pues los potenciales de acción asociados a tal movimiento serán registrados en más canales.

A continuación se presentan los datos obtenidos mediante el análisis expuesto anteriormente – Tabla 3.

ELECTRODOS	FICHEROS		PARÁMETROS RED			ERRORES %		
	Entrenados	Simulados	Velocidad	Épocas	Neuronas	Acción	Reposo	Total
TODOS	50	39	0.08	1000	50	0	23	12
TODOS	10	79	0.08	2000	5	34	29	32
TODOS	23	66	0.08	1000	50	5	2	4
FRONTALES	50	39	0.08	1000	50	0	15	8
FRONTALES	10	79	1	255	2	100	6	53
FRONTALES	10	79	0.08	1000	5	20	66	43
FRONTALES	10	79	0.08	2000	5	20	8	14
FRONTALES	23	66	1	255	2	0	100	50
FRONTALES	23	66	0.08	1000	50	0	100	50
FRONT+CENTRALES	70	19	0.08	1000	50	0	0	0
FRONT+CENTRALES	10	79	1	255	50	0	87	44
FRONT+CENTRALES	23	66	0.08	1000	50	27	2	15
FRONT+CENTRALES	50	39	0.08	1000	50	41	0	21
FRONT+CENTRALES	75	14	0.08	1000	50	0	0	0

Tabla 3 Resultado más significativo experimento Acción Puño-Reposo

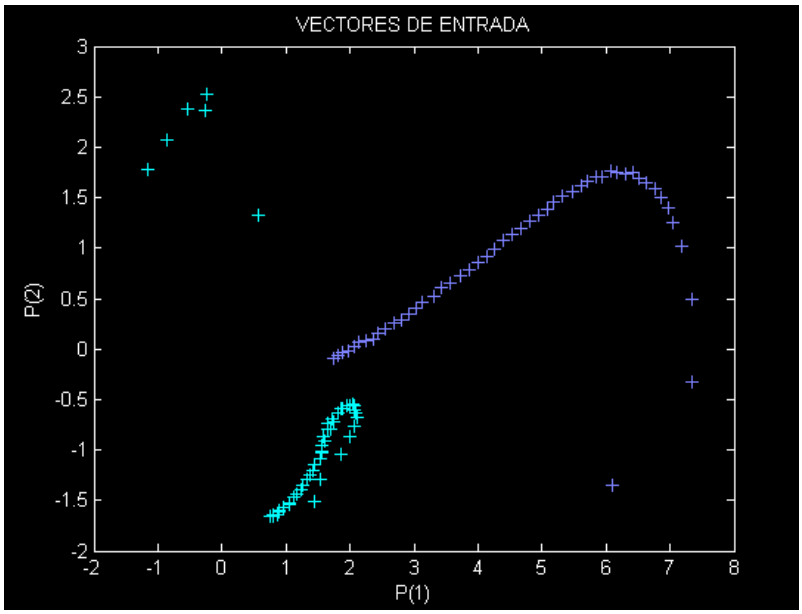


Figura 31 Vectores de entrada proceso sensores frontales – Azul claro – Pensamiento Acción – Azul Oscuro – Pensamiento Reposo

ELECTRODOS	FICHEROS		PARÁMETROS RED			ERRORES %		
	Entrenados	Simulados	Velocidad	Épocas	Neuronas	Acción	Reposo	Total
FRONT+CENTRALES	70	19	0.08	1000	50	0	0	0
FRONT+CENTRALES	75	14	0.08	1000	50	0	0	0
TODOS	23	66	0.08	1000	50	5	2	4
FRONTALES	50	39	0.08	1000	50	0	15	8
TODOS	50	39	0.08	1000	50	0	23	12
FRONTALES	10	79	0.08	2000	5	20	8	14
FRONT+CENTRALES	23	66	0.08	1000	50	27	2	15
FRONT+CENTRALES	50	39	0.08	1000	50	41	0	21
TODOS	10	79	0.08	2000	5	34	29	32
FRONTALES	10	79	0.08	1000	5	20	66	43
FRONT+CENTRALES	10	79	1	255	50	0	87	44
FRONTALES	23	66	1	255	2	0	100	50
FRONTALES	23	66	0.08	1000	50	0	100	50
FRONTALES	10	79	1	255	2	100	6	53

Tabla 4 Orden de menor a mayor Error resultado más significativo experimento Acción Puño -Reposo

Al igual que en el caso de Acción Compleja , los mejores resultados se han establecido dentro del rango de electrodos FRONTALES+CENTRALES (ver tabla 5), llegando a una clasificación sin errores, aunque el resultado obtenido con todos los canales seleccionados arroja un resultado no desdeñable, del 4%, sin embargo es importante considerar la menor carga computacional al seleccionar un grupo menor de sensores.

7. CONCLUSIONES FINALES

Con el presente estudio se ha intentado descubrir una asociación entre la actividad eléctrica cerebral y la actuación de un sujeto singular. Únicamente se ha procurado justificar, más o menos razonablemente, las características de las señales cerebrales, obtenidas en la superficie craneal, que un determinado sujeto emite ante situaciones muy dispares.

Tal y como se ha expuesto, desde el comienzo de un estímulo dado hasta su propagación, las señales asociadas sufren una serie de transformaciones para poder transmitir la información necesaria que produce un evento o un estado determinado. Sin embargo aún no es posible discernir la cantidad de señales que entran en juego en dicho sistema, así como qué componentes determinan la ejecución o inhibición de tal acción.

El cerebro humano es analizado como un sistema global del cual únicamente se conoce la salida que proporciona, siendo ésta la que establece el proceso asociado al estado de sujeto determinado. Teniendo en cuenta que los niveles potenciales generados por las neuronas están intrínsecamente relacionados con la persona analizada, no es posible crear un estándar de comportamiento, debiendo diferenciar las señales generadas singularmente.

Numerosos estudios científicos han podido tipificar diversas señales ante estados muy determinados, siendo generalizados para posteriores artículos, tales como sucesos de ensueño o actividad motora. Tales patrones como los ritmos alfa, beta y theta pueden establecer si un sujeto está en estado de letargo o de actividad mental. Por tanto observando la frecuencia y la amplitud de los registro EEG es posible encuadrar de forma generalizada qué estado se está observando.

El principal avance de la siguiente investigación es, no sólo poder separar las señales por simple observación sino crear un sistema que automáticamente realice dicho proceso. Posiblemente aún se esté demasiado lejos de poder definir qué tipo de señales generan ciertos tipos de movimiento. La localización de la zona que

gobierna cada una de las partes del cuerpo humano es bien conocida aunque no la diferenciación de las distintas ondas que generan.

Posiblemente éste y otros estudios análogos sean el comienzo de una análisis más profundo del SISTEMA NEURONAL CENTRAL. En principio sólo es posible diferenciar con un tanto por ciento de acierto dos estados fundamentales, pensar en "nada" o en "algo".

Aunque se ha podido alcanzar un porcentaje aceptable de clasificación de las dos acciones, no se debe olvidar que la efectividad de este estudio no ha podido ser demostrada con ninguna otra acción ni individuo, por tanto existe un porcentaje alto de incertidumbre de los resultados y por tanto de su posible aplicación.

Sin embargo la conclusión a la que es posible llegar está relacionada con los métodos utilizados. Tanto la función runica() como las redes LVQ, son instrumentos importantes a ser desarrollados e implementados en futuras investigaciones. Así mismo se ha de destacar que para poder alcanzar un análisis adecuado es preciso realizar una preselección de las señales obtenidas, evitando de esta forma introducir componentes que puedan encubrir los resultados esperados. Obteniendo un conjunto de datos adecuados facilita en gran medida la clasificación de los mismos. Teniendo en cuenta la distribución de la señales registradas considerando la comparación de sus órdenes estadísticos, tales como la media, varianza, simetría y kurtosis, proporcionan un instrumento de análisis importante para la caracterización de señales de ésta y otra naturaleza.

En nuestro caso el estudio se centra en la obtención de señales cuya distribución sea SUPERGAUSSIANA, característica principal de las señales cerebrales, por tanto el estadístico elegido para la preselección de las componentes independientes a clasificar es la Kurtosis y siempre teniendo en cuenta que los registros EEG son mezclas lineales de éstas y otras señales involucradas, tales como artefactos musculares o el ruido derivado de movimiento de fluidos cerebrales, teniendo distribuciones generalmente subgaussianas.

Se ha de tener en cuenta que el conjunto total de datos analizados está en torno a 256 ficheros o muestras de un único sujeto, en acciones muy determinadas, situación de relajación total e imaginación de un movimiento preciso. Por tanto es

muy difícil llegar a la conclusión de que puedan darse los mismos resultados analizando otro sujeto u otra acción. Así pues no sería adecuado dar por finalizado dicho estudio, siendo esencial ampliar aún más este trabajo para llegar a resultados más fiables.

Resumiendo, el estudio se basa en seis puntos:

1. Recopilar registros EEG de un número limitado, (en nuestro caso 50) de dos acciones a distinguir.
2. Filtrado en banda de las señales: <40Hz
3. Reducir el número de componentes a entrenar.
4. Entrenamiento de la red.
5. Obtención de la matriz de pesos que clasificará las señales.
6. Introducir una señal cualquiera al sistema, obteniendo un resultado específico, con un error medio relativo del 14%.

En este proceso de experimentación sólo ha sido posible la distinción de dos acciones con unas condiciones a priori muy diferencias, acción reposo y acción motora mental. Al tratarse de muestras de dos sujetos, no ha sido posible verificar la posible diferenciación entre señales de dos acciones mentales motoras distintas. Así pues este estudio queda inconcluso al obtener únicamente una clasificación de dos únicos procesos.

Tal y como se indico al principio del documento la principal idea es englobar este proceso en un sistema cuya entrada sea un registro de electrodos, en este caso el mejor resultado sería frontales y/o centrales, y su salida la acción clasificada, dando como resultado generar un movimiento complejo a través de un sistema mecánico (mover un ratón de ordenador hacia la derecha con la mente) asociado a uno simple (imaginar cerrar la mano derecha). La posibilidad de añadir acciones mentales diversas y asociarlas unívocamente a un movimiento mecánico preciso, sería la culminación de proceso.

VII. PRESUPUESTO

El presente estudio tiene como fin último poder dar validez a la ejecución de un sistema automático de captación, procesamiento y clasificación de señales encefalográficas procedentes de un sistema periférico craneal no invasivo. El presupuesto global del proyecto se puede desglosar en tres apartados principales: Sistema EEG, software adicional y mano de obra.

Sistema EEG

Sistema EEG portátil basado en PC tipo notebook, incluye licencia S.O. Windows 7, electrodos, filtro de reducción de artefactos y amplificador de señal

Producto	Cantidad	Precio (euros)	Total (euros)
ELECTROENCEFALOGRAFO NEUROFAX EEG-9100 NIHON KOHDEN o similar	1	7.598,15 €	7.598,15 €
		TOTAL	**7.598,15 €**

Software adicional

Software de desarrollo Matlab 7, Matlab Runtime (conjunto independiente de bibliotecas compartidas necesario para la ejecución de aplicaciones desarrolladas sin necesidad de obtener una licencia MATLAB) y Toolbox ICA.

Producto	Cantidad	Precio (euros)	Total (euros)
MATLAB Component Runtime	1	free	free
Matlab 7 o Superior	1	500,00 €	500,00 €
Matlab toolbox ICA	1	free	free
		TOTAL	**500,00 €**

Mano de obra

Respecto a la mano de obra, el precio por hora seria de aproximadamente 65€/h técnico especialista y auxiliar 30€/h desempeñando varias labores.

Producto	Cantidad	Precio (euros)	Total (euros)
Recopilación datos técnico especialista	40	65,00 €	2.600,00 €
Auxiliar/es	20	30,00 €	600,00 €
Investigación	240	65,00 €	15.600,00 €
Programación	120	65,00 €	7.800,00 €
Documentación	40	65,00 €	2.600,00 €
		TOTAL	**29.200,00 €**

Presupuesto total:

Producto	Total	I.V.A (21%)	
Sistema EEG	7.598,15	9.193,76 €	
Software adicional	500,00 €	605,00 €	
Mano de obra	29.200,00	35.332,00 €	
	TOTAL	**45.130,76 €**	

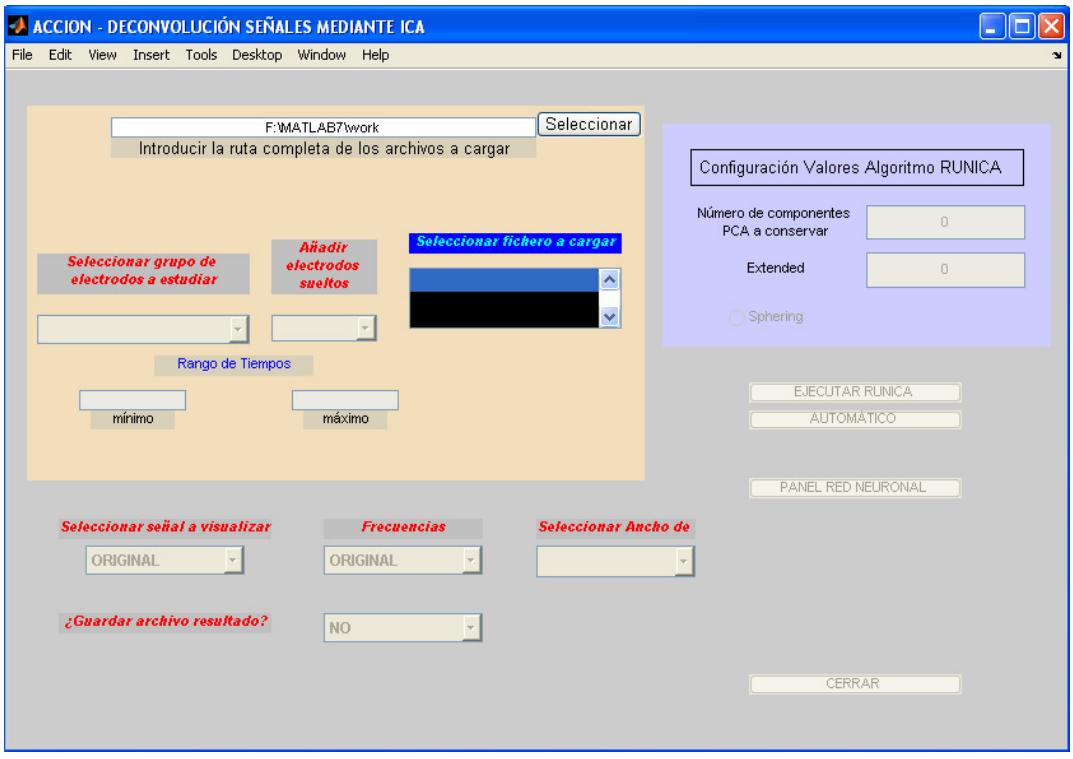
Tiempo estimado de ejecución aprox. Tiempo completo	*3 meses*

VIII. MANUAL USUARIO

Las aplicaciones realizadas para el experimento serán detalladas en los siguientes puntos.

Para poder ejecutar las aplicaciones se hace preciso la instalación del paquete de desarrollo MATLAB Component Runtime (*MATLAB Runtime es un conjunto independiente de bibliotecas compartidas que permite la ejecución de aplicaciones o componentes de MATLAB 7 compilados en ordenadores que no tienen instalado MATLAB*). Dicho paquete está incluido en soporte digital del presente proyecto.

La aplicación a ejecutar se denomina Accion.exe. y su primera ventana de selección es la siguiente:

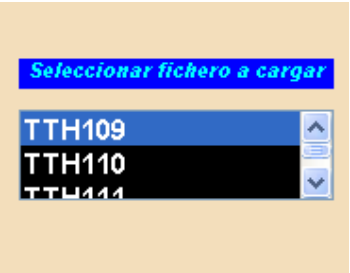


El primer paso a realizar es la selección del directorio donde se encuentren la muestras a analizar. Los ficheros a ser analizados deben tener la siguiente nomenclatura:

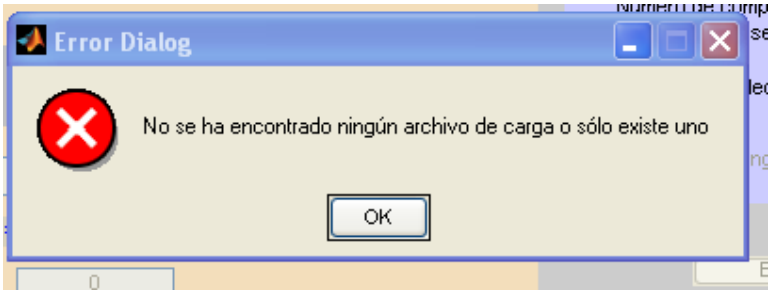
- Fichero de Acción Compleja: TT*A.asc - * pudiéndose sustituir por cualquier identificación, en nuestro caso número de muestra válida – Ej: TTH109A.ASC
- Fichero de Accion Puño: TT*P.asc - * pudiéndose sustituir por cualquier identificación, en nuestro caso número de muestra válida – Ej: TTN250R.ASC
- Fichero de Reposo: TT*R.asc - * pudiéndose sustituir por cualquier identificación, en nuestro caso número de muestra válida – Ej: TTH109R.ASC

En ambas colecciones de muestras un fichero Acción Compleja o Acción Puño lleva asociado un fichero Reposo con el mismo número de muestra válida. Esto nos indica que dichos ficheros son muestras recogidas dentro de los 12 sg. de muestreo y separados posteriormente en sendos ficheros.

En el caso de que el directorio recoja ficheros válidos, éstos serán mostrados en el siguiente recuadro:



En el caso contrario aparecerá un cuadro de diálogo indicando tal situación.



En el mensaje se puede apreciar que además de indicar que no se ha encontrado ningún archivo es posible que no se cargue ninguno, aunque tenga la codificación correcta, si sólo existe un único archivo. Esto es debido a que los archivos deben ir emparejados – acción + reposos, si no es así, el análisis no tiene

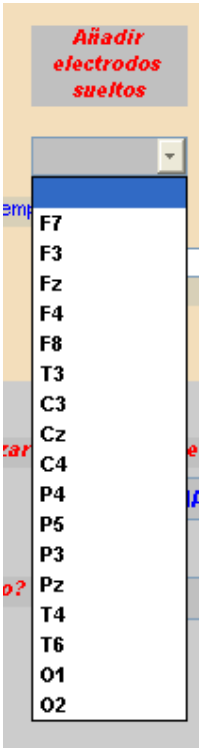
sentido ya que se trata de extraer las componentes independientes de ambas acciones para posteriormente ser introducidas en la red LVQ.

Una vez seleccionado el directorio correcto, aparecerán los nombres de la colección, sin el carácter final de A/P y R, ya que al seleccionar un fichero a cargar, se seleccionan ambos – acción y reposo.

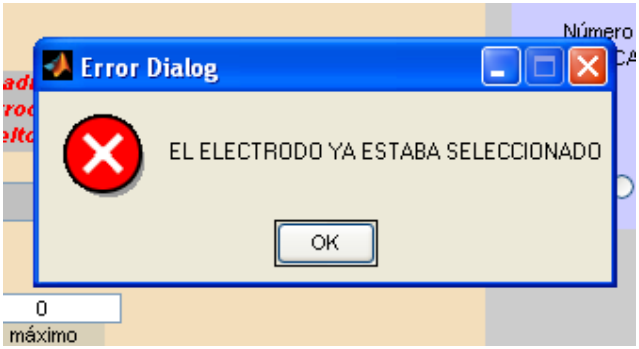
El siguiente paso es seleccionar el grupo de electrodos a estudiar.

Como se puede comprar es posibles seleccionar distintos grupos, tal y como se muestra en al anterior figura. La base para permitir la selección de distintos grupos de electrodos es para posibilitar las distintas opciones en las que es posible agrupar las señales que más nos acerquen a la consecución de un resultado idóneo. Dependiendo del objetivo principal del mismo, una correcta selección puede conllevar un resultado de éxito.

Una vez seleccionado el grupo de electrodos, si no se ha seleccionado Todos, es posible añadir otros electrodos individualmente, añadiendo esta muestra a la matriz inicial de estudio.



Se irán seleccionando los electrodos uno a uno. En el caso de que dicho electrodo estuviera ya seleccionado aparecería el siguiente aviso:



El siguiente paso sería seleccionar el rango de tiempos. El sentido de esta selección estriba en considera que los primeros milisegundos del comienzo de cualquier acción no sean relevantes o al revés, que los últimos no sean considerados. Por este motivo se da opción a establecer el rango de tiempos a introducir en la matriz de datos iniciales.

Aunque el rango de tiempos es de 0 a 2000 msg. la primera muestra se establece en 7,8 sg. por tanto si se selecciona un rango inferior o superior a ese valor, se mostrará el siguiente aviso:

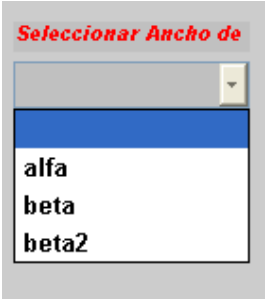
La siguiente selección corresponde a funcionalidades de visualización para comprobación visuales de las señales a estudio:

Tanto en la ***Seleccionar señal a visualizar (gráfico Amplitud/tiempo)*** como en ***Frecuencias (densidad de frecuencias)*** muestran las siguientes posibilidades:

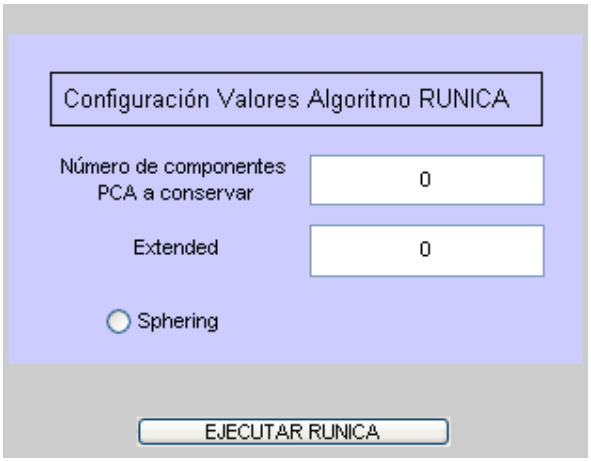
ORIGINAL: Tal y como se recogen la señales EEG.

ICA: Una vez realizada la operación de extracción de componentes independientes podrán mostrarse las señales resultantes.

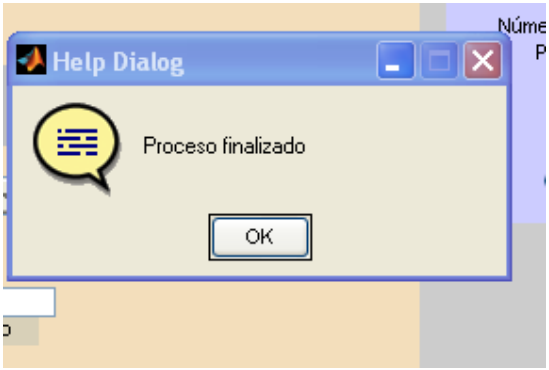
FILTRADA: Es posible filtrar la señal, antes de ser introducida para la extracción de componentes independientes, en tres tipos de filtros, seleccionables a través de ***Seleccionar Ancho de Banda***.



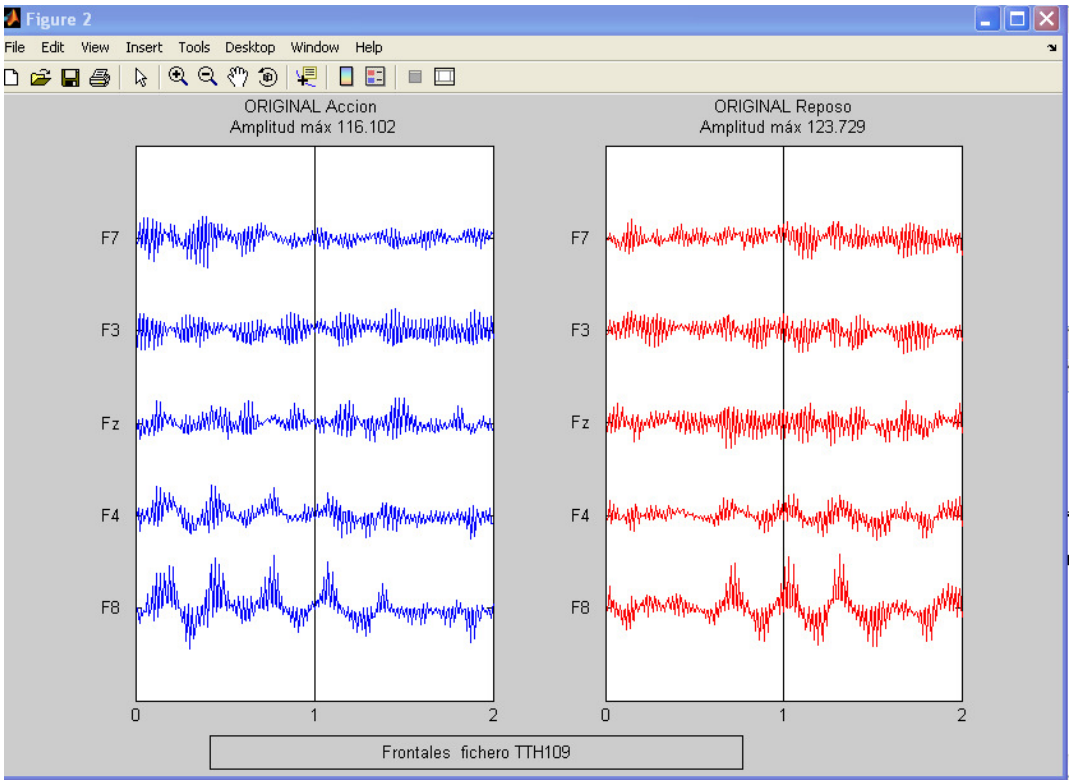
Finalmente, una vez seleccionado y/o modificado la señal a estudio, es posible seleccionar los parámetros de la función de extracción de componentes independientes – RUNICA – a través de siguiente cuadro de selección:



Una vez determinados los parámetros a introducir en la red neuronal, se pulsará EJECUTAR RUNICA para realizar la operación, dicha operación finaliza en el momento en que aparezca el siguiente diálogo:



Ya será posible visualizar las distantes señales para su comparación:



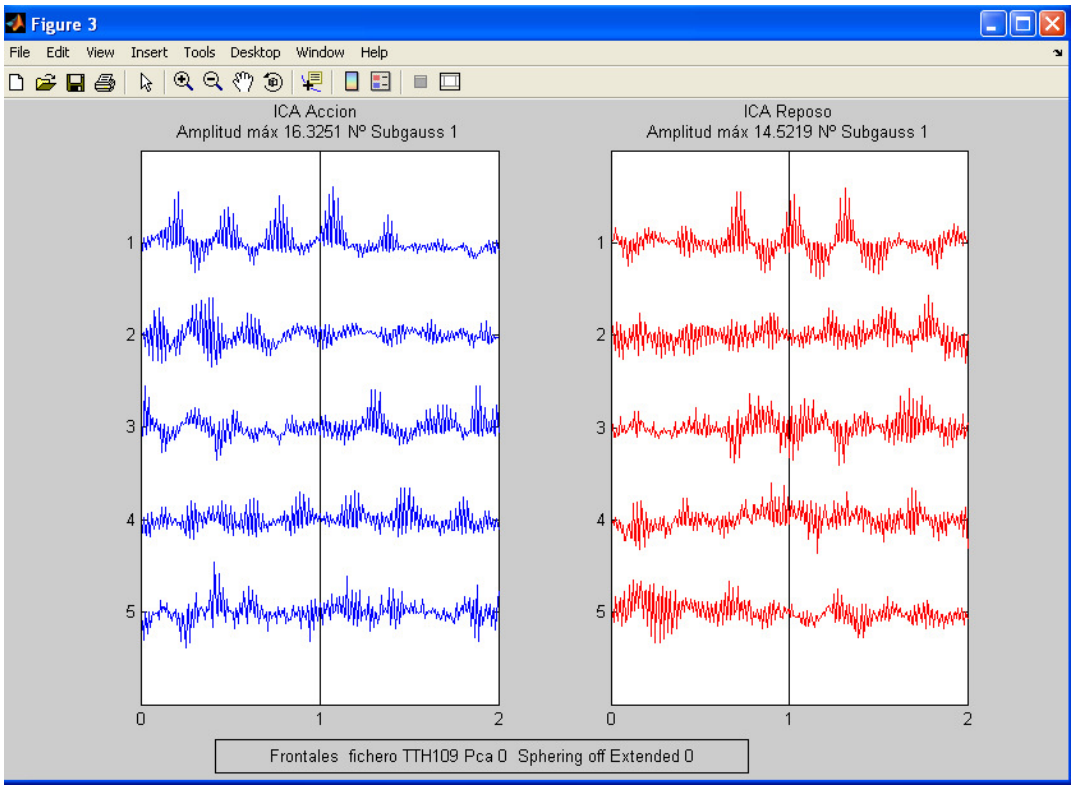
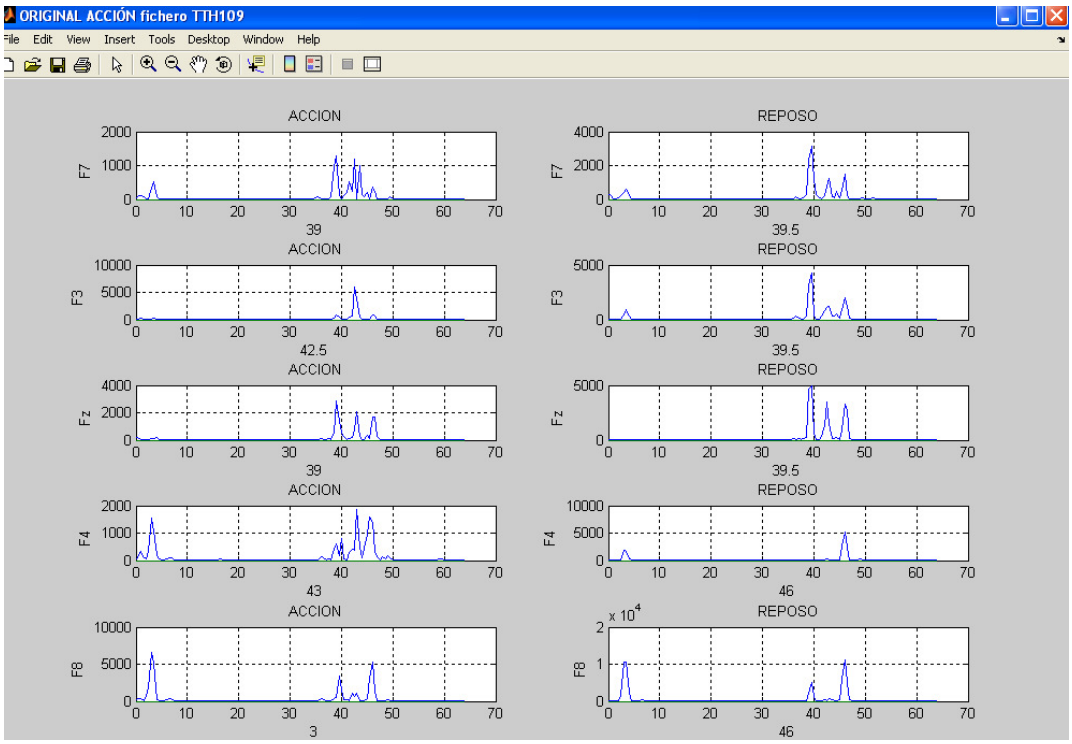
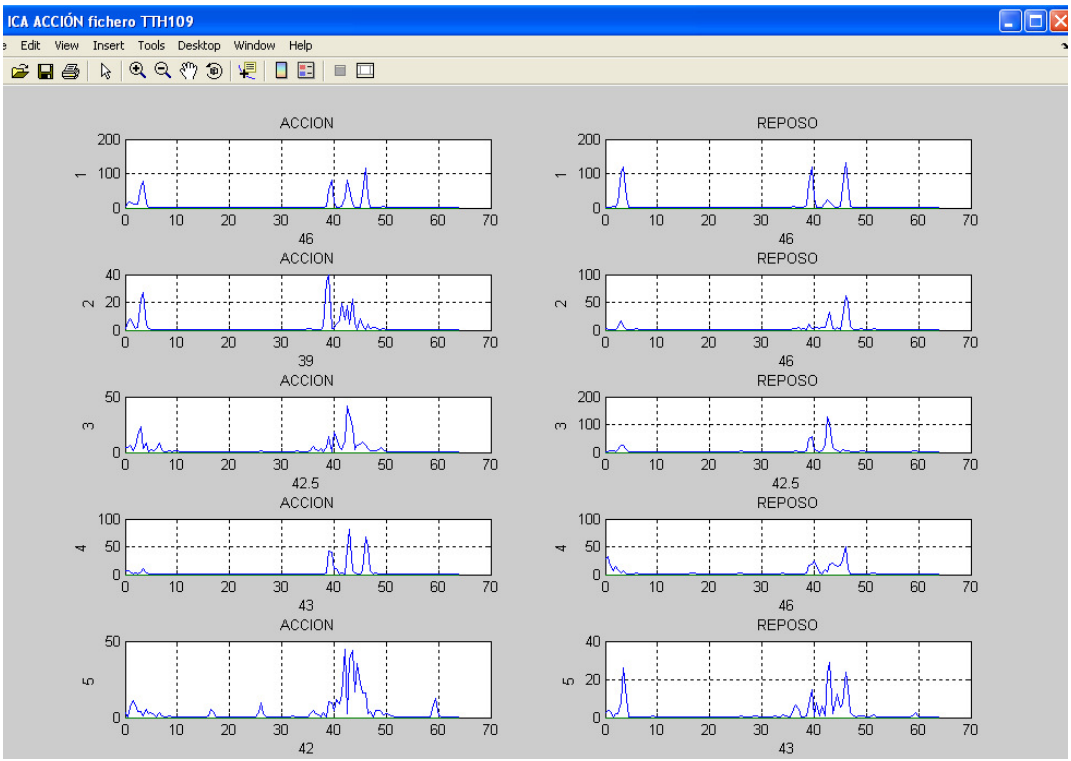
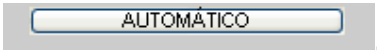


Figure 3
File Edit View Insert Tools Desktop Window Help
ICA Accion
Amplitud máx 16.3251 Nº Subgauss 1
ICA Reposo
Amplitud máx 14.5219 Nº Subgauss 1
Frontales fichero TTH109 Pca 0 Sphering off Extended 0


ORIGINAL ACCIÓN fichero TTH109
File Edit View Insert Tools Desktop Window Help
ACCION
REPOSO
F7
F3
Fz
F4
F8



Una vez comprobado visualmente los resultados, es posible realizar la misma operación para toda la colección de muestras. En ese caso se pulsaría el botón de 

. Dicha operación no tendría sentido si no se almacenarán los resultados por tanto es importante marcar la casilla de guardar.

El resultado será almacenado en el siguiente directorio por defecto e invariable: 'c:\muestras\resultado'. Si no existe, será creado.

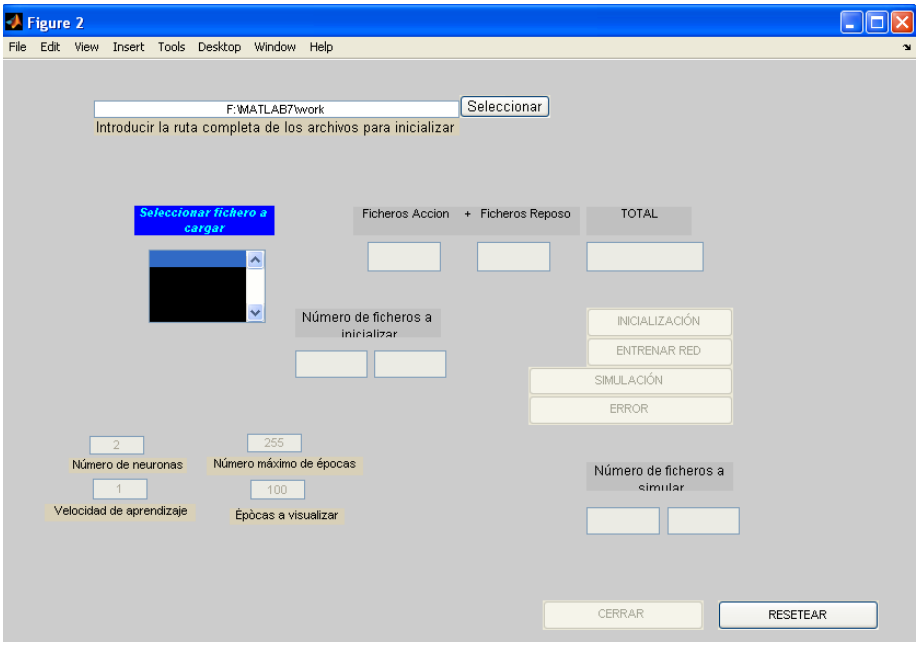
Con ello posibilitamos el acceso de la colección de resultado para su posterior inserción en la red LVQ de discriminación de acciones.

Una vez finalizado todo el proceso, los ficheros resultados serán nombrados según la siguiente nomenclatura:

TTHnºmuestraELECTRODOSA/R.ASC.

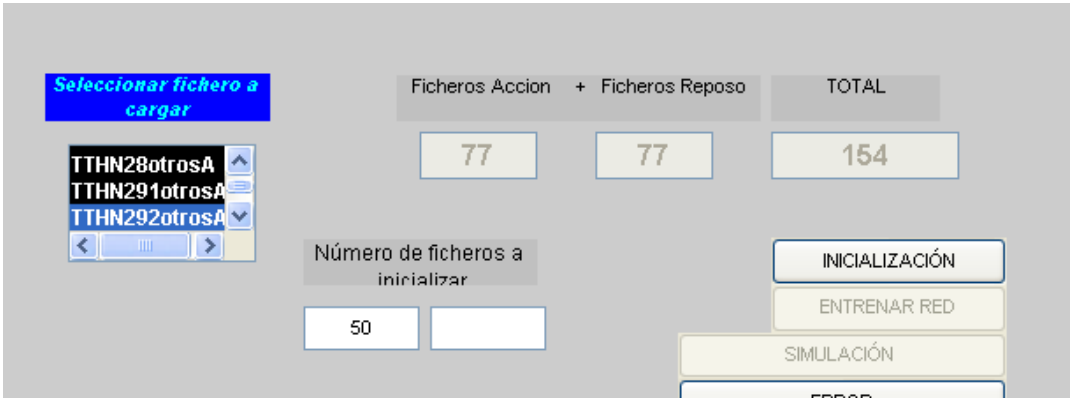
EJEMPLO: TTH109frontalA.ASC (fichero resultado de selección sólo frontales, muestra 109, fichero acción)

El siguiente paso ya sería introducir los ficheros resultado en la red LVQ, pulsando 



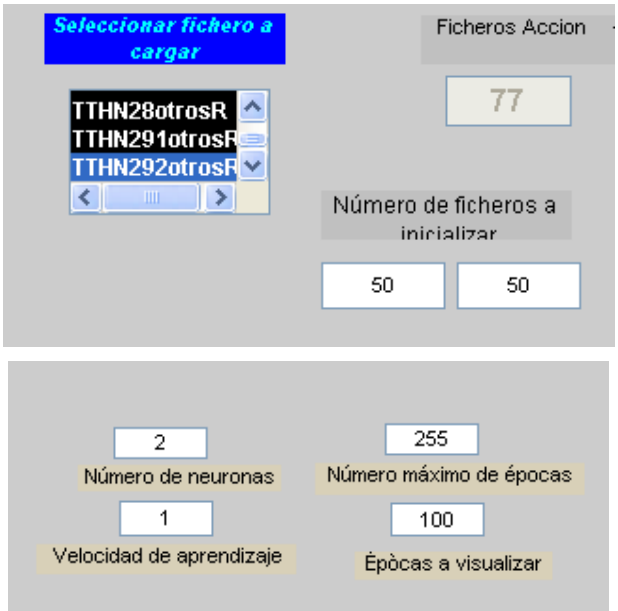
Como se puede apreciar la dinámica de selección es la misma que en el caso de Análisis de Componentes Independientes.

Primeramente se debe seleccionar el directorio donde se ubican los ficheros a entrenar.



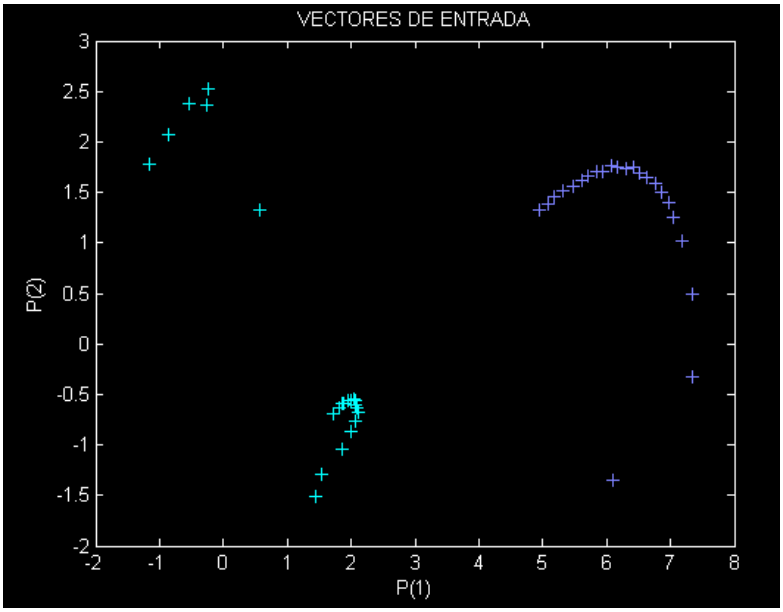
Aparecen todos los ficheros resultado pero esta vez sí están diferenciados por su Acción, con la letra correspondiente A/R. También se indica el número de ficheros existentes. Es de vital importancia este dato ya que para el aprendizaje de la red LVQ se deberá seleccionar por lo menos TOTAL – 1 fichero de los existentes. En principio se deja la posibilidad de seleccionar un número distinto de ficheros de acción y reposo, aunque lo lógico entrenar el mismo número de muestras de ambos tipos.

Una vez seleccionado, el cursor de *Seleccionar fichero a cargar* irá recorriendo los ficheros hasta el número seleccionado. Una vez terminado ambos procesos es posible cambiar los parámetros de inicialización y entrenamiento de la red:

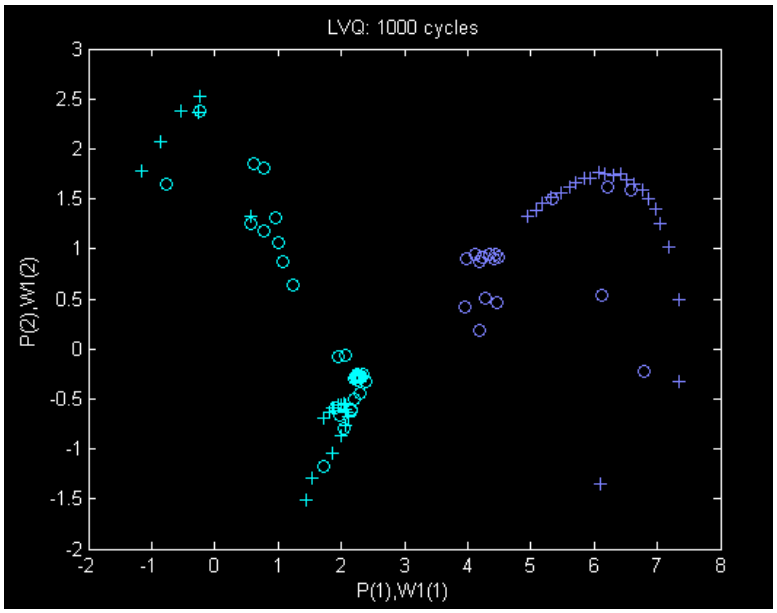


Para la inicialización pulsar 

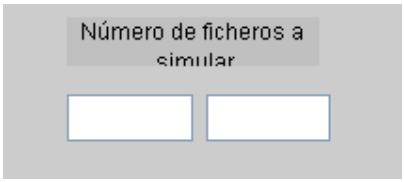
Una vez realizada la inicialización aparece el gráfico de vectores:



Pudiendo ver es espacio de ocupación de ambos resultados. En ese momento estará activo el botón de entrenamiento 



Por último, para poder comprobar el resultado de la red de aprendizaje se procede a la simulación. Con ello se seleccionan el número de ficheros a introducir en la red.



Al igual que en el momento de entrenar, al introducir el número se irán desplazando el cursor según se vayan cargando y simulando los ficheros. El primer fichero a simular será el último +1 a entrenar.

Un vez terminado el proceso, se puede comprobar el error cometido en la simulación.

En este caso, lo ficheros de acción se han clasificado todos correctamente y los reposo ninguno, por tanto el experimento no tendría valor, ya que no se ha podido conseguir distinguir ambas acciones.

IX. CÓDIGO APLICACIONES

FUNCIÓN PROCESO DE ENTRADA ICA-INFOMAX

```
function accion(accion,varargin)
global auto Muestras fs tiempomax;
auto=0;
Muestras=256;
fs=128;
tiempomax=2000;%en milisegundos

if nargin<1,
  accion='iniciali';
end;

feval(accion,varargin{:});
return


function iniciali()
global nomb menus indice;

%en el caso de que prueba ya estuviera ejecutándose traerlo a
primer plano
h = findobj(allchild(0));
if ~isempty(h)
  figure(h(1))
  return
end

screenD = get(0, 'ScreenDepth');
if screenD>8
  grayres=256;
else
  grayres=128;
end
nomb='  ';
d=what;
actual=d.path;
%figura principal
a = figure(...
  'Color',[0.8 0.8 0.8], ...
        'Colormap',gray(grayres), ...
    'NumberTitle' ,'off',...
  'Position',[68 120 834 533], ...
  'pointer','arrow',...
        'name','ACCION - DECONVOLUCIÓN SEÑALES MEDIANTE
ICA');

%menu para seleccionar los archivos
```

```
menu = uicontrol('Parent',a, ...
         'Units','normalized', ...
        'FontUnits','centimeters', ...
          'BackgroundColor',[0 0 0], ...
        'FontSize',0.4, ...
        'FontWeight','bold', ...
        'ForegroundColor',[1 1 1], ...
        'Position',[0.38 0.62 0.2 0.09], ...
        'String',nomb, ...
        'Style','listbox', ...
        'Tag','Listbox1', ...
          'Callback','accion(''cargar'')');

menu19 = uicontrol('Parent',a, ...
        'Units','normalized',...
        'BackgroundColor',[0.847059 0.815686 0.721569], ...
        'Position',[0.81  0.75 0.15 0.051],...
        'String','0',...
        'Style','Edit', ...
        'Tag','pca',...
        'Enable','on');
textomenu19 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'BackgroundColor',[0.8 0.8 1], ...
        'Position',[0.65  0.75 0.15 0.051],...
        'String','Número de componentes PCA a conservar', ...
        'Style','text', ...
        'Tag','StaticText19');

menu20 = uicontrol('Parent',a, ...
        'Units','normalized',...
        'Position',[0.68  0.61 0.15 0.051],...
        'BackgroundColor',[0.8 0.8 1], ...
        'String','Sphering',...
        'Style','radiobutton',...
        'Tag','Sphering',...
        'Enable','on',...
        'Value',[0.0]);

menu21 = uicontrol('Parent',a, ...
        'Units','normalized',...
        'Position',[0.81  0.68 0.15 0.051],...
        'String','0',...
        'Style','Edit', ...
        'Tag','Extended',...
        'Enable','on');
textomenu21 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'BackgroundColor',[0.8 0.8 1], ...
        'Position',[0.65  0.67 0.15 0.051], ...
        'String','Extended', ...
        'Style','text', ...
```

```
        'Tag','StaticText19');

annotation2 = annotation(a,...
            'rectangle',[0.6199 0.591 0.3657 0.3287],...
            'FaceColor',[0.8 0.8 1],...
            'LineWidth',2,...
            'EdgeColor',[0.6784 0.9216 1],...
            'LineStyle','none');

annotation3 = annotation(a,'textbox',...
            'Position',[0.6451 0.8311 0.3141 0.05103],...
            'FitHeightToText','off',...
            'String',{'Configuración Valores Algoritmo RUNICA'});

 textomenu = uicontrol('Parent',a, ...
            'Units','normalized', ...
            'BackgroundColor',[0 0 1], ...
            'FontSize',8, ...
            'FontAngle','italic', ...
            'FontWeight','bold', ...
            'ForegroundColor',[0 1 1], ...
            'Position',[0.38 0.73 0.2 0.03], ...
            'String','Seleccionar fichero a cargar', ...
            'Style','text', ...
             'Tag','StaticText1');


%Menu para seleccionar el grupo de electrodos
menu2 = uicontrol('Parent',a, ...
        'Units','normalized', ...
            'FontUnits','centimeters', ...
            'BackgroundColor',[0.752941 0.752941 0.752941], ...
            'FontSize',0.3, ...
            'FontWeight','bold', ...
            'ForegroundColor',[0 0 1], ...
            'Position',[0.03 0.24 0.2 0.4], ...
             'String',['          ';'Todos      ';'Frontales  ';'Parietales
            ';'Centrales  ';...
            'Temporales ';'Occipitales'],...
            'Style','popupmenu', ...
            'Tag','PopupMenu2',...
            'Callback','accion(''electrodos'')');
textomenu2 = uicontrol('Parent',a, ...
            'Units','normalized', ...
            'BackgroundColor',[0.752941 0.752941 0.752941], ...
            'FontAngle','italic', ...
            'FontWeight','bold', ...
            'ForegroundColor',[1 0 0], ...
            'Position',[0.03 0.67 0.2 0.06], ...
            'String','Seleccionar grupo de electrodos a estudiar', ...
            'Style','text', ...
              'Tag','StaticText2');
```

```
%Menu para añadir electrodos sueltos
menu3 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'FontUnits','centimeters', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'FontSize',0.282008, ...
        'FontWeight','bold', ...
        'ForegroundColor',[0 0 1], ...
        'Position',[0.25 0.56 0.1 0.08], ...
        'String',['  ';'F7';'F3';'Fz';'F4';'F8';'T3';'C3';'Cz';'C4';'P4';'P5';...
    'P3';'Pz';'T4';'T6';'O1';'O2'],...
        'Style','popupmenu', ...
        'Tag','PopupMenu3',...
        'Callback','accion(''anadir'')');
textomenu3 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'FontAngle','italic', ...
        'FontWeight','bold', ...
        'ForegroundColor',[1 0 0], ...
        'Position',[0.25 0.67 0.1 0.08], ...
        'String','Añadir electrodos sueltos', ...
        'Style','text', ...
        'Tag','StaticText3');

%Botón que ejecuta el algoritmo ICA
menu4 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'Callback','accion(''Activar2'')', ...
        'Position',[0.7 0.51 0.2 0.03], ...
        'String','EJECUTAR RUNICA', ...
        'Tag','Pushbutton4');

%menu rango de tiempo accion
%mínimo
menu5 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'BackgroundColor',[1 1 1], ...
        'Position',[0.07 0.50 0.1 0.03], ...
        'Style','edit', ...
        'Callback','accion(''Rango1a'')', ...
        'Tag','EditText5');
  textomenu5 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'BackgroundColor',[0.847059 0.815686 0.721569], ...
        'Position',[0.08 0.47 0.08 0.03], ...
        'String','mínimo', ...
        'Style','text', ...
        'Tag','StaticText5');

%máximo
menu6 = uicontrol('Parent',a, ...
```

```
        'Units','normalized', ...
        'BackgroundColor',[1 1 1], ...
        'Position',[0.27 0.50 0.1 0.03], ...
        'Style','edit', ...
        'Callback','accion(''Rango2a'')', ...
        'Tag','EditText6');
textomenu6 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'BackgroundColor',[0.847059 0.815686 0.721569], ...
        'Position',[0.28 0.47 0.08 0.03], ...
        'String','máximo', ...
        'Style','text', ...
        'Tag','StaticText6');

texto = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'BackgroundColor',[0.847059 0.815686 0.721569], ...
        'ForegroundColor',[0 0 1], ...
        'Position',[0.14 0.55 0.15 0.03], ...
        'String','Rango de Tiempos', ...
        'Style','text', ...
        'Tag','StaticText');

%Menu para cargar los archivos del directorio seleccionado
menu7 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'BackgroundColor',[1 1 1], ...
        'Position',[0.1 0.9 0.4 0.03], ...
        'Style','edit', ...
        'string',actual,...
        'Callback','accion(''extraer'')', ...
        'Tag','EditText7');
textomenu7 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'FontSize',10, ...
        'BackgroundColor',[0.847059 0.815686 0.721569], ...
        'Position',[0.1  0.87 0.4 0.03], ...
        'String','Introducir la ruta completa de los archivos a cargar',
...
        'Style','text', ...
        'Tag','StaticText7');

menu18 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'FontSize',10, ...
        'Callback','accion(''extraer'')', ...
        'Position',[0.5 0.9 0.1 0.04], ...
        'String','Seleccionar', ...
        'Tag','Pushbutton13');

%Menu de visualización de señales
menu8 = uicontrol('Parent',a, ...
        'Units','normalized', ...
```

```
        'FontUnits','centimeters', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'FontSize',0.3, ...
        'FontWeight','bold', ...
        'ForegroundColor',[0 0 1], ...
        'Position',[0.075 0.27 0.15 0.03], ...
        'String',['ORIGINAL';'ICA     ';'FILTRADA'],...
        'Style','popupmenu', ...
        'Tag','PopupMenu8',...
        'Callback','accion(''visual'')');
textomenu8 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'FontAngle','italic', ...
        'FontWeight','bold', ...
        'ForegroundColor',[1 0 0], ...
        'Position',[0.05 0.31 0.2 0.03], ...
        'String','Seleccionar señal a visualizar', ...
        'Style','text', ...
        'Tag','StaticText8');


% SACAR COMPONENTES EN FRECUENCIA
menu10 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'FontUnits','centimeters', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'FontSize',0.3, ...
        'FontWeight','bold', ...
        'ForegroundColor',[0 0 1], ...
        'Position',[0.3 0.27 0.15 0.03], ...
        'String',['ORIGINAL';'ICA     ';'FILTRADA'],...
        'Style','popupmenu', ...
        'Tag','PopupMenu10',...
        'Callback','accion(''frecu'')');
textomenu10 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'FontAngle','italic', ...
        'FontWeight','bold', ...
        'ForegroundColor',[1 0 0], ...
        'Position',[0.3 0.31 0.15 0.03], ...
        'String','Frecuencias', ...
        'Style','text', ...
        'Tag','StaticText10');

menu11 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'FontUnits','centimeters', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'FontSize',0.3, ...
        'FontWeight','bold', ...
        'ForegroundColor',[0 0 1], ...
```

```
        'Position',[0.3 0.17 0.15 0.03], ...
        'String',['NO              ';'SI              ';'AÑADIR AL
ARCHIVO'],...
        'Style','popupmenu', ...
        'Tag','PopupMenu11',...
        'Callback','accion(''guardar'')');
textomenu11 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'FontAngle','italic', ...
        'FontWeight','bold', ...
        'ForegroundColor',[1 0 0], ...
        'Position',[0.05 0.17 0.2 0.03], ...
        'String','¿Guardar archivo resultado?', ...
        'Style','text', ...
        'Tag','StaticText11');

menu13 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'Callback','redn', ...
        'Position',[0.7 0.37 0.2 0.03], ...
        'String','PANEL RED NEURONAL', ...
        'Tag','Pushbutton13');

menu14 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'Callback','accion(''cerrar'')', ...
        'Position',[0.7 0.08 0.2 0.03], ...
        'String','CERRAR', ...
         'Tag','Pushbutton13');

menu15 = uicontrol('Parent',a, ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'Units','normalized', ...
        'Callback','accion(''entrar'')', ...
        'Enable','on', ...
        'FontSize',10, ...
        'FontWeight','bold', ...
        'Position',[0.5 0.27 0.15 0.03], ...
        'String',['     ';'<40Hz';'alfa ';'beta ';'beta2'], ...
        'Style','popupmenu', ...
        'Tag','PopupMenu15', ...
        'Value',1);
textmenu15 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'FontAngle','italic', ...
        'FontWeight','bold', ...
        'ForegroundColor',[1 0 0], ...
        'Position',[0.5 0.31 0.15 0.03], ...
        'String','Seleccionar Ancho de Banda', ...
        'Style','text', ...
        'Tag','StaticText');
```

```
menu17 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'Callback','accion(''auto'')', ...
        'Position',[0.7 0.47 0.2 0.03], ...
        'String','AUTOMÁTICO', ...
        'Tag','Pushbutton17');

annotation1 = annotation(a,...
         'rectangle',[0.02158 0.394 0.5815 0.5516],...
         'FaceColor',[0.9529 0.8706 0.7333],...
         'LineWidth',2,...
         'EdgeColor',[0.03922 0.1412 0.4157],...
         'LineStyle','none');

menus=[menu menu2 menu3 menu4 menu5 menu6...
     menu7 menu8 menu10 menu11...
     menu13 menu14 menu15 menu17 menu18 menu19...
     menu20 menu21]

set(a,'UserData',menus,'Visible','on');
set(a,'resize','on');

%drawnow
set(menus, 'Enable', 'off');
set(menu7,'Enable', 'on');
set(menu18,'Enable', 'on');

drawnow
set(a,'HandleVisibility','Callback');

return

%función que recorre el directorio donde se encuentran los ficheros a
examinar
function extraer(directo)
global atual nomb menus archivos ;
global electrodo cargo totalficheros accion1;

accion1=0;
cargo=0;
if nargin<1
  directo = gcf;
end

   actual = uigetdir();

   set(menus(7), 'String', actual)
   set(directo,'Pointer','arrow');
   menu = get(directo,'UserData');
   v=get(menus(7),{'value','string'});
   archivos= deblank(v{2});
   existe=exist(archivos,'dir');
```

```
    if (isempty(archivos) || existe ~=7)
        errordlg(sprintf('Directorio no seleccionado o erroneo'));
        return;
    end;

directorio=dir(archivos);
[fil,col]=size(directorio);
for i=3:fil
    nombres(i-2)=cellstr(directorio(i).name);
end;

%averiguamos los ficheros que concuerdan con fichero a cargar
indice=strncmp('TT',nombres,length('TT'));
  if (isempty(indice)| indice==0)
        set(menus(6), 'String', sprintf('%i', get(menus(6), 'Value')))
        errordlg(sprintf('No se ha encontrado ningún archivo de carga o
sólo existe uno'));
        return;
    end;

nombre1=nombres(indice);
nombre2=char(nombre1);
%los ficheros seleccionados los ordenamos alfabéticamente
[nombres,I]=sortrows(nombre2);
nombre=nombre2(I,:);
[totalficheros,total]=size(nombre);

%agrupamos los nombre de dos en dos - accion y reposo
j=0;
for i=1:totalficheros
    posic=findstr(nombre(i,:),'A.');

   if isempty(posic)
       posic=findstr(nombre(i,:),'P.');

       if ~isempty(posic)
           cargo=1;
       end;

   end;

    if ~isempty(posic)
       pos=posic-1;
         if i==totalficheros
            break;
     end;
        posi=findstr(nombre(i+1,:),'R.');
        if ~isempty(posi)
            j=j+1;
            nombress(j,1:pos)=nombre(i,1:pos);
       end;
   end;
```

```
   if i==totalficheros
       break;
    end;
  i=i+1;
end;

nomb=char(nombress);
[accion1,columm]=size(nomb);

set(menus(1), 'Enable', 'on');
set(menus(1),'string',nomb);
set(menus(7), 'Enable', 'off');
set(menus(14),'Enable', 'on');
set(menus(15), 'Enable', 'off');
set(menus(16),'Enable', 'on');
set(menus(17),'Enable', 'on');
set(menus(18),'Enable', 'on');

drawnow

return;

%FUNCIÓN QUE CARGA LOS FICHEROS
function cargar(archivo)
global fichero1 fichero2 name1 nomb totalficheros archivos;
global auto menus cargo indic;


clear global canaloriginal s activations retardada origen...
    componenntesa componenntesr ordena ordenr;
lasterr('');
if nargin<1
   archivo = gcf;
end

set(archivo,'Pointer','arrow');
menu = get(archivo,'UserData');
v=get(menus(1),{'value','string'});
name1 = deblank(v{2}(v{1},:));
[fil,col]=size(nomb);
[filmax,colmax]=size(name1);

drawnow

for i=1:fil

    if strcmp(name1,nomb(i,1:colmax))
        %fichero acción
        if cargo~=1
            nom=strcat(archivos,'\',nomb(i,:),'A.ASC');
        else  nom=strcat(archivos,'\',nomb(i,:),'P.ASC');
        end;
        load(nom)
```

```
        if cargo~=1
            nombr=strcat(nomb(i,:),'A');
        else
            nombr=strcat(nomb(i,:),'P');
        end;
        nombr
        [fichero1]=eval(nombr);
        fprintf(nombr);

        if strcmp(name1,nomb(i,1:colmax))
        %fichero reposo
        nom=strcat(archivos,'\',nomb(i,:),'R.ASC');
        load(nom);
        nombr=strcat(nomb(i,:),'R');
        [fichero2]=eval(nombr);
    end;

        if isempty(fichero1)
            indic=indic+1;
            feval('auto',indic);
        end;
    end;
end;

if auto==1
else
set(menus(2), 'Enable', 'on');
set(menus(2),'value',1);
set(menus(3),'value',1);

set(menus(11),'value',1);
set(menus(15),'value',1);

if ~isempty(fichero1)
feval('electrodos');
end;
end;

%%%%.....hasta aquí hemos cargado en el menú los archivos a
analizar.....%%
%.....%%

%SELECCION DE FICHERO A ANALIZAR.....
function [name,nombre]=entradas()
global primera data1 data2 fichero1 fichero2 electrodo name1 name2
name;
global salida1 salida2 activations1 activations2;
global menus minimo aa;

activations1=[];
activations2=[];

name3=('fichero ');
```

```
name4=('  electrodos ');
name=[name3 name1];
nombre=[name4 name2];

%accion
salida1=modificar(fichero1);%transforma la matriz
[tiempos,electrodos]
%eliminamos la referencia común
salida1 = averef(salida1);
%eliminamos electrodos nº 1,2,21 y 20
salida1=salida1(:,3:19);
%cambias los índices
salida1=salida1(:,electrodo);
data1=salida1;
data1=data1'; %transformamos la matriz [electrodos, tiempos]

%reposo
salida2=modificar(fichero2);%transforma la matriz
[tiempos,electrodos]
%eliminamos la referencia común
salida2 = averef(salida2);
%eliminamos electrodos nº 1,2,21 y 20
salida2=salida2(:,3:19);
%cambias los índices
salida2=salida2(:,electrodo);
data2=salida2;
data2=data2';

set([menus(3) menus(4) menus(5) menus(6) menus(7) menus(8)
menus(9) menus(10)...
        menus(11) menus(12) menus(13)  menus(14)] , 'Enable',
'on');
set(menus(7), 'Enable', 'off');

if ~isempty(minimo)
   feval('Rango2a')
end;


function entrar()
global menus activations1 activations2;
global minimo maximo salida1 salida2 filtrada1 filtrada2;
global data1 data2 origen1 origen2 resultados1 resultados2 filtro;

bajo=0;
alto=0;


if (isempty(origen1))
  errordlg(sprintf('NO SE HAN ESTABLECIDO LOS RANGOS DE
TIEMPO'));
  set(menus(13),'value',1);
  return;
```

```
end;

nombre=get(menus(13),{'value','string'})
filtro = deblank(nombre{2} (nombre{1},:))

final='no'

if isempty(resultados1)
   matriz1=origen1;
   matriz2=origen2;
else matriz1=resultados1;
    matriz2=resultados2;
   final='si';
end;

aa1=matriz1;
aa2=matriz2;
switch filtro
case '',
   aa1=[];
   filtrada1=[];
   aa2=[];
   filtrada2=[];
case '<40Hz',
   alto=35;
   bajo=0;
case 'alfa',
   alto=13;
   bajo=7;
case 'beta',
   alto=18;
   bajo=13;
case 'beta2',
   alto=30
   bajo=18
end;
sal1=[];
sal2=[];
if ~isempty(aa1)
   [sal1] = eegfilt(aa1,128,bajo,alto,0,0);
   filtro1=sal1;
   filtrada1=filtro1;
   data1=filtrada1;
   if final=='si'
      resultados1=filtrada1;
   end;

   if ~isempty(aa2)
   [sal2] = eegfilt(aa2,128,bajo,alto,0,0);
   filtro2=sal2;
   filtrada2=filtro2;
   data2=filtrada2;
   if final=='si'
```

```
    resultados2=filtrada2;
  end;
    helpdlg('FILTRO SELECCIONADO')
  else
     data1=original1;
     data2=original2;
     resultado1=[];
     resultado2=[];
  end;
return;
end;



%FUNCIÓN PARA SELECCIONAR EL GRUPO DE ELECTRODOS A
ANALIZAR
function electrodos(electro)
global electr electrodo name2 name222 canal canaloriginal;
global auto menus e nombreelectro;

if nargin<1
  electro = gcf;
end
set(electro,'Pointer','arrow');
menu = get(electro,'UserData');
nombre=get(menus(2),{'value','string'})
name2 = deblank(nombre{2} (nombre{1},:));

if ~strcmp(canaloriginal,'otros.m')


  switch name2
  case 'Todos',
    electrodo=[1 2 3 4 5 7 8 9 10 11 12 13 14 15 16 17];
    canal='Todos2.m';
  case 'Frontales',
    electrodo=[1 2 3 4 5];
    canal='frontal2.m';
  case 'Parietales',
    electrodo=[10 11 12 13];
    canal='parietal2.m';
    %nombreelectro='P4..P5..P3..Pz..';
  case 'Centrales',
    electrodo=[7 8 9];
    canal='central2.m';
    %nombreelectro='C3..Cz..C4..';
  case 'Temporales',
    electrodo=[6 14 15];
    canal='temporal2.m';
    %nombreelectro='T3..T4..T6..';
  case 'Occipitales',
    %    electrodo=[17];
```

```
      electrodo=[16 17];
      canal='occipital2.m';
      %nombreelectro='O1..O2..';

 end;

end;
    if isempty(name2)
      canal='';
      e=errordlg(sprintf('INTRODUZCA LOS ELECTRODOS'));
      return
    end;

   canaloriginal=canal;
   fprintf('ELECTRODOS SELECCIONADOS\n');
   set(menus(14), 'Enable', 'on');
   name222=[];
   fid=fopen('otros.m','w');
   fclose(fid);
   feval('anadir');
   %end;

%FUNCIÓN PARA SELECCIONAR EL GRUPO DE ELECTRODOS A
ANALIZAR
function anadir(electro)
global electrodo name222;
global nombreelectr canal canaloriginal menus retardada;

if nargin<1
  electro = gcf;
end

if strcmp(canal,'unosolo')
   canal=canaloriginal;
end;
[filas,columnas]=size(electrodo);

if ~isempty(canal)
   fid=fopen(canal)
   [nuevos]=fscanf(fid,'%d %f %f %s',[7,columnas])
   fclose(fid);
   nombreelectr=nuevos';
end;


set(electro,'Pointer','arrow');
menu = get(electro,'UserData');
nombre=get(menus(3),{'value','string'});
name22 = deblank(nombre{2} (nombre{1},:));
drawnow

 switch name22
 case 'F7',
```

```
   electrod=[1];
   nombreelectrodo='1 -54 .352 F7..';
case 'F3',
   electrod=[2];
      nombreelectrodo=' 2            -39    .231      F3..';
case 'Fz',
   electrod=[3];
   nombreelectrodo=' 3             0     .181      Fz..';
case 'F4',
   electrod=[4];
      nombreelectrodo=' 4            39    .231      F4..';
case 'F8',
   electrod=[5];
      nombreelectrodo=' 5            54    .352      F8..';
case 'T3',
   electrod=[6];
      nombreelectrodo='6 -90 .352 T3..';
case 'C3',
   electrod=[7];
      nombreelectrodo= ' 7           -90    .181      C3..'
case 'Cz',
    electrod=[8];
       nombreelectrodo='8            0       0       Cz..';
case 'C4',
   electrod=[9];
      nombreelectrodo=' 9            90    .181      C4..';
case 'P4',
   electrod=[10];
      nombreelectrodo='10            142   .231      P4..';
case 'P5',
   electrod=[11];
      nombreelectrodo='11            -18    .352     P5..';
case 'P3',
   electrod=[12];
      nombreelectrodo='12            -142   .231    P3..';
case 'Pz',
   electrod=[13];
      nombreelectrodo='13            180   .181    Pz..';
case 'T4',
  electrod=[14];
     nombreelectrodo='14             90    .352   T4..';
case 'T6',
  electrod=[15];
     nombreelectrodo='15             126   .352 T6..';
case 'O1',
  electrod=[16];
     nombreelectrodo='16            -162   .352  O1..';
case 'O2',
  electrod=[17];
     nombreelectrodo='17             162   .352 O2..';
end;

 if ~isempty(name22)
```

```
    [fil,col]=size(electrodo);

    for i=1:col
      if electrodo(1,i)==electrod;
        igual=1;
        e=errordlg(sprintf('EL ELECTRODO YA ESTABA
SELECCIONADO\n'));

       return;
      else
         igual=0;
      end;

    end;          plus='+';

              name222=[name222 plus name22]

    if igual==0
      electrodo=[electrodo electrod];
      formato='%s';
    end;

    if ~strcmp(canaloriginal,'otros.m')
    [s,mess,messid]=copyfile(canaloriginal,'otros.m')
    end
    fid=fopen('otros.m','a+');
    fprintf(fid,formato,char(10));
    fprintf(fid,formato,nombreelectrodo);
    fclose(fid);
    canal='otros.m';
    canaloriginal=canal;
    set(menus(2), 'Enable', 'off');
    feval('entradas');
  end;

  if isempty(name22)
    feval('entradas');
  end;

  %RANGO ACCIÓN.....
function Rango1a()
global minimo tiempomax Muestras menus min;

if isempty(min)
   set(gcf,'Pointer','arrow');
   v=get(menus(5),{'value','string'});
   minimo1= str2num(v{2});
   drawnow
else minimo1=str2num(min);
  set(menus(5),'string',minimo1);
end;

if (isempty(minimo1) | ((minimo1 < 7.8) | (minimo1 > tiempomax)))
```

```
    errordlg(sprintf('Debe estar entre 7,8 and %i',tiempomax));
    return;
 end

minimo=minimo1*Muestras/tiempomax;
minimo=round(minimo);

%MAXIMO.....
function Rango2a()
global maximo minimo minimo1 maximo1;
global Muestras tiempomax;
global salida2 salida22 salida1 salida11;
global tamano menus origen1 origen2 max;

if isempty(max)
    set(gcf,'Pointer','arrow');
    menu = get(gcf,'UserData');
    v=get(menus(6),{'value','string'});
    maximo1= str2num(v{2});
else maximo1=str2num(max);
       set(menus(6),'string',maximo1);

end;

maximo=maximo1*Muestras/tiempomax;
maximo=round(maximo);
drawnow

if isempty(maximo1)
    errordlg(sprintf('Debe estar entre 7.8 and %i y ser mayor que el
valor mínimo %i',tiempomax,minimo1));
    return;
 end;
%salida1=salida1';% quitar

salida11=salida1(minimo:maximo,:);
tamano=maximo-minimo+1;
data1=salida11;
origen1=data1';
data1=data1';

%salida2=salida2';% quitar

salida22=salida2(minimo:maximo,:);
tamano=maximo-minimo+1;
data2=salida22;
origen2=data2';
data2=data2';

fprintf('RANGOS ESTABLECIDOS ACCIÓN Y REPOSO\n');

%FUNCIÓN ICA.....
function Activar2
```

```
global menus data1 data2 tamano electrodo filtrada;
global windex1 activations1 resultados1;
global auto windex2 activations2 resultados2;

global num_ica;
global Pca Spheringg Extended sub1 sub2;

     v=get(menus(16),{'value','string'});
       Pca=str2num(v{2});

    boton_estado = get(menus(17),'value');

    if boton_estado == 1
         Spheringg = 'on'; else
         Spheringg ='off';
     end;

 v=get(menus(18),{'value','string'});
 Extended=str2num(v{2});

if ~isempty(filtrada)
   data=filtrada;
end;

[fil,col]=size(data1);
fprintf('%i fila %i columna',fil,col)

if fil>17
   fil=15;
   data1=data1(1:fil,:);
end;

[fil,col]=size(data2);
fprintf('%i fila %i columna',fil,col)

if fil>17
   fil=15;
   data2=data2(1:fil,:);
end;

 if Pca == 0

[weights1,sphere1,activations1,bias,signs,lrates,y,sub1] =
runica2(data1,'sphering',Spheringg,'extended',Extended);
[weights2,sphere2,activations2,bias,signs,lrates,y,sub2] =
runica2(data2,'sphering',Spheringg,'extended',Extended);

 else
[weights1,sphere1,activations1,bias,signs,lrates,y,sub1] =
runica2(data1,'sphering',Spheringg,'pca',Pca,'extended',Extended);
[weights2,sphere2,activations2,bias,signs,lrates,y,sub2] =
runica2(data2,'sphering',Spheringg,'pca',Pca,'extended',Extended);
```

```
 end;

datamean = mean(data1')';% la media es sobre las columnas por eso
se traspone la matriz de datos
fprintf('%i media',datamean);

[activations1] = icaact(data1,weights1,sphere1,datamean);
[windex1,maxvar,maxframe,maxepoch,maxmap] =
compsort(data1,weights1,sphere1,1);

            activations1=activations1(windex1,:);
            resultados1=activations1;
            k=kurt(activations1');
            %fprintf('%i kurtosis',k);


datamean = mean(data2')';% la media es sobre las columnas por eso
se traspone la matriz de datos
fprintf('%i media',datamean);

[activations2] = icaact(data2,weights2,sphere2,datamean);
[windex2,maxvar,maxframe,maxepoch,maxmap] =
compsort(data2,weights2,sphere2,1);

            activations2=activations2(windex2,:);
            resultados2=activations2;
            k=kurt(activations2');
            %fprintf('%i kurtosis',k)
if(auto==0)
            helpdlg('Proceso finalizado');
end;

feval('guardar');

function frecu(senal)
global origen1 origen2 retardada resultados1 resultados2 activations1
activations2 name filtrada1 filtrada2;
global canaloriginal menus num_ica;


if nargin<1
  senal = gcf;
end

set(senal,'Pointer','arrow');
menu = get(senal,'UserData');
nombr=get(menus(9),{'value','string'});
name2 = deblank(nombr{2} (nombr{1},:));
drawnow

switch name2
case 'ICA',
   if isempty(activations1)
```

```
    errordlg(sprintf('AÚN NO SE HA REALIZADO ESTE PROCESO'));
  else
    [num_ica,col]=size(activations1);
    %canales='compone.m';
    canales=0;
    frecuencia(activations1,activations2,name2,name,canales);
  end;

case 'ORIGINAL',
    if isempty(origen1)
      errordlg(sprintf('AÚN NO HA SELECCIONADO LOS
ELECTRODOS\n O LOS TIEMPOS NO SON CORRECTOS'));
    else
      %[fil,col]=size(origen);
      frecuencia(origen1,origen2,name2,name,canaloriginal);
    end;

case 'FILTRADA',
  if isempty(filtrada1)
    errordlg(sprintf('AÚN NO SE HA REALIZADO ESTE PROCESO'));
  else
     frecuencia(filtrada1,filtrada2,name2,name,canaloriginal);
 end;
end;


%FUNCIÓN ESPECTRO DE LA SEÑAL - VISUALIZA ORIGINAL - ICA -
FILTRADA
function frecuencia(senal1,senal2,nombre1,nombre2,nombre3)
global ordena num_ica;
fs=128;
[filas,col]=size(senal1);

p1=senal1;%ACCION
p2=senal2;%reposo
j=filas;


 %TEXTO CURVAS
      if nombre3==0
         canales=[];
         valor=[];

         for i = 1:num_ica
            valor=i;
             canales=cat(1,canales,valor);
             [filas,column]=size(canales);
            column=1;

         end;
       else
         f=fopen(nombre3);
         canales = fscanf(f,'%d %f %f %s',[7 j]);
```

```
        canales = canales';
        canales = setstr(canales(:,4:7)); % convert ints to chars
        [r c] = size(canales);
        for i=1:r
              for j=1:c
              if canales(i,j)=='.',
                 canales(i,j)=' '; % convert dots to spaces
              end;
           end;
      end;
          fclose(f);

      end;


[filas,column]=size(canales);

v=figure;
set(v,'Position',[0 0 1000 650]);
nombre=[nombre1 ' ACCIÓN ' nombre2];
set(v,'name',nombre);
set(v,'numbertitle','off');
fila=zeros(filas,col);
posicion=1;


j=filas

for i=1:filas
   [Power,frqs]=spectrum(p1(i,:),[],[],[],fs);
   pow=abs(Power);
   subplot(j,2,posicion),plot(frqs,pow);
   [fil,colu]=max(pow);
   fila=colu(1,1);
   maximo(i)=fil(1,1);
   etiqueta=canales(i,1:column);
   title('ACCION')

   ylabel(etiqueta);
   frec=frqs(fila,1);
   xlabel(frec);
   grid;

   [Power,frqs]=spectrum(p2(i,:),[],[],[],fs);
   pow=abs(Power);
   subplot(j,2,posicion+1),plot(frqs,pow);
   [fil,colu]=max(pow);
   fila=colu(1,1);
   maximo(i)=fil(1,1);
   etiqueta=canales(i,1:column);
   title('REPOSO')
   ylabel(etiqueta);
   frec=frqs(fila,1);
```

```
   xlabel(frec);
   grid;
   posicion=posicion+2;
end;

[maximoa, ordena] = sort(maximo);

%VISUALIZACIÓN DE SEÑALES
function visual(senal)
global origen1 retardada1 resultados1 activations1 name Muestras
windex1 canal numcomp;
global origen2 retardada2 resultados2 activations2 windex2;

global canaloriginal m filtrada1 filtrada2 componenntes menus;

if nargin<1
  senal = gcf;
end

set(senal,'Pointer','arrow');
menu = get(senal,'UserData');
nombr=get(menus(8),{'value','string'});
name2 = deblank(nombr{2} (nombr{1},:));
drawnow

switch name2
case 'ICA',
   if isempty(activations1)
      errordlg(sprintf('AÚN NO SE HA REALIZADO ESTE PROCESO'));
   else
      [fil,col]=size(activations1);

      canales = 0;
      visualizar(activations1,activations2,name2,name,col,canales);
   end;

case 'ORIGINAL',
    if isempty(origen1)
       errordlg(sprintf('AÚN NO HA SELECCIONADO LOS
ELECTRODOS\n O LOS TIEMPOS NO SON CORRECTOS'));
    else
       [fil,col]=size(origen1);
       fprintf('fila %d columna %d',fil,col);
       visualizar(origen1,origen2,name2,name,col,canaloriginal);
    end;

case 'FILTRADA',

    if isempty(filtrada1)
       errordlg(sprintf('AÚN NO SE HA REALIZADO ESTE PROCESO'));
    else
       [fil,col]=size(filtrada1);
```

```
      visualizar(filtrada1,filtrada2,name2,name,col,canaloriginal);

   end;

end;



function visualizar(datos1,datos2,etiqueta,nombre,Muestras,canal)
global minimo maximo ver;
global Pca Spheringg Extended sub1 sub2;
global name2 name222;

%en milisegundos en vez de muestras

   [fil,col]=size(datos1);

     %ACCION
   minimo1=minimo*2000/256;
   maximo1=maximo*2000/256;
   dato11=datos1;
   compmax1=max(max(dato11));
      sprintf('%i compmax',compmax1);
      compmin1=min(min(dato11));

    intervalo1=[minimo1 maximo1 compmin1 compmax1];
   space1=compmax1-compmin1;

   %REPOSO

   dato22=datos2;
   compmax2=max(max(dato22));
      sprintf('%i compmax',compmax2);
      compmin2=min(min(dato22));

    intervalo2=[minimo1 maximo1 compmin2 compmax2];
   space2=compmax2-compmin2;


   espacio=[' '];
   Pca2=[];
   Sphering2=[];
   Extended2=[];

   if strcmp(etiqueta,'ORIGINAL') || strcmp(etiqueta,'FILTRADA')


   subg1=0;
   subg2=0;

   else
```

```
    texto='Pca';
    pca=num2str(Pca);
    Pca2=[texto espacio pca espacio];
    texto='Sphering';
    Sphering2=[texto espacio Spheringg espacio];
    texto='Extended';
    extended=num2str(Extended);
    Extended2=[texto espacio extended];
    subg1=num2str(sub1);
    subg1=[' Nº Subgauss ' subg1];
    subg2=num2str(sub2);
    subg2=[' Nº Subgauss ' subg2];

    end;


    nombre2=[Pca2 espacio Sphering2 Extended2];

    off    = [25 -25 0 0];
    pos = get(gcf,'Position');

    figure('Position',pos+2.5*off);

       subplot(1,2,1);
       eegplot('noui',dato11,128,space1,canal,0,'b');
       mas=[etiqueta ' Accion '];
       ampli=num2str(space1);
       nom = ['Amplitud máx ' ampli subg1];
       nom2= [name2 ' ' name222 ' ' nombre ' ' nombre2];
       title({mas ; nom});
       dim = [.2 .01 .50 .05];


annotation('textbox',dim,'String',nom2,'HorizontalAlignment','center')
;

       subplot(1,2,2);
       eegplot('noui',dato22,128,space2,canal,0,'r');
       mas=[etiqueta ' Reposo'];
       ampli=num2str(space2);
       nom = ['Amplitud máx ' ampli  subg2];
       title({mas; nom});


function guardar(si)
global resultados1 resultados2 name1 canal;
global ultimoaccion menus;
global borrar1 numcomp ver cargo canaloriginal;

if nargin<1
   si = gcf;
end
[filas,columnas]=size(resultados1);
```

```
mues=columnas;
canales=canaloriginal;

canales = strtok(canales,'2.m')



set(si,'Pointer','arrow');
menu = get(si,'UserData');
nombre=get(menus(10),{'value','string'});
guarda = deblank(nombre{2} (nombre{1},:));
drawnow

if strcmp(guarda,'SI')

    direct='c:\muestras\resultado';
    existe=exist(direct,'dir');
    if (existe == 0)
        errordlg(sprintf('No existe - por tanto creamos el directorio
"RESULTADO"'));
        !mkdir c:\muestras\resultado;
    end;
    fprintf('El directorio ya esta creado : directorio %s \n',direct);
    %creamos archivo acción

    if cargo==1
        nombrefic=['c:\muestras\resultado\' name1 canales 'A.ASC'];
    else nombrefic=['c:\muestras\resultado\' name1 canales 'A.ASC'];
    end;

    nombrefic=char(nombrefic);
    ultimoaccion=nombrefic;
    dlmwrite(nombrefic,resultados1(:,1:mues),'\t');
    fprintf(' ........Archivo de acción creado - componentes
independientes ........\n ');

    fprintf(' PROCESO FINALIZADO\n ');

     if strcmp(canal,'unosolo')
   %CUANDO SEA UN SOLO CANAL SOLO GRABAREMOS TRES
RESULTADOS CARACTERÍSTICOS
   fid=fopen(canal,'w+');
   [titulo]=fscanf(fid,'%s');
   canales=titulo(:,1:2);
   fclose(fid);
   size(resultados2)

   resultados1=resultados2(1:3,:);
 end;
    direct='c:\muestras\resultado';
    existe=exist(direct,'dir');
    if (existe == 0)
```

```
        errordlg(sprintf('No existe - por tanto creamos el directorio
"RESULTADO"'));
        !mkdir c:\muestras\resultado;
    end;
    fprintf('El directorio ya esta creado : directorio %s \n',direct);
    %creamos archivo acción
    if cargo==1
        nombrefic=['c:\muestras\resultado\' name1 canales 'R.ASC'];
    else nombrefic=['c:\muestras\resultado\' name1 canales 'R.ASC'];
    end;

    nombrefic=char(nombrefic);
    ultimoaccion=nombrefic;
    dlmwrite(nombrefic,resultados2(:,1:mues),'\t');
    fprintf(' .........Archivo de acción creado - componentes
independientes .........\n ');

    fprintf(' PROCESO FINALIZADO\n ');
     %close(ver);


end;

set(menus(11),'value',1);

function componentea()
global resultados resultadosa componenntesa menus;
global resultadoa ;%compp;
global ordena;

canales='compone.m';

fclose('all');

if nargin<1
   com1 = gcf;
end
set(com1,'Pointer','arrow');
menu = get(com1,'UserData');
v=get(menus(16),{'value','string'});
compo1= str2num(v{2});

compp=[];%añado para automatizar mas del 50% de las
componentes

[fil,col]=size(resultados);
 if compo1>fil
    errordlg(sprintf('NUMERO MÁXIMO DE COMPONENTES %i',fil));
    return;
 end;

if ~isempty(ordena) & isempty(compo1)
    [a,num]=size(ordena);
```

```
    ordena1=ordena(1,num-3:num);
    %ordena2=ordena(1,1:2);
    ordena=[ordena1];
    for i=1:4
       compo1=ordena(i);
       if ~isempty(compo1)
          if compo1~=0
             result=resultados(compo1,1:col);
             resultadoa=[resultadoa ; result];
             fid=fopen(canales,'w+');
             [componen]=fscanf(fid,'%s');
             componen=componen(:,(compo1-1)*4+1:(compo1*4));
             compp=[compp  componen];
             fclose(fid);
          end;
       end;
    end;

          resultadosa=resultadoa;
          resultadoa=[];
          componenntesa=resultadosa;
          fid=fopen('componentesa.m','w+');
          compp=compp';
          fprintf(fid,compp);
          fclose(fid);
          compp=[];
          size(componenntesa);


end;
feval('fin');

 function fin()
 global resultadosa resultados componenntes;
 resultados=resultadosa;
 componenntes=resultados;
 [totalfilas totalcolumnas]=size(resultados)
   fclose('all');

 function cerrar()
clear all;
close all;
lasterr('');


function auto(ind)
global auto accion1 menus e min max indic;

auto=1;
if nargin<1
    indic=1;
else indic=ind;
end;
```

```
accion1;
for i=indic:accion1
   set(menus(1),'value',i);
   feval('cargar');

   feval('electrodos');

   feval('Rango1a');

   feval('Rango2a');
   %filtro la señal en todo el rango
   feval('entrar');

   %ejecuto ICA
   feval('Activar2');

   %visualizo el resultado


   %guardo el resultado
   set(menus(11),'value',2);
   feval('guardar');
   set(menus(11),'value',1);
                              end;
```

FUNCIÓN PROCESO DE CLASIFICACIÓN – RED LVQ

```
function redn(inicio);

if nargin<1,
  inicio='inicializa';
end;

feval(inicio);
return

function inicializa()
global nomb2 menus2 simul C P;
simul=[];
C=[];
P=[];

%en el caso de que prueba ya estuviera ejecutándose traerlo a primer plano
h = findobj(allchild(0), 'tag', 'Prueba de visualización');
if ~isempty(h)
  figure(h(1))
  return
end
nomb2='  ';
d=what;
actual=d.path;

screenD = get(0, 'ScreenDepth');
```

```
if screenD>8
   grayres=256;
else
   grayres=128;
end

%figura principal
a = figure(...
   'Color',[0.8 0.8 0.8], ...
'Colormap',gray(grayres), ...
'Position',[68 120 834 533], ...
'Tag','RED NEURONAL');

menu1 = uicontrol('Parent',a, ...
        'Units','points', ...
        'FontUnits','centimeters', ...
        'BackgroundColor',[0 0 0], ...
        'FontSize',0.3, ...
        'FontWeight','bold', ...
        'ForegroundColor',[1 1 1], ...
        'Position',[100 220 80 50], ...
        'String',nomb2, ...
        'Style','listbox', ...
        'Tag','Listbox1', ...
          'Callback','redn(''cargar2'')');
textomenu1 = uicontrol('Parent',a, ...
        'Units','points', ...
        'BackgroundColor',[0 0 1], ...
        'FontAngle','italic', ...
        'FontWeight','bold', ...
        'ForegroundColor',[0 1 1], ...
        'Position',[90 280 96.2069 19.8621], ...
        'String','Seleccionar fichero a cargar', ...
        'Style','text', ...
        'Tag','StaticText1');

menu2 = uicontrol('Parent',a, ...
        'Units','points', ...
        'Callback','redn(''Iniciar'')', ...
        'Enable','off', ...
        'Position',[400 210 100 20], ...
        'String','INICIALIZACIÓN', ...
        'Tag','Pushbutton1');

menu3 = uicontrol('Parent',a, ...
        'Units','points', ...
        'Callback','redn(''Entrenar'')', ...
        'Enable','off', ...
        'Position',[400 190 100 20], ...
        'String','ENTRENAR RED', ...
        'Tag','Pushbutton1');
```

```
menu4 = uicontrol('Parent',a, ...
        'Units','points', ...
        'Callback','redn(''Simular'')', ...
        'Enable','off', ...
        'Position',[360 170 140 20], ...
        'String','SIMULACIÓN', ...
        'Tag','Pushbutton1');

menu5 = uicontrol('Parent',a, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','redn(''neuronas'')', ...
        'Enable','off', ...
        'Position',[59.5862 129.724 37.2414 13.0345], ...
        'String','2', ...
           'Style','edit', ...
        'Tag','EditText1');
textomenu5 = uicontrol('Parent',a, ...
        'Units','points', ...
        'BackgroundColor',[0.847059 0.815686 0.721569], ...
        'Position',[45.3103 116.69 80.6897 11.1724], ...
        'String','Número de neuronas', ...
        'Style','text', ...
        'Tag','StaticText1');

menu6 = uicontrol('Parent',a, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','redn(''Velocidad'')', ...
        'Enable','off', ...
        'string','1',...
        'Position',[61.4483 98.6897 37.2414 14.2759], ...
        'String','1', ...
        'Style','edit', ...
        'Tag','EditText2');
textomenu6 = uicontrol('Parent',a, ...
        'Units','points', ...
        'BackgroundColor',[0.847059 0.815686 0.721569], ...
        'Position',[28.5517 83.7931 103.655 12.4138], ...
        'String','Velocidad de aprendizaje', ...
        'Style','text', ...
        'Tag','StaticText1');

menu7 = uicontrol('Parent',a, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','redn(''Epocas'')', ...
        'Enable','off', ...
        'Position',[167.586 131.586 37.2414 12.4138], ...
        'string','255',...
        'Style','edit', ...
        'Tag','EditText2');
```

```
textomenu7 = uicontrol('Parent',a, ...
        'Units','points', ...
        'BackgroundColor',[0.847059 0.815686 0.721569], ...
        'Position',[139.655 115.448 107.379 13.0345], ...
        'String','Número máximo de épocas', ...
        'Style','text', ...
        'Tag','StaticText1');

menu8 = uicontrol('Parent',a, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','redn(''Tramo'')', ...
        'Enable','off', ...
        'Position',[169.448 99.3103 37.2414 13.0345], ...
        'String','100', ...
        'Style','edit', ...
        'Tag','EditText2');
textomenu8 = uicontrol('Parent',a, ...
        'Units','points', ...
        'BackgroundColor',[0.847059 0.815686 0.721569], ...
        'Position',[155.172 81.931 78.8276 11.1724], ...
        'String','Èpòcas a visualizar', ...
        'Style','text', ...
        'Tag','StaticText1');

menu9 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','redn(''extraer2'')', ...
        'Position',[0.1 0.9 0.4 0.03], ...
        'string',actual,...
        'Style','edit', ...
        'Tag','EditText2');
textomenu9 = uicontrol('Parent',a, ...
        'Units','normalized', ...
        'BackgroundColor',[0.847059 0.815686 0.721569], ...
        'FontSize',10, ...
        'Position',[0.1  0.87 0.4 0.03], ...
        'String','Introducir la ruta completa de los archivos para inicializar la
red', ...
        'Style','text', ...
        'Tag','StaticText3');

menu10 = uicontrol('Parent',a, ...
        'Units','points', ...
        'Callback','redn(''cerrar'')', ...
        'Enable','off', ...
        'Position',[370 10 110 20], ...
        'String','CERRAR', ...
        'Tag','Pushbutton1');

menu11 = uicontrol('Parent',a, ...
        'Units','points', ...
```

```
        'Callback','redn(''visualerror'')', ...
        'Enable','off', ...
        'Position',[360 150 140 20], ...
        'String','ERROR', ...
        'Tag','Pushbutton1');


menu12 = uicontrol('Parent',a, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'FontSize',12, ...
        'Position',[400 255 80 20], ...
        'FontWeight','bold', ...
        'Style','edit', ...
        'Tag','EditText3');
textomenu12 = uicontrol('Parent',a, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'Style','text', ...
        'Position',[400 280 72 19], ...
        'String',' TOTAL ', ...
        'Tag','StaticText2');


menu15 = uicontrol('Parent',a, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'FontSize',12, ...
        'Position',[250 255 50 20], ...
        'FontWeight','bold', ...
        'Style','edit', ...
        'Tag','ACCION');
textomenu15 = uicontrol('Parent',a, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'Style','text', ...
        'Position',[240 280 72 19], ...
        'String','Ficheros Accion', ...
        'Tag','StaticText2');

menu16 = uicontrol('Parent',a, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'FontSize',12, ...
        'Position',[325 255 50 20], ...
        'FontWeight','bold', ...
        'Style','edit', ...
        'Tag','REPOSO');
textomenu16 = uicontrol('Parent',a, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'Style','text', ...
        'Position',[310 280 85 19], ...
```

```
        'String','+   Ficheros Reposo', ...
        'Tag','StaticText2');

menu13 = uicontrol('Parent',a, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','redn(''ficherosinia1'')', ...
        'Position',[200 182 50 19], ...
        'Style','edit', ...
        'Tag','EditText4');

menu14 = uicontrol('Parent',a, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','redn(''ficherosinia2'')', ...
        'Position',[254 182 50 19], ...
        'Style','edit', ...
        'Tag','EditText4');

textomenu = uicontrol('Parent',a, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'FontSize',9, ...
        'Position',[200 210 100 25.8621], ...
        'Style','text', ...
        'String','Número de ficheros a inicializar', ...
        'Tag','StaticText4');

menu17 = uicontrol('Parent',a, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','redn(''ficherossimu1'')', ...
        'Position',[400 75 50 19], ...
        'Style','edit', ...
        'Tag','EditText4');

menu18 = uicontrol('Parent',a, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','redn(''ficherossimu2'')', ...
        'Position',[455 75 50 19], ...
        'Style','edit', ...
        'Tag','EditText4');

textomenu = uicontrol('Parent',a, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'FontSize',9, ...
        'Position',[400 105 100 19.8621], ...
        'Style','text', ...
        'String','Número de ficheros a simular', ...
        'Tag','StaticText4');
```

```
        menu19 = uicontrol('Parent',a, ...
                'Units','normalized', ...
                'FontSize',10, ...
                'Callback','redn(''extraer2'')', ...
                'Position',[0.5 0.9 0.1 0.04], ...
                'String','Seleccionar', ...
                'Tag','Pushbutton13');

        menu20 = uicontrol('Parent',a, ...
                'Units','points', ...
                'Callback','redn(''reset'')', ...
                'Enable','off', ...
                'Position',[490 10 110 20], ...
                'String','RESETEAR', ...
                'Tag','Pushbutton20');


menus2=[menu1 menu2 menu3 menu4 menu5 menu6 menu7 menu8 menu9
menu10 menu11 menu12 menu13 menu14 menu15 menu16 menu17 menu18
menu19 menu20];
        set(a,'UserData',menus2,'Visible','on');
        drawnow
        set([menu1 menu2 menu3 menu4 menu5 menu6 menu7 menu8 menu10
menu11 menu12 menu13 menu14 menu15 menu16 menu17 menu18] , 'Enable',
'off');
        set([menu20] , 'Enable', 'on');

        set([menu1 menu2 menu3 menu4 menu5 menu6 menu7 menu8 menu11
menu12 menu13 menu14 menu15 menu16 menu17 menu18] , 'Enable', 'off');
        set(a,'HandleVisibility','Callback');
        drawnow
        return

        %función que recorre el directorio donde se encuentran los ficheros a
examinar
        function extraer2(directo)
        global nomb2 menus2 archivos2;
        global nombres1 nombres2 accion reposo actual1;

        if nargin<1
          directo = gcf;
        end
           actual1 = uigetdir();


           set(menus2(9), 'String', actual1)
           set(directo,'Pointer','arrow');
           menu = get(directo,'UserData');
           v=get(menus2(9),{'value','string'});
           archivos2= deblank(v{2});
           existe=exist(archivos2,'dir');

           if (isempty(archivos2) || existe ~=7)
```

```
            errordlg(sprintf('Directorio no seleccionado o erroneo'));
            return;
        end;

    directorio2=dir(archivos2);
    [fil,col]=size(directorio2);
    for i=3:fil
        nombres(i-2)=cellstr(directorio2(i).name);
    end;



    %averiguamos los ficheros que concuerdan con fichero a cargar
    indice=strmatch('TT',nombres);

    if (isempty(indice)| indice==1)
        errordlg(sprintf('No se ha encontrado ningún archivo de carga o sólo
existe uno'));
        return;
    end;

    nombre1=nombres(indice);
    nombre2=char(nombre1);
    %los ficheros seleccionados los ordenamos alfabéticamente

    [nombres,I]=sortrows(nombre2);
    nombre=nombre2(I,:);
    [totalficheros,total]=size(nombre);

    set(menus2(12),'string',totalficheros);
    accion=0;
    reposo=0;
    j=0;

    for i=1:totalficheros
        posic=findstr(nombre(i,:),'A.');
        posic2=findstr(nombre(i,:),'p.');
        if ~isempty(posic)| ~isempty(posic2)
            j=j+1;
            if isempty(posic2)
                pos=posic;
            else pos=posic2;
            end;
            nombress1(j,1:pos)=nombre(i,1:pos);
            accion=accion+1;
        end;
        if i==totalficheros
            break;
        end;
    end;


    j=0;
```

```
        for i=1:totalficheros
           posi=findstr(nombre(i,:),'R.');
           pos=posi;
           if ~isempty(posi)
              j=j+1;
              nombress2(j,1:pos)=nombre(i,1:pos);
              reposo=reposo+1;
           end;
           if i==totalficheros
              break;
           end;
        end;

        nombress1=char(nombress1);
        nombress2=char(nombress2);
        nomb2=strvcat(nombress1,nombress2);

        set([menus2(1) menus2(2) menus2(5) menus2(6) menus2(7) menus2(8)
menus2(10) menus2(13) menus2(14) menus2(17) menus2(18)],'Enable', 'on');
        set(menus2(1),'string',nomb2);
        set(menus2(9), 'Enable', 'off');
        set(menus2(15),'string',accion);
        set(menus2(16),'string',reposo);

        drawnow


        return;

        %FUNCIÓN QUE CARGA LOS FICHEROS
        function cargar2(archivo)
        global C P nomb2 archivos2;
        global fichero1 fichero2 indice indice2;
        global simul index1 index2 menus2;


        fichero1=[];
        fichero2=[];

        if nargin<1
          archivo = gcf;
        end

        set(archivo,'Pointer','arrow');
        menu = get(archivo,'UserData');
        v=get(menus2(1),{'value','string'});
        name1 = deblank(v{2}(v{1},:));
        [fil,col]=size(nomb2);
        drawnow
        [filmax,colmax]=size(name1);

        for i=1:fil
           if strcmp(name1,nomb2(i,1:colmax))
```

```
        %fichero acción1
        nom=strcat(archivos2,'\',nomb2(i,:),'.asc');
        st=char(nom);
        load(st);
        posic=findstr(nom,'A.');
        if isempty(posic)
           posic=findstr(nom,'P.');
        end;
        nombr=nomb2(i,:);
        if ~isempty(posic)
           [fichero1]=eval(nombr);
           if isempty(fichero1)
              indice=index1+1;
              feval('ficherosinia1',indice);
           else fichero1=fichero1';
           end;
        else [fichero2]=eval(nombr);
           if isempty(fichero2)
              feval('ficherosinia2',indice2);
           else fichero2=fichero2';
           end;
        end;
        break;
     end;
end;

if ~isempty(fichero1)
   [fil,col]=size(fichero1);
   for i=1:col
      for j=1:fil
         valor=fil*(i-1);
         datos1(j+valor,1)=fichero1(j,i);
      end;
   end;
   fichero1=datos1;
end;

if ~isempty(fichero2)
   [fil,col]=size(fichero2);
   for i=1:col
      for j=1:fil
         valor=fil*(i-1);
         datos2(j+valor,1)=fichero2(j,i);
      end;
   end;
fichero2=datos2;
end;

if isempty(simul)
  feval('cogerdatos');
end;
```

```
 %Transformamos los ficheros para inicializar la red

function cogerdatos()
global fichero1 fichero2 C P;
global menus2;

ac1=[];
ac2=[];

if ~isempty(fichero1)
   [fi,co]=size(fichero1);
   ac1=ones(1,co);
end;

if ~isempty(fichero2)
   [fi,co]=size(fichero2);
   ac2=ones(1,co);
   ac2=ac2+1;
end;


Cinicial=[ac1 ac2];
if isempty(C)
   C=Cinicial;
   ac1=[];
   ac2=[];

else
   C=[C Cinicial];
   ac1=[];
   ac2=[];
end;

Pinicial=[fichero1 fichero2];

if isempty(P)
  P=Pinicial;
else
  P=[P Pinicial];
end;
[fil,col]=size(P);
fprintf('TOTAL FICHEROS CARGADOS %i \n',col);

fprintf('.......HECHO.........\n');
set(menus2(2), 'Enable', 'on');
set(menus2(11), 'Enable', 'on');
fichero1=[];
fichero2=[];

return;
```

%%%%....hasta aquí hemos cargado en el menú los archivos para la inicialización.....%%

```
%.....%%

%SELECCION DE FICHERO PARA LA INICIALILZACIÓN.....
function Iniciar()
global C P T; %LO QUE NECESITA
global W1 W2; %LO QUE DEVUELVE
global menus2;

S=feval('neuronas');%NOS DARÁ LA VARIABLE S - número de neuronas a
utilizar
T=ind2vec(C);


figure;
colormap(hsv);
colordef(gcf,'none')
plotvec(P,C);
alabel('P(1)','P(2)','VECTORES DE ENTRADA');

[W1,W2] = initlvq(P,S,C);


%hasta aquí se inicializa
fprintf('.......HECHO.....\n');


set(menus2(3) , 'Enable', 'on');
drawnow


return;
%-------------FUNCIÓN ENTRENAR RED NEURONAL-----------------------%

function Entrenar()
global W1 W2 df me lr P T; % LO QUE NECESITA
global menus2 numeroficheros error1 error2 ambiguo totalac1 totalac2;
totalac1=0;
totalac2=0;
ambiguo=0;
numeroficheros=0;
error1=0;
error2=0;

%S=eval('neuronas')
lr=eval('Velocidad');
me=eval('Epocas');
df=eval('Tramo');

figure;
hold on
clf reset;
colordef(gcf,'none')
%echo on
```

```
        %clc

        colormap(hsv)
        plot(W1(1,1),W1(1,2),'ow')
        alabel('P(1), W(1)','P(2), W(3)','Entrada/Vector de pesos')

        tp = [df me lr];

        [W1,W2] = trainlvq(W1,W2,P,T,tp);
        set(menus2(4) , 'Enable', 'on');
        drawnow

        return;

        %----------FINAL DEL ENTRENAMIENTO-----------------%%


        %-----------AHORA VAMOS A COMPROBAR QUE ES CORRECTO----------------
----%
        function Simular()
        global fichero1 fichero2 W1 W2 T;% LO QUE NECESITA
        global menus2 nomb2;
        global a b numeroficheros; %lo que devuelve
        global simul;

        set(menus2(1) , 'Enable', 'on');
        set(menus2(1),'string',nomb2);
        drawnow

        numeroficheros=numeroficheros+1;
        simul=1;

        if ~isempty(fichero1)
           p=fichero1;
           %falta capturar el fichero de prueba almacenándose en p
           a = simulvq(p,W1,W2);
           %Debería dar clase 1
        end;

        if ~isempty(fichero2)
           p=fichero2;
           b = simulvq(p,W1,W2);
        end;


        %Debería dar clase 2
        %VAMOS CALCULANDO EL ERROR
        feval('Error');
        return;

        function Error()

        global a b numeroficheros;
```

```
global error1 error2;
global totalerror totalerr err1 err2; %lo que devuelve
global col1 col2 ss1 ss2;

error=0;


%clase 1
%se supone que a es de la clase 1
if ~isempty(a)
  [fil1,col1]=size(a); %las columnas son el número de señales
    if a(1,1)==0
       error=error+1;
       error1=error1+1;
    end;
else
  col1=0;
end;


if ~isempty(b)
   [fil2,col2]=size(b);
   if b(2,1)==0
      error=error+1;
      error2=error2+1;
   end;
else
   col2=0;
end;

%calculamos el error
 totalsenal=col1+col2;

if (~isempty(totalsenal)|totalsenal~=0)
  totalerror=(100*error)/totalsenal;
  totalerror=round(totalerror);
end;

if ~isempty(ss1)
   err1=(100*error1)/ss1;
   err1=round(err1);
end;

if ~isempty(ss2)
   err2=(100*error2)/ss2;
   err2=round(err2);
end;

totalerr=(err1+err2)/2;
totalerr=round(totalerr);

return;
```

```
%visualizar el error
function visualerror()
global totalerror totalerr err1 err2 error1 error2;

errordlg(sprintf('SE HA COMETIDO UN ERROR TOTAL EN LOS ÚLTIMOS ARCHIVOS CARGADOS %i%%.....\n',totalerror));
errordlg(sprintf('SE HA COMETIDO UN ERROR TOTAL DEL %i%%\n CON ERROR EN ACCION DE %i%%\n Y ERROR DE REPOSO %i%%',totalerr,err1,err2));

%contadores a cero
error1=0;
error2=0;
return;

%SELECCIÓN DE DATOS DE INTERÉS

%SELECCIÓN NÚMERO DE NEURONAS.....
function [S]=neuronas(numero)
global S;%LO QUE DEVUELVE
global menus2;

if nargin<1
  numero= gcf;
end

set(numero,'Pointer','arrow');
menu = get(numero,'UserData');
v=get(menus2(5),{'value','string'});
S = str2num(v{2});
drawnow

%SELECCIÓN VELOCIDAD DE APRENDIZAJE.....
function [lr]=Velocidad(rate)
global lr; %LO QUE DEVUELVE
global menus2;

if nargin<1
  rate = gcf;
end
set(rate,'Pointer','arrow');
menu = get(rate,'UserData');
v=get(menus2(6),{'value','string'});
lr= str2num(v{2});
drawnow
if (isempty(lr) | (lr > 1) | (lr < 0))
   set(menus2(6), 'string', sprintf('%i', get(menus2(6), 'Value')));
   errordlg(sprintf('Debe estar entre 0 and 1'));
   return;
end

%SELECCIÓN DE  NÚMERO MÁXIMO DE ÉPOCAS PARA EL APRENDIZAJE
function [me]=Epocas(Valor)
global me; %LO QUE DEVUELVE
```

```
global menus2;

if nargin<1
   Valor = gcf;
end
set(Valor,'Pointer','arrow');
menu = get(Valor,'UserData');
v=get(menus2(7),{'value','string'});
me= str2num(v{2});
drawnow

if (isempty(me) | (me < 250))
    set(menus2(7), 'string', sprintf('%i', get(menus2(7), 'Value')));
    errordlg(sprintf('Tiene que ser un valor mayor que 250'));
    return;
 end

%SELECCIÓN DE  NÚMERO TRAMOS DE EPOCAS A VISUALIZAR
function [df]=Tramo(Valor)
global me; %LO QUE DEVUELVE df
global menus2;

if nargin<1
   Valor = gcf;
end
set(Valor,'Pointer','arrow');
menu = get(Valor,'UserData');
v=get(menus2(8),{'value','string'});
df= str2num(v{2});
drawnow

if (isempty(df) | (df > me))
    set(menus2(8), 'string', sprintf('%i', get(menus2(8), 'Value')));
    errordlg(sprintf('Tiene que ser múltiplo del número de epocas de
entrenamiento'));
    return;
end


function reset()
global menus2 simul C O numerofichero
global error1 error2;
simul=[];
error1=0;
error2=0;
  % clear simul C O numerofichero Pinicial Cinicial

      set(menus2(13),'string',0);
      set(menus2(14),'string',0);


    function cerrar()
clear all;
```

```
close all;

function ficherosinia1(ind)
global menus2 indice a1 P C;
 C=[];
 P=[];
if nargin<1
    indice=1;
else indice=ind;
end;

v=get(menus2(13),{'value','string'});
a1= str2num(v{2});
set(gcf,'Pointer','arrow');

for j=indice:a1
    set(menus2(1),'value',j);
    feval('cargar2');
end;

function ficherosinia2(ind)
global accion menus2 indice2 a2;

if nargin<1
    indice2=accion+1;
else indice2=accion+1+ind;
end;

set(gcf,'Pointer','arrow');
v=get(menus2(14),{'value','string'});
a2= str2num(v{2});
a2=a2+accion;

for j=indice2:a2
    set(menus2(1),'value',j);
    feval('cargar2');
end;

function ficherossimu1()
global menus2 a1 simul ss1 b;
clear global b;

simul=1;

v=get(menus2(17),{'value','string'});
s1= str2num(v{2});
set(gcf,'Pointer','arrow');
ss1=s1;
s1=s1+a1;
a11=a1+1;


for j=a11:s1
```

```
   set(menus2(1),'value',j);
   feval('cargar2');
   feval('Simular');
end;


function ficherossimu2()
global accion1 menus2 a a2 simul ss2;

clear global a;


simul=1;
set(gcf,'Pointer','arrow');
v=get(menus2(18),{'value','string'});
s2= str2num(v{2});
ss2=s2;
s2=s2+a2;
a22=a2+1;

for j=a22:s2
   set(menus2(1),'value',j);
   feval('cargar2');
   feval('Simular');
end;
```

X. BIBLIOGRAFÍA

- ✓ A.C.Guyton. (1972). *Structure of the Nervous System.* Philadelphia: W.B. Saunders.
- ✓ A.Papoulis. (1991). *Probability. Random Variables and Stochastic Processes.* McGraw-Hill International.
- ✓ Amari, S.-i. (1998*). Natural Gradient Works Efficiently in Learning. Neural Computation .*
- ✓ Adrian & Matthews (1934). *The interpretation of potential waves in the cortex*
- ✓ Barlow, H. (1989). *Finding minimum entropy codes. Neural Computation.*
- ✓ Barlow, H. (2000). *Redundancy reduction revisited.*
- ✓ Berger, H. (1969). *Hans Berger on the Electroencephalogram of Man: The Fourteen Original Reports on the Human Electroencephalogram.*
- ✓ Borbély, A. (1993). *El secreto del sueño: nuevos caminos y conocimientos.*
- ✓ Comon, P. (1994). *Independent component analysis - a new concept? Signal Procesing.*
- ✓ Coto, I. C. (Junio de 2013). *Proyecto NeuroTrip*. Obtenido de http://brainbci.blogspot.com.es/
- ✓ Finger, S. (1994). *Origins of Neuroscience: A History of Explorations Into Brain Function.*
- ✓ Gross, C. (2007). *The Discovery of Motor Cortex and its Background Journal of the History of the Neurosciences.*
- ✓ JEANNEROD, M. (1997). *The Cognitive Neuroscience of Action.*
- ✓ Jr, J. W. (1998). *THE ORIGIN OF BIOPOTENTIALS.*
- ✓ Lee, T.-W. (1998). *Independent Component Analysis: Theory and Applications.*
- ✓ Leigh R. Hochberg, M. D. (2006). Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature* , 164.
- ✓ Pedro A. Carrión, J. R. (2007). *Procesado de señales biomédicas.* Cuenca: Ediciones Univ. Castilla La Mancha.
- ✓ R., L. (1989). *How to generate ordered maps by maximizing the mutual information between input and output signals. Neural computation.*
- ✓ Schmidhuber, J. (1992). *Learning factorial codes by predictability minimization. Neural Computation.*

- ✓ Sejnowski, A. J. (1995). *An information-maximisation approach to blind separation and blind deconvolution.*
- ✓ *Swart Center for computational Neuroscience*. (2001). Obtenido de SCCN: http://cognitrn.psych.indiana.edu/busey/temp/eeglabtutorial4.301/eeglabtut.html
- ✓ *Swartz Center for Computational Neuroscience*. (2001). Recuperado el 2017, de https://sccn.ucsd.edu/