Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA EN TECNOLOGÍA INDUSTRIAL

TRABAJO FIN DE GRADO

APLICACIÓN MATLAB PARA EL ENTRENAMIENTO DE REDES NEURONALES PARA UN CLASIFICADOR DE ACTIVIDAD USANDO DATOS DE UNA MULETA SENSORIZADA

Alumno: Pascual, León, Sergio

Director: Zubizarreta, Pico, Asier

Curso: 2018-2019

Fecha: <10, julio, 2019>

Índice

1 Resumen

1.1 Castellano

En este trabajo de fin de grado se lleva a cabo el desarrollo de una interfaz gráfica de usuario (GUI-graphical user interface) en el entorno de programación MATLAB. Esta aplicación se utiliza para realizar el entrenamiento de redes neuronales, incluye las funciones de selección de los datos con los que se desea trabajar y el tipo de entrenamiento de la red. Finalmente muestra el resultado del entrenamiento y algunas gráficas de utilidad para la interpretación del ensayo. Esta aplicación simplifica la tarea de trabajar con redes neuronales y conjuntos de datos, permite trabajar sin necesidad de conocimientos de programación. Este proyecto forma parte de uno de mayor envergadura llevado a cabo por un grupo de investigación de la UPV/EHU : 'Contera inteligente para el diagnóstico funcional de la marcha en pacientes con Esclerosis Multiple' (SMARTIP). En este proyecto se desarrolla un prototipo de muleta sensorizada que permite la obtención de datos y valores que se utilizan para realizar las pruebas con las redes neuronales.

1.2 Euskera

Gradu amaierako lan honetan, MATLAB programazio inguruari dagokion interfaze grafikoa (GUI graphical user interface) lantzen da. Aplikazio honen bidez, sare neuronalen entrenamendua gauzatzen da, lan egiteko erabilgarriak diren datuen hautaketa eginez baita sarearen entrenamendu mota ere kontuan hartuz. Azkenik, saiakeraren interpretaziorako hainbat erabilgarritasun grafika eta entrenamenduen emaitzak aztertzen dira. Aplikazio honek, sare neuronalekin eta datu multzoekin lan egitea errazten du, programazioaren inguruko jakinduria izan gabe lan egitea baimenduz. Proiektu hau UPV / EHU ko garrantzi handiko ikertzaile talde baten eskutik landuta dago: "Esklerosi anitza duten pazienteen ibilaldiaren diagnostiko funtzionalerako txurro inteligentea (SMARTIP)". Proiektu honetan, makulu sentsorizatuaren prototipoa garatzen da, sare neuronalekin egindako probak burutzeko erabiltzen diren datu eta baloreak jasotzeko ahalbidetzen duena.

1.3 Inglés

In this final degree work, the development of a graphical user interface (GUI) is carried out with the MATLAB´s software. This application is used for neural network training, the app includes the functions of selecting the data with which you want to work and the type of training of the network. Finally it shows the result of the training and some graphs of utility for the interpretation of the process. This application simplifies the task of working with neural networks and data sets, allows working without programming knowledge. This project is part of a larger one carried out by a research group of the UPV / EHU: 'Smart tip for the functional diagnosis of walking in patients with Multiple Sclerosis' (SMARTIP). In this project, a prototype

of a sensorized crutch is developed that allows obtaining data and values that are used to perform the tests with neural networks.

2 Lista de tablas e ilustraciones

2.1 Lista de tablas

2.2 Lista de ilustraciones

3 Introducción

La Esclerosis Múltiple es una enfermedad degenerativa del sistema nervioso, para conseguir un buen tratamiento es necesario un correcto diagnóstico. Las formas tradicionales de diagnosticar esta enfermedad no son exactas, por lo que en la actualidad se está estudiando la implementación de medios técnicos para ayudar a evaluar a los pacientes.

Dentro de estos medios técnicos se encuentra el proyecto "Contera inteligente para el diagnóstico funcional de la marcha en pacientes con Esclerosis Múltiple" (SMARTIP), cuyo objetivo principal es el desarrollo de un dispositivo que recoja información del estado físico del paciente mediante sensores instalados en una muleta. Gracias a la información obtenida y a un posterior tratamiento de los datos y análisis de resultados se obtienen características sobre el estado del paciente para posteriormente plantear un diagnóstico.

Para realizar el tratamiento de estos datos se utiliza inteligencia artificial, redes neuronales artificiales [RNA/ANN "Artificial Neural Network"] en este caso. Los entrenamientos de estas redes se pueden realizar desde sistemas de cómputo numéricos diferentes, pero todos coinciden en que hay que conocer el lenguaje de programación y la forma de ejecutarlo.

En este trabajo se desarrolla una interfaz gráfica de usuario [GUI "Graphical User Interface"] en MATLAB que permita realizar esta tarea, reduciendo la obtención de conjuntos de datos para el entrenamiento y el uso de la red a una interacción directa con la aplicación.

Se busca un diseño intuitivo y simple de la aplicación con el objetivo de que pueda ser utilizada sin que el usuario necesite conocimientos sobre el lenguaje de programación y el software utilizado.

El objetivo final es ayudar con la tarea de procesamiento de datos obtenidos de pacientes reales y ayudar así al fisioterapeuta a valorar con precisión a los pacientes de Esclerosis Múltiple. Un buen diagnóstico y tratamiento puede mejorar la calidad de vida de los pacientes y del entorno que les rodea.

4 Contexto

4.1 La Esclerosis Múltiple

DEFINICIÓN

La EM (Esclerosis Múltiple) es una enfermedad del sistema nervioso central, concretamente ataca a las zonas del cerebro y de la médula espinal. Es una enfermedad que se desarrolla a edades tempranas, crónica, a largo plazo y de carácter autoinmune, es decir, viene provocada porque el propio sistema inmunitario ataca por error a las células sanas del cuerpo. Actualmente es una enfermedad sin cura y con origen desconocido, aunque algunos estudios formulan que puede deberse a factores genéticos, siendo más frecuente en personas con predisposición genética, y ambientales. El único tratamiento posible es la utilización de fármacos y terapia para que el avance de la afección sea más lento.

La mielina es una sustancia grasa que actúa de capa aislante alrededor de los nervios. Esta componente permite que viajen los impulsos eléctricos entre las neuronas asegurando la correcta transmisión de la información. Consigue que las personas puedan realizar sus acciones de manera coordinada y sin esfuerzo de consciencia.

La EM se produce cuando el sistema inmunológico ataca a esta sustancia, produciendo el fenómeno de desmielinización. La desmielinización tiene diversos síntomas, principalmente son de tipo motor, sensitivo y visceral. El deterioro de la mielina es gradual, desde una primera inflamación hasta pérdida de conexión total entre vaina y axón, pudiendo llegar a dañar el nervio. La inutilización de la mielina deja cicatrices en los tejidos sanos afectados, los cuales dan lugar a que los impulsos se transmitan de manera ineficiente y se acabe desarrollando la esclerosis.

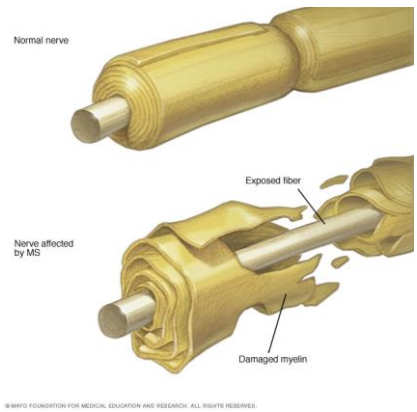


Ilustración 1. Estado normal de la vaina mielina(arriba). Estado dañado de la vaina (abajo).

COSTE E IMPACTO EM

En el mundo 2.500.000 de personas tienen EM, en Europa 700.00 y en España cada año crece el número de casos, en 2018 había 47.000 casos detectados (70% mujeres) y cada año se diagnostican 1.800 nuevos. En los últimos 20 años los casos diagnosticados se han multiplicado por 2,5. En el último informe de 2019 los casos detectados en España van ya por 55.000, con un 75 % de mujeres entre los afectados.

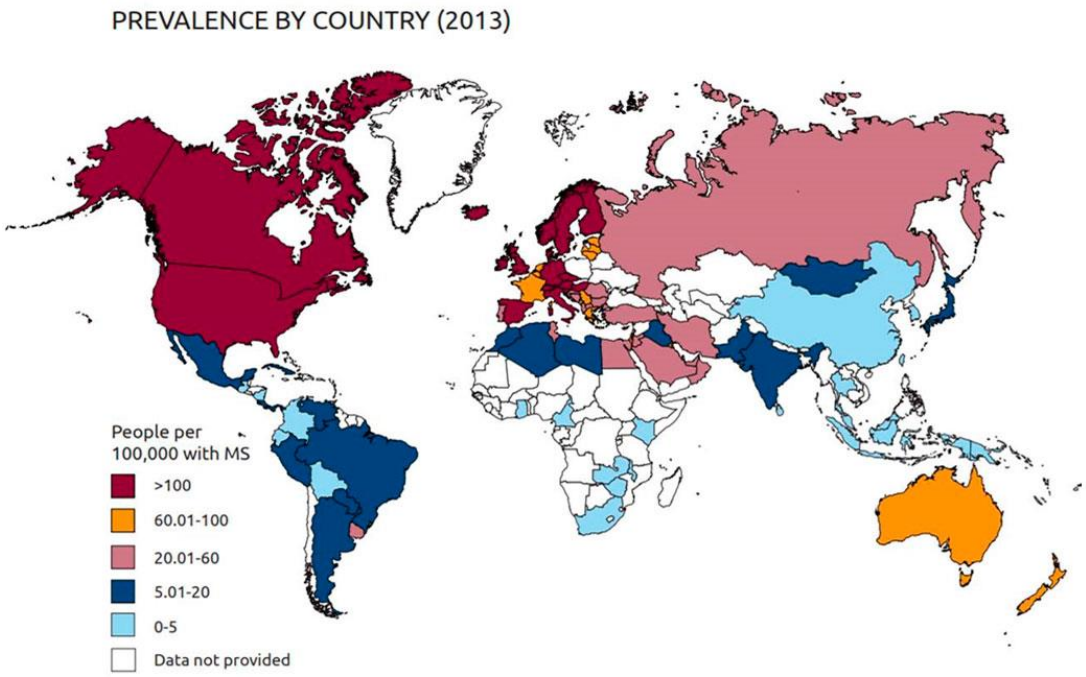


Ilustración 2. Prevalencia de la EM en el mundo (MSIF, 2018, p.8)

Las enfermedades de carácter neurodegenerativas suponen un gasto muy elevado para la sanidad pública. Se distinguen costes directos, aquellos relacionados con necesidades propias de la enfermedad (medicamentos, hospitalización, tratamiento, ...), y costes indirectos, procedentes de las limitaciones en la vida del paciente a causa de la enfermedad (jubilaciones anticipadas, mortalidad, …). Se deben tener en cuenta los costes intangibles, que son aquellos que sufre el propio paciente en su calidad de vida.

En Europa el coste supone aproximadamente 9.000 millones de euros al año, según la SEN (Sociedad Española de Neurología) el 80 % de los gastos que genera la EM guardan relación con la discapacidad y no con la terapia.

SÍNTOMAS-COMPLICACIONES

Aunque su origen es desconocido existen algunos indicadores sobre el desarrollo de la enfermedad. Aunque puede aparecer a cualquier edad, es más común en adultos-jóvenes, entre los 20 y los 40 años. Es más frecuente que se de en mujeres que en hombres, en una proporción cercana a dos mujeres afectadas por cada hombre. Según algunos estudios la enfermedad podría afectar en mayor medida a las personas que viven en climas templados que a los tropicales. Hay una mayor tasa de afectados en las regiones del norte de Europa y América del Norte. Existe también una componente genética que adiverte de la posibilidad de desarrollar EM si un familiar la padece se encuentra entere el 1 y el 10%.

Los síntomas de una persona que sufre EM son muy diferentes según el paciente y el avance de la enfermedad, que se aprecia en el daño que han recibido las fibras nerviosas. En mayor medida se dan manifestaciones de tipo sensorial y muscular. Algunos de los más frecuentes son:

- Cansancio, vértigo, falta de equilibrio o debilidad en los miembros acompañada de pérdida de fuerza.
- Problemas de habla, visión y control urinario.
- Dificultad para coordinar movimientos.
- Pérdida de memoria, concentración y otros problemas emocionales.

TIPOS-EVOLUCIÓN DE LA ENFERMEDAD

La enfermedad se puede desarrollar en cuatro fases diferentes, pudiendo evolucionar de una a sus posteriores.

- La primera fase se denomina remitente-recurrente (EMRR), es la forma más común, afectando al 80% de los pacientes. Se puede manifestar en forma de brote, donde los síntomas aparecen y desaparecen con el tiempo. En torno el 60-70% de estos casos acaban convirtiéndose en EMPS.
- La segunda fase es progresiva primaria (EMPP), esta fase afecta al 10% de los dolientes. Se caracteriza por un avance progresivo de la enfermedad sin mejorías en el tiempo.
- La tercera fase recibe el nombre de progresiva secundaria (EMPS), es el desarrollo evolutivo de la EMRR. Durante el desarrollo de las fases progresivas puede haber intervalos de estabilidad.
- La última fase se denomina progresiva recurrente (EMPR), se desarrolla con o sin brotes de manera progresiva desde el comienzo.

Algunos estudios recientes mencionan otra fase de la enfermedad, conocida como esclerosis benigna. Su característica principal es que no se pierden capacidades móviles en las personas que la padecen. Afecta solo a un 5-10% de los pacientes.

DIAGNÓSTICO

El diagnóstico de la enfermedad es llevado a cabo por un neurólogo, como no existen pruebas específicas para la EM el médico se basa en el historial clínico y la exploración física del paciente para ir descartando otras enfermedades.

Algunos métodos que se utlizan para la determinación del estado son:

- Punción lumbar en la parte baja de la espalda, se extrae una porción de líquido cefalorraquídeo (LCR), su examinación permite determinar el estado del cerebro y médula espinal.
- Análisis de sangre en busca de biomarcadores específicos que puedan ayudar a diagnosticar EM.
- Resonancia magnética, permite ver el estado de las lesiones de cerebro y médula espinal.
- Pruebas de potenciales provocados, basadas en un control de las señales eléctricas y la velocidad de conducción de estas a través de los nervios. Permite evaluar el grado de deterioro de la mielina.

Los casos más complicados de diagnosticar son aquellos en los que la enfermedad se encuentra en fases progresivas, por el contrario las fases remitente-recurrente son las más fáciles de detectar.

TRATAMIENTO

La tratamientos para la enfermedad varían según esta se manifieste en momentos puntuales en forma de ataques temporales o si el objetivo del tratamiento es frenar o modificar el avance de la EM.

Cuando la enfermedad se manifiesta como ataques puntuales en forma de brotes, con dolores puntuales, se recurre al uso de corticoesteroides para reducir la inflamación de los nervios o al intercambio de plasma. Este segundo método consiste en la extracción de plasma sanguíneo, mezcla posterior con una proteína llamada albúmina e introducción de vuelta al sistema sanguíneo.

La mayoría de las terapias modificadores de la enfermedad utilizan fármacos muy potentes con muchos efectos secundarios, son terapias agresivas y con resultados como insomnio, infertilidad, naúseas, irritaciones, fiebre, otros problemas de salud y alto coste económico.

Según la fase de la enfermedad se aplican unas terapias u otras. Por ejemplo para la EMPP la Food And Drug Administation (FDA) reconoce un único tratamiento, en el que se utiliza un fármaco denominado Ocrevus.

Para la EMRR existen muchos tratamientos, se enumeran a continuación algunos de ellos:

- Interferones beta. Es un medicamento de inyección subcutánea para reducir frecuencia y gravedad de las recaídas. Es el más utilizado, pero por el contra puede causar un daño hepático muy grande.

- Ocrevus (ocrelizumab). Es el tratamiento mencionado anteriormente para la fase de EMPP, es un fármaco de inyección intravenosa que consigue reducir las recaídas y la pérdida de movilidad que causa la enfermedad.
- Otros fármacos menos utilizados son: Copaxone, Tecfidera, Gilenya, Aubagio, Tysabril, Lemtrada, Mitoxantrona y relajantes musculares.

PRINCIPIO ACTIVO	NOMBRE COMERCIAL	INDICACIÓN	LABORATORIO TITULAR	AÑO DE APROBACIÓN (EMA)	MODO DE ADMINISTRACIÓN	FRECUENCIA DE ADMINISTRACIÓN
Acetato de glatirámero	Copaxone® 20 ó 40 mg.	EMR	Teva Pharmaceuticals Ltd	2004 y 2015	Subcutáneo	Cada día o 3 veces por semana
Acetato de glatirámero	Glatiramero Mylan 20 ó 40 mg. (genérico de Copaxone®)	EMR	Mylan	2016 y 2017	Subcutáneo	Cada día o 3 veces por semana
Alemtuzumab	Lemtrada®	EMRR	Genzyme Therapeutics Ltd	2013	Intravenoso	Ciclos de 5 ó 3 días anuales
Cladribina	Mavenclad®	EMR	Merck	2017	Oral	Dos cursos de tratamiento a lo largo de dos años
Dimetilfumarato	Tecfidera®	EMRR	Biogen	2014	Oral	Dos veces/día
Fingolimod	Gilenya®	EMRR	Novartis Europharm Ltd	2011	Oral	Cada día
Interferón beta-1a	Avonex®	EMR	Biogen	1997	Intramuscular	Una vez/semana
Interferón beta-1a	Rebif®	EMR	Merck	1998	Subcutáneo	Tres veces/semana
Interferón beta-1b	Betaferon®	EMR	Bayer Pharma Ag	1995	Subcutáneo	Cada dos días
Interferón beta-1b	Extavia®	EMRR/SP	Novartis Europharm Ltd	2008	Subcutáneo	Cada dos días
Interferón beta-1a pegilado	Plegridy®	EMRR	Biogen	2014	Subcutáneo	Cada 2 semanas
Mitoxantrona (en genérico desde 2006)	Novantrone®	EMR	Meda Pharma, S.A.U.	1998 (proc. Nacional)	Intravenoso	Frecuencia variable. Dosis máxima acumulada: 140 mg/m2
Natalizumab	Tysabri®	EMRR	Biogen	2006	Intravenoso	Cada 4 semanas
Ocrelizumab	Ocrevus®	EMR/EMPP	Roche Farma	2018	Intravenoso	Cada 6 meses
Teriflunomida	Aubagio ®	EMRR	Sanofi-Aventis Groupe	2013	Oral	Una vez/día

Nota: existen otros fármacos que se usan fuera de indicación ocasionalmente

Esclerosis Múltiple España www.esclerosismultiple.com
Para colaborar en la investigación de la EM, infórmate en www.eme1.es

Información correspondiente a enero de 2019

Tabla 1.Tratamientos modificadores de la Esclerosis Múltiple aprobados por la European Medicines Agency (EMA). Enero 2019.

Existen otros tratamientos sin fármacos, la fisioterapia es el que más se utiliza. Los objetivos fundamentales que persigue un fisioterapeuta cuando trata a un paciente con EM son: mejorar la estabilidad y los patrones de movimiento, movilidad articular, mejorar la fatiga y prevenir la aparición de problemas por la dificultad de movimiento del paciente. La finalidad es mejorar la calidad de vida. El tratamiento se realiza a través de ejercicios de estiramiento, fortalecimiento y masajes.

Con objetivo de adaptar el tratamiento de la mejor manera posible es necesario conocer no solo el estado de salud en el momento de la evaluación, si no a lo largo del tiempo. Para recoger información sobre el estado del paciente durante el día a día surge el proyecto SMARTIP, que

se encarga de desarrollar un dispositivo que permite captar esta información y ayudar a la movilidad de las personas.

4.2 SMARTIP

SMARTIP o "Contera inteligente para el diagnóstico funcional de la marcha en pacientes con Esclerosis Múltiple" es un proyecto de investigación llevado a cabo por la UPV/EHU.

El objetivo principal es conseguir ayudar a la obtención de un diagnóstico correcto de los pacientes con EM gracias a un dispositivo técnico acoplado a la contera de una muleta.

Los tres propósitos más importantes son:

1. Diseño de una contera inteligente adaptable a un dispositivo de ayuda para la marcha, el cual permita monitorizar y hacer un seguimiento de la actividad física del paciente.
2. Desarrollo de un sistema de tratamiento de datos inteligente, el cual permita obtener datos objetivos sobre la evaluación del paciente.
3. Validación clínica del sistema propuesto con pacientes reales de EM.

Para desarrollar el proyecto ha sido necesaria una gran base teórica sobre la EM y las escalas de valoración de la enfermedad.

Los estudios más novedosos argumentan que el diagnóstico es mucho más exacto si se añade a la experiencia del fisioterapeuta una información sobre la actividad a lo largo del día del paciente.A tal fin, es necesario recoger datos a lo largo del tiempo para que la información disponible sea lo más real y describa la situación de manera óptima. En enfermedades de este tipo la movilidad se puede reducir a lo largo del día debido a la fatiga acumulada y a los dolores de articulaciones o musculares.

La mayoría de pacientes con EM necesitarán un apoyo para moverse, como puede ser bastón, muletas o andadores. En este proyecto se propone sensorizar este apoyo técnico para obtener la información necesaria de la actividad realizada a lo largo de un período de tiempo.

A tal fin se contempla el diseño de una contera inteligente que incorpora un sensor de fuerza, un inclinómetro y una IMU (acelerómetro más giroscopio de tres ejes).

La ventaja de colocar estos sensores en una muleta rígida es que las medidas tendrán una referencia casi constante. Si se colocan en cualquier parte del cuerpo las vibraciones y movimientos naturales pueden hacer que se pierda precisión en los datos recogidos. Con la implantación de estos sensores se consigue ahorrar costes y un peso reducido, por lo que no causarán más fatiga ni dolencias por la sobrecarga en el portador de la muleta, además de que el tiempo de puesta en marcha se reduce a unos pocos segundos, ya que no es necesario arrancar ni ajustar ningún dispositivo.

Ilustración 3. Piezas que componen el sistema de sujeción de sensores.

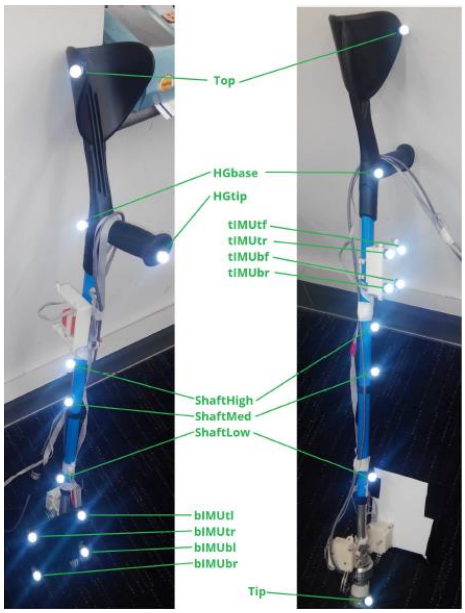


Ilustración 4. Muleta con marcadores para recoger el movimiento con sistemas de captura 3D.

A continuación se resumen los datos técnicos de la muleta sensorizada:

COMPONENTES DE LA MULETA	
VARIABLES	SENSORES
Fuerza	Sensor de fuerza C9C (HBM) + Amplificador bajo consumo INA118 (Texas Instruments)
Posición angular	Inclinómetro de dos ejes SCA100T-D02 (Murata)
Aceleraciones y velocidades angulares	MPU IMU Click (MikroElektronika) + integrado MPU-6000 (InvenSense) = acelerómetro triaxial, giroscopio triaxial y unidad de procesamiento de movimiento (MPU)
Medidas de los sensores	myRIO (National Instruments). Alimentación: bateria recargable NiMH (Ansmann) 3000mAh 7,2V

Tabla 2. Componentes de la muleta.

Los datos obtenidos en bruto gracias a los sensores no son de utilidad, ya que la información que aportan no es apta para trabajar. Es necesario un procesamiento posterior en busca de obtener indicadores que permitan clasificar el movimiento. Para realizar esta tarea se utiliza inteligencia artificial, en este caso redes neuronales artificiales.

4.3 Redes neuronales artificiales

Las redes neuronales artificiales (RNA/ANN "Artificial Neural Net") son modelos con un comportamiento similar al de el cerebro humano. Son capaces de predecir sucesos, organizar estructuras , reconocer patrones, aprender a partir de datos y otras muchas funciones. Estos métodos computacionales cobran gran importancia y están en constante evolución, cada vez se utilizan en más sectores y aplicaciones.

Estas características que reúnen las hacen aspirantes perfectas para procesar los datos que se obtienen de los sensores de la muleta y conseguir clasificar los diferentes patrones de movimiento. En este trabajo se han utilizado cuatro patrones diferentes: andar recto, subir escaleras, bajar escaleras y estático. A partir de estas clasificaciones y procesamiento de datos se puede mejorar el diagnóstico clínico que se realiza con los pacientes.

Los elementos básicos que componen un a red neuronal son tres: entradas, que contienen la información del exterior, pesos y salidas. Los pesos son coeficientes que ponderan la importancia de cada dato de entrada a la red, cada neurona trabaja con muchos datos y gracias a la variación de trascendencia de las entradas se obtienen salidas más exactas.

- Función de ponderación o propagación: transforma las entradas en el potencial de la neurona, normalmente se utiliza la suma ponderada de las entradas multiplicada por los pesos, aunque existen otras reglas como la distancia euclídea o la distancia de Manhattan.
- Función de activación: combina el potencial de la neurona con el estado actual para conseguir el estado futuro de activación de la neurona. Normalmente es una función creciente monótona como pueden ser las lineas, escalón, hiperbólicas o tangenciales.

- Función de salida: proporciona la salida de una neurona para que pueda ser transmitida a la siguiente.

Las redes constan de múltiples capas, una de entrada, otra de salida y número variable de capas ocultas intermedias, todas ellas trabajan en paralelo unidas mediante sinaposis, igual que las redes biológicas.

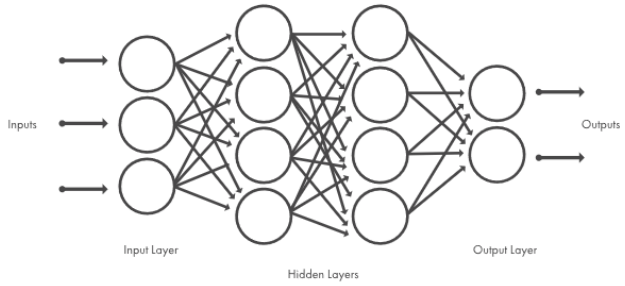


Ilustración 5. Arquitectura típica de una red neuronal.

Las neuronas se pueden conectar de diversas maneras: unión con todos, cada neurona de una capa con todas las de la capa siguiente; unión lineal, cada neurona de una cada únicamente con otra de la siguiente capa; predeterminado, consiste en una conexión previamente fijada, se utiliza en redes con capacidad de variar el número de neuronas y las conexiones entre capas.

La red debe ser capaz de asignar una salida a cada conjunto de datos de entrada, para ello hay que realizar un proceso de entrenamiento o aprendizaje. El aprendizaje es un proceso mediante el cual se espera que la propia red neuronal sea capaz de ponderar cada dato para la obtención de la salida correcta, esta ponderación se traduce en variación de parámetros de las neuronas o modificaciones entre las conexiones. Existen dos métodos de aprendizaje: supervisado y no supervisado. Otra clasificación posible se basa en si la red puede aprender durante el entrenamiento (on line) o para aprender hay que desconectarla cuando termina el entrenamiento (off line).

El aprendizaje supervisado se basa en que el usuario o manipulador de la red modifica los parámetros de entrenamiento para que la salida de la red sea una previamente calculada o deseada. El supervisor modifica las conexiones y sus pesos en base a criterio propio. Se puede realizar mediante corrección del error, por refuerzo o por aprendizaje estocástico.

El otro modelo, aprendizaje no supervisado, se basa en una automatización completa de la red. No se conoce la salida correcta ni la deseada, es la propia inteligencia artificial la que se encarga de variar los parámetros e interpretar los resultados obtenidos para ir mejorando las salidas en base a los entrenamientos y experiencia previa. Aprendizaje hebbiano y competitivo-comparativo son los modos más utilizados en esta categoría.

Posterior al proceso de entrenamiento es necesario hacer una validación de la red. Para esto se utiliza un conjunto de datos diferente al de entrenamiento, conjunto de validación o test, en este proceso se introduce a la red un conjunto de datos de entrada y se contrasta la salida de la red

con la solución que se tiene del conjunto de datos. Se verifica si la red funciona correctamente y se puede utilizar para resolver problemas posteriores.

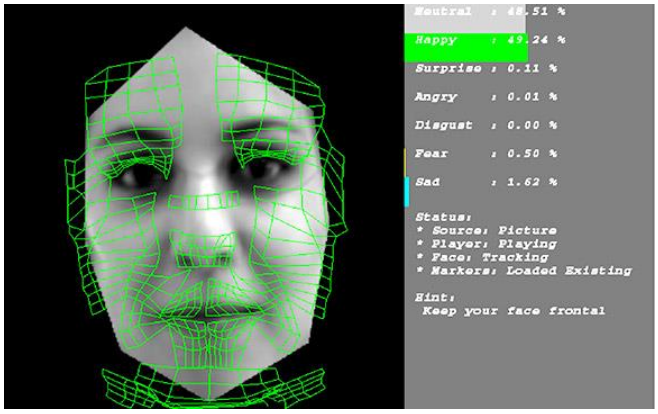


Ilustración 6. Aplicación de la inteligencia artificial en reconocimiento facial.

En conclusión, la EM es una enfermedad muy perjudicial, causa mucho dolor y actualmente carece de cura. Para mejorar la calidad de vida de los pacientes se necesita un buen tratamiento, imposible de obtener sin un buen diagnóstico. Para esto necesitamos monitorizar a los pacientes en el día a día, obtener información de su fatiga y movilidad en el tiempo. Con este fin se utiliza la muleta sensorizada, a partir de estos sensores se tratan los datos en redes neuronales y se clasifican los patrones de movimiento para investigar.

Existen muchas configuraciones de red neuronal, con el objetivo de encontrar la mejor en este TFG se abordará el diseño de una aplicación en la que se pueden variar los parámetros de red para encontrar la estructura de red óptima.

4.4 Base de Datos Experimental: Datos de partida.

Para llevar a cabo este Trabajo de Fin de Grado se parte de una base de datos que anteriormente han sido obtenidos gracias a pruebas realizadas con la muleta. Estas pruebas se han realizado en los laboratorios de análisis de movimiento de Fisioterapia de la UPV/EHU y en el Servicio de Rehabilitación de ADEMBI. Todos los ensayos cuentan con la autorización del CEISH (Comité de Ética para las Investigaciones relacionadas con Seres Humanos). El comité está formado por un grupo multidisciplinar encargado de evaluar y seguir todos los planes que conlleven intervenciones con seres humanos, uso de muestras biológicas y datos personales.

Estos datos han sido filtrados y procesados previamente con el siguiente proceso:

- Segmentación en pasos: Se han identificado las ventanas correspondientes a cada paso realizado con la muleta.
- Extracción de indicadores: A partir de cada ventana, se han extraido indicadores estadísticos:

- Media.
- Desviación estándar.
- Varianza.
- Kurtois.
- Coeficiente de correlación.
- Percentiles (25, 50 y 75).
- Área debajo de las curvas.
- Rango intercuartil.

Todas las variables se han obtenidos en los tres ejes (X, Y, Z) para el acelerómetro y giroscopio.

- Identificación de la actividad: Tras el procesado, estos datos se han clasificado por actividades: Subir Escaleras, Bajar Escaleras, Estar quieto y Andar.

Los indicadores extraídos se han almacenado en la estructura de datos de MATLAB llamada Datos_Estadisticos. En la estructura de datos aparecen también los valores de datos estadísticos del barómetro y la presión sin ejes porque en estos casos carece de sentido la orientación.

A continuación se detalla la disposición de los indicadores estadísticos dentro de la estructura. Los datos vienen también clasificados en el tipo de movimiento al que pertenecen: andar recto, subir escaleras y bajar escaleras. También existe una clasificación en estático.

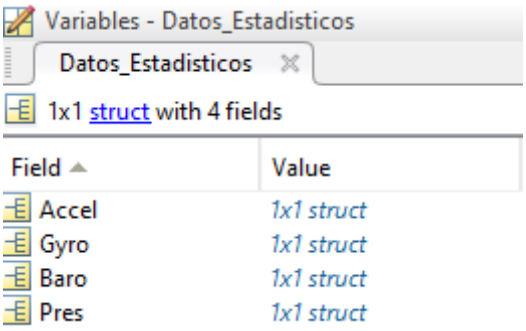


Tabla 3. Primer nivel del archivo de datos "Datos_Estadisticos".

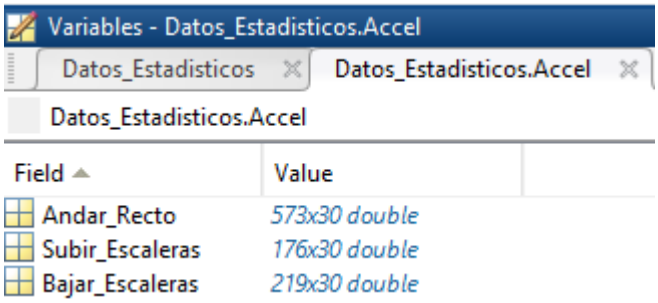
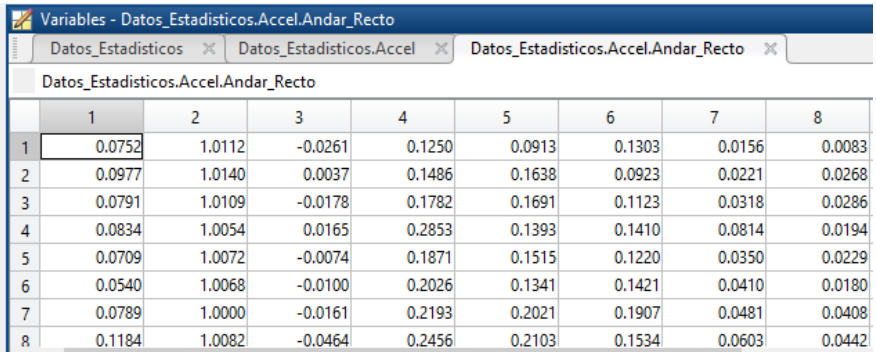


Tabla 4. Segundo nivel del archivo de datos "Datos_Estadisticos", dentro del sensor acelerómetro.



	1	2	3	4	5	6	7	8
1	0.0752	1.0112	-0.0261	0.1250	0.0913	0.1303	0.0156	0.0083
2	0.0977	1.0140	0.0037	0.1486	0.1638	0.0923	0.0221	0.0268
3	0.0791	1.0109	-0.0178	0.1782	0.1691	0.1123	0.0318	0.0286
4	0.0834	1.0054	0.0165	0.2853	0.1393	0.1410	0.0814	0.0194
5	0.0709	1.0072	-0.0074	0.1871	0.1515	0.1220	0.0350	0.0229
6	0.0540	1.0068	-0.0100	0.2026	0.1341	0.1421	0.0410	0.0180
7	0.0789	1.0000	-0.0161	0.2193	0.2021	0.1907	0.0481	0.0408
8	0.1184	1.0082	-0.0464	0.2456	0.2103	0.1534	0.0603	0.0442

Tabla 5. Datos contenidos en "Datos_Estadisticos.Accel.Andar_Recto".

ACELERÓMETRO Y GIRÓSCOPO	
COLUMNAS (Ejes X-Y-Z)	INDICADOR
1-2-3	Media
4-5-6	Desviación estándar
7-8-9	Varianza
10-11-12	Kurtois
13-14-15	Coeficiente de correlación
16-17-18	Percentiles 25
19-20-21	Percentiles 50
22-23-24	Percentiles 75
25-26-27	Área debajo de las curvas
28-29-30	Rango intercuartil

Tabla 6. Distribución de salidas en Acelerómetro y Giróscopo.

BARÓMETRO	
COLUMNAS (Valor puro-filtrado)	INDICADOR (Ejes X-Y-Z)
1-2	Media
3-4	Desviación estándar
5-6	Varianza
7-8	Kurtosis
9-10	Percentiles 25
11-12	Percentiles 50
13-14	Percentiles 75
15	Pendiente
16-17	Área debajo de las curvas
18-19	Rango intercuartil

Tabla 7. Distribución de salidas en Barómetro.

PRESIÓN	
COLUMNAS	INDICADOR
1	Media
2	Desviación estándar
3	Varianza
4	Kurtosis
5	Percentiles 25
6	Percentiles 50
7	Percentiles 75
8	Pendiente
9	Área debajo de las curvas

Tabla 8. Distribución de salidas de Presión.

4.5 Problemática

A partir de los datos anteriores se desea realizar un estudio para verificar qué red neuronal (estructura, configuración, tipo) es la adecuada para realizar tareas de clasificación. A tal fin, se requiere realizar un conjunto elevado de pruebas que conviene automatizar de forma sencilla.

A tal fin se plantea crear un programa en Matlab que automatice el proceso y ayude en su resolución. Así se reducirá a la interacción con la interfaz. Desde una única ventana se conseguirá realizar todo el proceso por orden sin necesidad de variar el código ni acceder a este.

La utilidad de la GUI resulta en la simplificación que se consigue cuando se realizan muchas pruebas en la selección de datos y variación de parámetros. Consiguiendo reducir los tiempos entre entrenamientos.

5 Objetivos

El objetivo prinicipal de este trabajo es crear una aplicación en MATLAB para permitir el entrenamiento de una red neuronal de una manera simple e intuitiva, reduciendo las labores de programación del entrenamiento y asignación de variables a una interacción con la aplicación. Esta red neuronal tendrá como objetivo ser un clasificador de actividades basado en los datos de la muleta asociada al proyecto SMARTIP.

A tal fin se desarrollará una GUI para simplificar la tarea de entrenamiento. Para llevar a cabo esta labor de manera simplificada se opta por utilizar el software de MATLAB para la creación de una GUI (grapich user interface o interfaz gráfica de usuario).

Con el fin de desarrollar este objetivo se plantean los siguientes subobjetivos:

- Estudio de las estructuras de datos de partida y comprensión de la información recibida tanto del proyecto como de la programación de redes neuronales.
- Diseño del aspecto de la interfaz.
- Definición de las variables y funciones necesarias para el correcto funcionamiento.
- Programación de las funciones mencionadas anteriormente.
- Validación y puesta en marcha.

Este proyecto está dentro de otro de mayor envergadura denominado SMARTIP, el cual ya se ha explicado en apartados anteriores. Nótese que el desarrollo de la muleta y la obtención de la base de datos queda fuera del alcance de este TFG, así como el determinar la mejor red para clasificar. El alcance de este proyecto queda ligado al diseño, desarrollo e implementación de las aplicaciones Matlab para facilitar el entrenamiento y la selección de la red.

6 Beneficios

BENEFICIOS APLICACIÓN

La manipulación de una red neuronal y la obtención de conjuntos de datos para realizar el entrenamiento a partir de la base de datos original puede ser muy complicada si el usuario no está familiarizado con el software de MATLAB. Mediante una interfaz gráfica de usuario se simplifica esta tarea de entrenamiento a la interacción con una aplicación, lo cual es más intuitivo qué acceder a las líneas de código programadas.

BENEFICIOS SMARTIP

El objetivo global de este proyecto es realizar un correcto diagnóstico en los pacientes que sufren esclerosis múltiple. Con un diagnóstico correcto se puede enfocar de manera óptima el tratamiento de rehabilitación gracias a esto se conseguiría mejorar la calidad de vida de los pacientes y un ahorro en sanidad en las arcas públicas.

La inclusión de métodos técnicos o inteligentes para la evaluación del estado del paciente permite una mejor valoración y monitorización del estado del paciente.

RESUMEN GLOBAL

El beneficio global de este proyecto es la obtención de un diagnóstico correcto del paciente. Como consecuencias directas de lo anterior derivan: una mejor forma de trabajo del terapeuta encargado de la rehabilitación, un ahorro en el coste total que conllevan estos pacientes a la sanidad pública ya que es una enfermedad neurodegenerativa sin cura y con un mejor tratamiento gracias a una mejor información del estado del paciente el tratamiento es más preciso.

7 Análisis de alternativas

Algunas de las herramientas más utilizadas para la creación de GUIs son Visual Basic (Microsoft), Python, LabView y MATLAB. Se explican a continuación algunos aspectos importantes de cada software para posteriormente tomar la decisión del que se utiliza en este trabajo.

Ilustración 7. Imagotipo VISUAL BASIC.

Visual Basic (VB) está diseñado para crear de manera productiva orientadas a objetos. Los programas escritos en Visual Basic se benefician de la seguridad y la interoperabilidad entre lenguajes. Con la última generación de VB se crean aplicaciones basadas en el entorno de trabajo .NET de Microsoft de manera rápida y sencilla.

La interfaz de programación gráfica de VB es Windows Forms , la componen un conjunto de bibliotecas que simplifican tareas como leer y escribir en un sistema de archivos, programar respuesta del usuario, crear cuadros de diálogo, etc. Para todas estas labores posee un paquete de herramientas con los siguientes controles: Pointer, PictureBox, TextBox, Frame, CommandButton, Control, CheckBox, OptionButton, ComboBox, ListBox. Horizontal and Vertical scroll bars, timer control, DriveListBox, DirListBox, FileListBox, Shape Control, Line Control, Data Control, Object Linkking and Embedding Control.

Para la manipulación y muestra de datos VB posee una herramienta de control llamada DataGridView, útil para la organización de información de forma simplificada. Esta herramienta resula muy cómoda a la hora de personalizar las tablas de datos y realizar operaciones complejas con ellos.

Una vez terminado el desarrollo de la GUI, ClickOnce es la herramienta encargada de administrar los elementos y objetos necesarios para que la instalación se lleve a cabo de manera óptima en el equipo donde se va a trabajar.

Ilustración 8. Imagotipo PYTHON.

Python es uno de los lenguajes de programación de código abierto más importantes. Administrado por la compañía Python Software Foundation, su misión principal es conseguir un código accesible y sencillo.

Las principales ventajas que presenta son la simplificación y sencillez en el código, herramientas que le hacen ser muy flexible en la depuración de errores y su portabilidad, tanto en Windows, Linux o Mac. Es importante mencionar también la gran comunidad de usuarios que utilizan Python, cuidan el lenguaje y aportan entre usuarios.

Las desventajas que tiene son que la curva de aprendizaje del lenguaje es complicada y que la mayoría de los servidores no tienen soporte a Python. Algunos usuarios utilizan librerías externas porque las que tiene por defecto no gustan a todo el mundo.

Python posee muchas herramientas para la creación de GUIs, entre ellas Tkinter.

Ilustración 9. Imagotipo LabVIEW.

LabView es una herramienta desarrollada por National Instruments que sirve para desarrollar sistemas con un lenguaje de programación gráfico, lo que simplifica crear aplicaciones, utilizando menos código y más lenguaje visual. Este lenguaje es similar a Simulink en MATLAB.

La ventaja principapal que presenta es su facilidad de uso, permite crear programas muy complejos que con otro software serían imposibles de crear. La velocidad para programar también es un punto fuerte de este software.

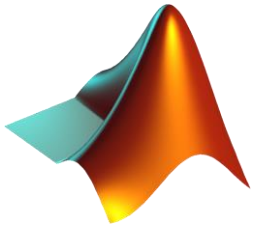


Ilustración 10. Logotipo MATLAB.

MATLAB es un entorno creado para trabajar originalmente con matrices y array, de ahí su nombre "MATrix LABoratory". Posee un lenguaje de programación propio, su uso está muy extendido en el campo de la ingeniería y otros muchos más gracias a que cuenta con muchas herramientas para multitud de procesos: aprendizaje automático, procesamiento de señales e imágenes, robótica, etc.

Las ventajas que posee este programa es la gran capacidad de cálculo matemático que tiene, sus Toolbox son de gran utilidad, es rápido y confiable. Cuenta con una gran comunidad de usuarios y en el foro de sus desarrolladores (MathWorks) hay infinidad de recursos y personas dispuestas a ayudar con sus conocimientos ante cualquier problema.

La mayor desventaja que tiene es que la gestión de memoria del programa no está optimizada y puede causar problemas de velocidad. Aunque existen versiones académicas este programa se utiliza en la industrial y la versión que se comercializa tiene un precio de licencia elevado.

Para la creación de GUIs MATLAB dispone de un conjunto de herramientas que facilitan la creación de estas interfaces. El comando GUIDE permite la cración de estas aplicaciones de manera interactiva y con infinidad de opciones de diseño y comportamiento de la app. MATLAB se encarga de generar el código correspondiente a medida que en la ventana de diseño gráfica se van añadiendo elementos a la interfaz, esto simplifica las tareas de desarrollo gracias a que únicamente se necesitan conocimientos básicos para ejecutar los procesos deseados para obtener resultados ya que la mayor parte de la programación estará hecha.

GUIDE posee la opción de comenzar a utilizar una interfaz ya creada y únicamente editar algunos aspectos, de manera que si se adecua a las necesidades del usuario ni siquiera es necesario realizar el diseño y la programación de la app.

Ilustración 11. Imagotipo Octave.

Octave es la opción más conocida para competir con MATLAB, los lenguajes de programación que utilizan son muy parecidos, busca compatibilidad total de los programas. Carece de muchas de las Toolbox que posee MATLAB y de la capacidad de funcionar con alto rendimiento en programsa completos, pero para la creación de GUIs, tarea que ocupa este trabajo de fin de grado, es un software bastante válido.

Ilustración 12. Imagotipo SageMath.

SageMath es una opción de código abierto que guarda mucha similitud con MATLAB, sus bibiliotecas están basadas en Python, software anteriormente descrito.

Ilustración 13. Imagotipo SciLab.

Junto con Octave es la alternativa más competitiva a día de hoy con MATLAB. Es compatible con las plataformas más utilizadas, Windows, Mac y Linux poseen afinidad con este software.

7.1 Elección de la solución

Para la elección de la solución utilizada para el desarrollo de la aplicación de este trabajo de fin de grado se han valorado las primeras opciones: Visual Basics, Python, LabView y MATLAB.

Los criterios escogidos para tomar la decisión sobre el programa a utilizar han sido los siguientes: herramientas disponibles para la creación de la GUI, que el propio software permite implementar la red neuronal construida en la aplicación, soporte de ayuda por parte de usuarios y desarrolladores y precio de las licencias.

Cada criterio se ha valorado con un número entre 1 y 5, siendo 1 poco adecuado o malo y 5 la opción óptima.

	Desarrollo GUI	Implantación red	Soporte	Precio de la licencia	TOTAL
Visual Basic	4	1	2	2	9
Python	3	3	4	5	15
LabView	4	2	4	1	11
MATLAB	4	5	5	3	17
Octave	3	3	3	5	14
Sagemath	2	2	3	5	12
Scilab	2	3	2	5	12

Tabla 9. Elección de la solución.

Como se muestra en la tabla la opción selecciónada es MATLAB, ha sido clave para optar por esta solución que los programas de creación y validación de la red neuronal que forman parte de los datos de partida estén programados en MATLAB. Otro aspecto importante es que aunque sea una herramienta comercial con un precio de licencia elevado la EHU tiene licencias de estudiante para los alumnos.

8 Diseño aplicación

8.1 Programas utilizados en la red neuronal

Estos datos son procesados gracias a inteligencia artificial, en este caso se han utilizado redes neuronales artificiales para ello. La red que se utiliza en este trabajo ya ha sido construida y validada anteriormente por otro estudiante, este proyecto se centra en la implementación de la red y la creación de conjuntos de datos para el entrenamiento desde una aplicación.

Toda la ejecución se lleva a cabo en MATLAB, este software informático posee una Toolbox propia para realizar el entrenamiento de la red neuronal.

Para realizar entrenamientos con la red neuronal se puede dividir la tarea en dos fases: la primera se ejecuta para la obtención de conjuntos de datos con los que realizar el entrenamiento, mientras que la segunda se puede identificar como el entrenamiento en si de la red.

Para la primera fase se utilizan los siguientes programas: "CrearTablaPorParticipantes", "Crear_Conjuntos_Por_Participantes" y "Crear_Conjunto_Entero_Solo1Bloque". Se explica a continuación en que consiste cada uno.

- CrearTablaPorParticipantes
- Crear_Conjuntos_Por_Participante
- Crear_Conjunto_Entero_Solo1Bloque

Una vez obtenidos los conjuntos de datos necesarios se realiza el entrenamiento con los siguientes programas: "RedNeuronalArtificial","RedNeuronalArtificialConTest" y "Entrenar_RNA_Final".

- RedNeuronalArtificial
- RedNeuronalArtificialConTest
- Entrenar_RNA_Final

8.2 Funcionamiento aplicación

La aplicación que se muestra a continuación se ha creado en el entorno de computación MATLAB, utilizando la herramienta GUIDE que permite crear una interfaz partiendo desde cero añadiendo elementos gráficos diferentes y con capacidad de variar todas las características de aspecto y diseño.

La estructura por módulos del programa es la siguiente:

1. Selección datos de entrada y creación del conjunto "Samples.mat" con esta información.
2. Ajuste parámetros de configuración de la red y entrenamiento. Creación de los conjuntos de datos "Conjunto.mat" y "Red.mat".

3. Mostrar resultados. Muestra en pantallas los resultados más relevantes y guarda en la memoria de MATLAB un nuevo conjunto de datos que tendrá un nombre creado automáticamente en función de la configuración seleccionada.

La interfaz general del programa, separada en los tres módulos mencionados anteriormente, es:

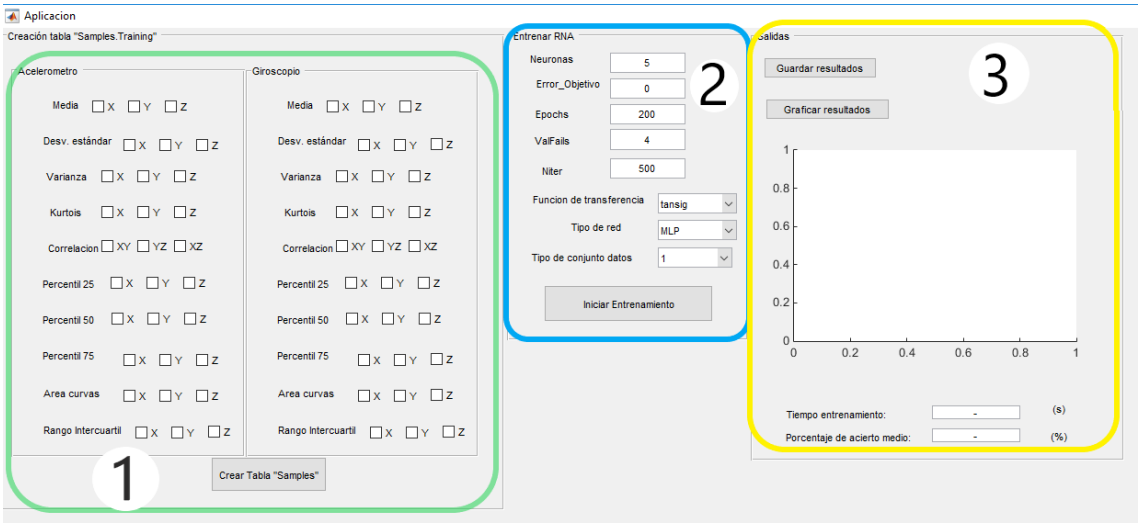


Ilustración 14. Aspecto Aplicacion.fig

Cuando se ejecuta el archivo Aplicacion.m de tipo Matlab Code aparece en pantalla la imagen anterior, que es el la interfaz de la aplicación desde la cual se realizan todas las tareas programadas.

Las tareas que facilita esta interfaz son tres: selección de los datos para el entrenamiento, ajuste de los parámetros de la red neuronal y obtención de resultados. Cada una se ha separado en un panel diferente para que la ejecución de las tareas sea más fácil de seguir.

Para su diseño se ha pensado en lo siguiente: que sea intuitiva, fácil de utilizar, simple y funcional.

En la carpeta de ejecución se encuentran los siguientes archivos:

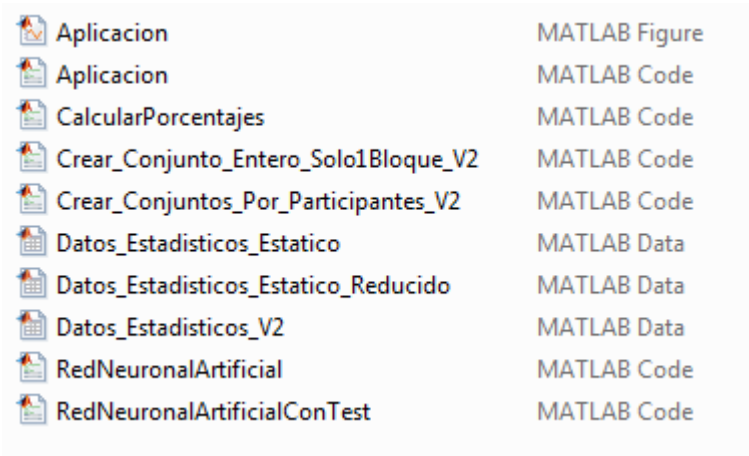


Ilustración 15. Archivos utilizados para la GUI.

Los archivos de tipo "MATLAB Data" Datos_Estadisticos_Estatico, Datos_Estadisticos_Estatico_Reducido y Datos_Estadisticos_V2 contienen la información de las variables estadísticas recogidas gracias a los sensores de la muleta, posteriormente se han filtrado y obtenido estas estructuras de datos que se utilizan para el entrenamiento.

Los programas "CalcularPorcentajes", "RedNeuronalArtificial", "RedNeuronalArtificialConTest" han sido creados y validados por otro estudiante, desde la aplicación se realizan llamadas a estos para realizar los entrenamientos. "Crear_Conjunto_Entero_Solo1Bloque_V2" y "Crear_Conjuntos_Por_Participantes_V2" también han sido creados por otro estudiante, la extensión "_V2" ha sido añadida porque estos archivos se han modificado para adaptarlos al funcionamiento de la aplicación.

El archivo Aplicacion tipo "MATLAB Code" contiene el código autogenerado por GUIDE y el necesario escrito para el para el funcionamiento, en "MATLAB Figure" se encuentra el aspecto de la interfaz, este último es editable en GUIDE de MATLAB. Desde esa función se añaden nuevos elementos, se edita el aspecto, etc. Cada elemento añadido genera automáticamente su código correspondiente en el archivo .m para que se corresponda en todo momento el código con la aplicación.

8.2.1 Módulo 1. Selección de datos de entrada.

Desde este primer panel se seleccionan las variables estadísticas que se quieren introducir al programa para trabajar con las redes neuronales. La programación está hecha de tal manera que la salida de este módulo es una estructura de datos llamada "Samples.mat", contiene una copia del archivo original de "Datos_Estadisticos.mat" con los datos seleccionados por las casillas.

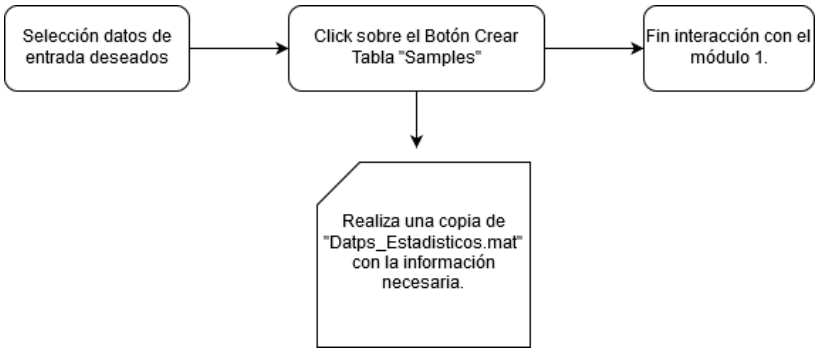


Ilustración 16. Diagrama de flujo del módulo 1.

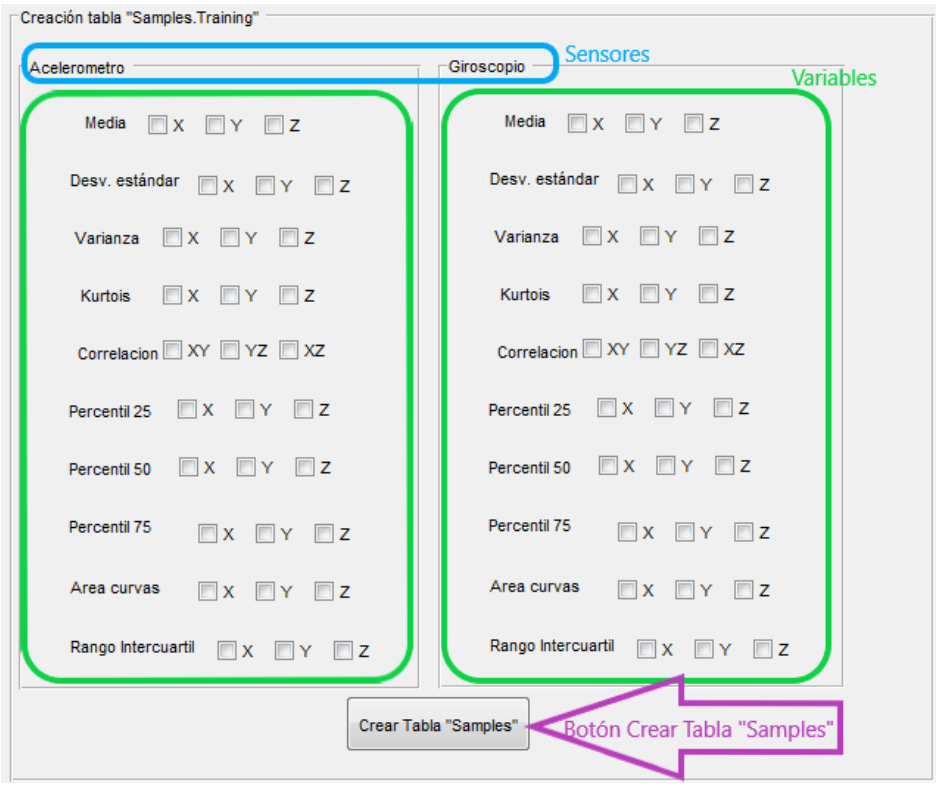


Ilustración 17. Panel 1-Cargar datos.

En este panel aparecen dos tablas con los nombres de los sensores Acelerómetro y Giroscopio. Dentro de cada uno se encuentran los nombres de las variables estadísticas que se utilizan para el estudio de las redes. Contiguos a cada variable aparecen los ejes donde se han tomado las medidas. Las casillas de verificación (checkboxes) permiten al usuario ir seleccionando de cada variable los ejes de los que desea obtener información.

En la parte inferior del panel se encuentra el **Botón "Crear Tabla Samples"**, este pushbutton es el encargado de la creación del archivo de tipo "Samples.mat" con los datos que hayan sido seleccionados. La interacción con este panel debe ser la primera acción a realizar, ya que sin este archivo de datos de entrada no se puede realizar ninguno de los procesos posteriores.

Si en la carpeta donde se está ejecutando la aplicación ya existe un archivo "Samples.mat" que se haya utilizado anteriormente y se quiere realizar un nuevo entrenamiento utilizando esos mismos datos, se puede saltar el paso de ejecutar el botón que se encuentra en este panel, ya que no será necesario crear una nueva tabla.

Con el objetivo de mantener un orden y poder detectar fallos durante la programación, a cada chekbox se le ha asignado un nombre de acuerdo al código descrito a continuación.

Posición término	Primero	Central	Final
Referencia	Sensor.	Variable estadística.	Eje de la medida.
Ejemplos:	A (acelerómetro) o G (giróscopo).	Utiliza la primera letra de cada variable. M (media), D(desviaciones), P25(percentil 25), etc.	X-Y-Z o XY-YZ-XZ para el caso de correlaciones.

Tabla 10. Código nombres chekbox.

Respecto al código, la programación de este primer panel se realiza debajo de la función correspondiente al botón para que todo se ejecute una vez se pulsa.

```
function Boton_Crear_Tabla_Callback(hObject, eventdata, handles)
%Al pulsar el boton de crear tabla:
```

Lo primero que realiza el programa es cargar las entradas "Datos_Estadisticos_V2" y "Datos_Estadisticos_Estatico_Reducido" y se inicializa la variable Samples, que se rellena posteriormente con los datos correspondientes a las varibles cuyo estado de chekboxes se encuentre activo.

La copia de datos se realiza mediante un bucle for que recorre todas las filas del archivo "Datos_Estadisticos_Cell_V2". Los datos se recorren en función del tipo de movimiento en el siguiente orden: andar recto, subir escaleras, bajar escaleras y estático.

En cada etapa se evalúa el estado del chekbox correspondiente, en caso de estar activo se procede a copiar su contenido en el archivo de salida. Para que la evaluación con los datos de salida sea lo mejor posible y el programa se ejecute sin fallos es necesario que no existan columnas sin rellenar o con datos nulos. Para solucionar este problema se ha optado por crear unas variables que cuenten el número de columnas que se están ocupando en cada momento.

```
%Se inicia en la columna "1" para guardar las salidas de
Samples.Training{k,1}.In(1:f,"1"), desde ese 1 hasta el número total
de columnas=indicadores escogidos

Candar=1;    %columnas andar recto
Csubir=1;    %columnas subir escaleras
Cbajar=1;    %columnas bajar escaleras
Cest=1;      %columnas estático
```

Tras esta etapa de carga de variables de entrada e inicialización de las que serán de salida se entra en el bucle for, donde para cada checkbox se efectúa la siguiente comprobación:

```
if get(handles.AMX,'Value')==1
            Samples.Training{k,1}.In(1:f,Candar) =
Datos_Estadisticos_Cell_V2{i, 1}.Accel.Andar_Recto(:,1); %Accel X
            Candar=Candar+1;
 end
```

El ciclo if se efectúa si el chekbox correspondiente marca el estado 1, que corresponde a casilla verificada. En ese caso se lleva a cabo la copia de información en la columna correspondiente y se incrementa en uno el valor de Candar/Csubir/Cbajar/Cest para que el siguiente dato se almacene en la columna posterior, evitando que queden algunas en blanco o se sobreescriba información.

8.2.2 Módulo 2. Configuración y entrenamiento de la RNA.

El segundo módulo se ejecuta desde el panel con nombre "Entrenar RNA". Se configuran los parámetros de entrenamiento y el tipo de red que se utiliza. Los datos introducidos a través de la interfaz se comprueban en el programa y si son correctos se procede a iniciar el entrenamiento a través del botón "**Iniciar Entrenamiento**".

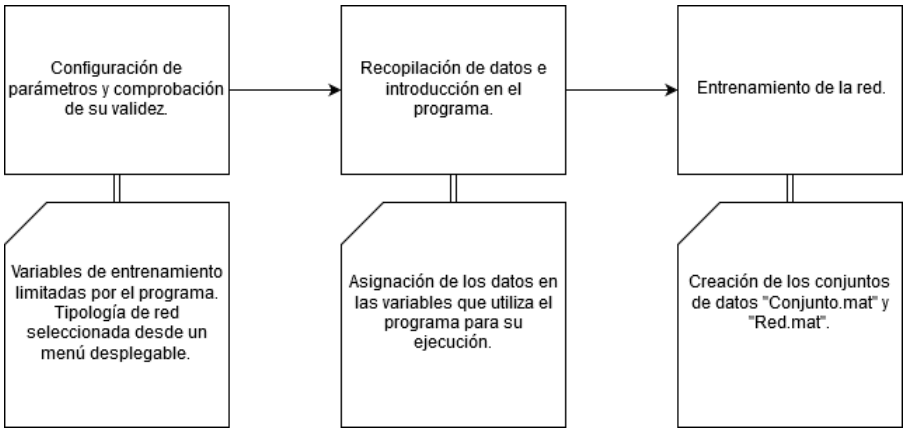


Ilustración 18. Diagrama de flujo para el módulo 2.

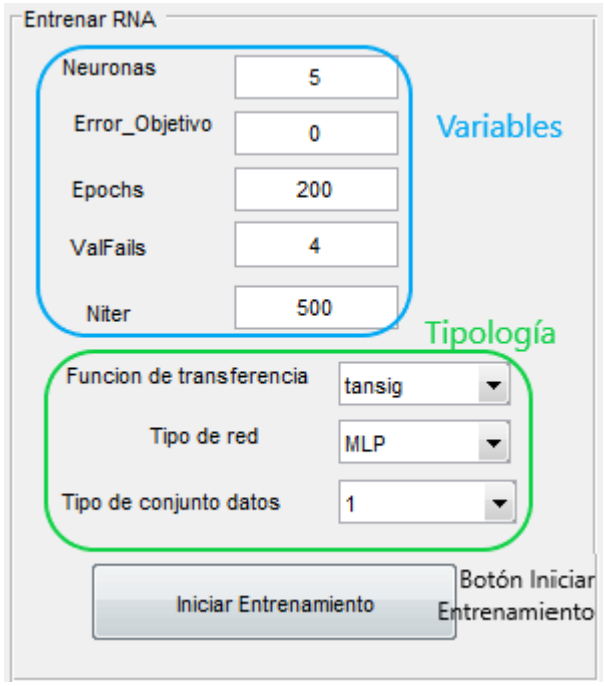


Ilustración 19. Panel 2- Entrenar RNA.

ASIGNACIÓN DE VARIABLES

El paso posterior a la creación de la tabla de datos "Samples.mat" es la introducción de los valores característicos de la red y los parámetros para realizar el entrenamiento. El paso final en este panel es el inicio del entrenamiento de la red, que se realiaza mediante el botón inferior **"Iniciar Entrenamiento".**

8.2.2.1 Estructura de la red

Las variables propias de la red son:

- Neuronas que posee la capa oculta.
- Error_Objetivo, es el deseado por la red, se asigna por defecto 0 aunque nunca llega a alcanzarlo.
- Epochs, máximo número de iteraciones.
- ValFAils, se utiliza para las redes con Test.
- Niter, número de iteraciones.

Los valores a estas variables los asigna el usuario desde teclado, mediante la herramienta "Edit Text" de MATLAB, la cual permite escribir desde teclado dentro de la aplicación.

Para utilizar los valores durante la programación es necesario cambiar el formato de los números, ya que al introducirlos el programa capta el valor como una variable de tipo String, para pasarlo a formado doble la primera parte de asignación utiliza el comando str2double. El comando handles.*NombreEditText* se utiliza para referenciar a cada tramo de EditText que ha sido creado fuera de la función de entrenamiento.

```
Neuronas = str2double(get(handles.Neuronas,'String')) ;
Error_Objetivo= str2double(get(handles.Error_Objetivo,'String')) ;
Epochs= str2double(get(handles.Epochs,'String')) ;
ValFails= str2double(get(handles.ValFails,'String')) ;
Niter= str2double(get(handles.Niter,'String')) ;
```

El entrenamiento se puede realizar dando cualquier valor a las variables, por eso a algunas de estas se les ha puesto un límite para impedir que se inicie un entrenamiento cuyo proceso se alargue mucho en el tiempo o que los resultados que se obtengan sean muy malos.

Por ejemplo para la red con los parámetros que se han puesto por defecto (tansig, MLP, Conjunto1 con 5 neuronas, 200 épocas, 500 iteraciones) se obtienen unos altos porcentajes de acierto (67.53%) para un tiempo de entrenamiento de unos veinte minutos aproximadamente.

Los valores que las variables toman se han limitado introduciendo en el código de programa nuevos parámetros NombreDeLaVariableMIN y NombreDeLaVariableMAX. Mediante un bucle se compara el valor introducido con los límites establecidos, si el valor se encuentra fuera de

rango no se inicia el entrenamiento gracias a poner la variable TipoConjunto a cero, que es la encargada de crear el conjunto y hacer la llamada a la red.

```
NeuronasMIN= 1 ;
NeuronasMAX= 20 ;
if Neuronas > NeuronasMAX || Neuronas < NeuronasMIN
    warndlg('El número de neuronas no es adecuado.') ;
TipoConjunto=0;
end
```

Los mensajes de aviso se muestran de la siguiente manera en pantalla al pulsar el botón de iniciar entrenamiento.

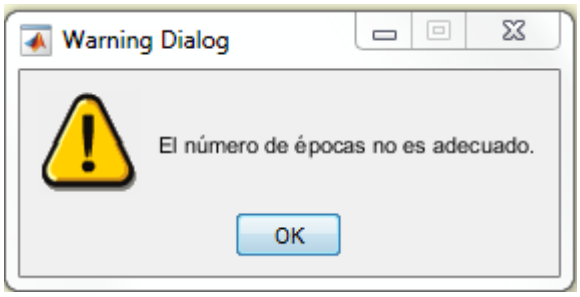


Ilustración 20. Aviso parámetros fuera de rango.

8.2.2.2 Tipo de red

El tipo de red que se utiliza para el entrenamiento se escoge justo debajo, se caracteriza con :

- Función de transferencia, se utiliza en la activación de la capa oculta tansig/radbas.
- Tipo de red, MLP/RBF
- Conjunto de datos 1/2 . El conjunto 1 crea una estructura de datos divida en entrenamiento y validación a partir de la estructura de entradas y salidas previamente construidas, utiliza para ello la función "Crear_Conjuntos_Por_Participantes_V2". El conjunto de datos tipo 2 utiliza la función "Crear_Conjunto_Entero_Solo1Bloque_V2". Este segundo tipo de conjunto será el que se utilice cuando la red a entrenar necesite datos para Test además de Entrenamiento y Validación.

Estas tres variables se escogen desde una ListBox, de esta manera se fijan las opciones de singularizar la red.

La toma de datos de estas variables se realiza de manera diferente a la que se utiliza con los EditText. En este caso primero se toman los valores que están escritos como opciones con el siguiente comando, se muestra con un ejemplo:

```
Funcion_Transferencia=cellstr(get(handles.Funcion_Transferencia,'Strin
g')) ;
```

"cellstr" guarda en la variable Funcion_Transferencia en forma de array of strings los valores que contiene la celda como opciones.

"get(handles.Funcion_Transferencia,'String')" obtiene la característica "String" que se encuentra en el elemento ListBox con nombre Funcion_Transferencia.

Posteriormente hay que escoger uno de los valores que se han guardado en la variable, para ello se acude a obtener el "Value" de la ListBox, que corresponde con un número que indica si se ha escogido la primera o segunda opción. Se reutiliza la misma variable pero una vez se conoce la posición de la opción necesaria se coloca entre corchetes y queda guardado con el nombre de la variable solo la opción seleccionada desde la lista.

```
Funcion_Transferencia=Funcion_Transferencia{get(handles.Funcion_Transf
erencia,'Value')} ;
```

Para el procesamiento interno en la red se utilizan números en lugar de nombres para identificar las opciones. Por ejemplo para la función de transferencia se utiliza el número 1 para el caso de "tansig" o 2 en el caso de "radbas".

Para realizar esta conversión se ha utilizado la sentencia case de MATLAB.

```
switch Funcion_Transferencia
    case 'tansig'
        FT_Elegida=1;
    case'radbas'
        FT_Elegida=2;
end
```

Con la variable que determina el tipo de red el procemiento es igual, en la variable NetType se almacena el nombre de la red escogida (MLP/RBF) y con una sentencia posterior switch-case se guarda en NetType_Elegido los valores 1 o 2 para utilizar en las partes siguientes del código.

8.2.2.3 Entrenamiento

Definidas todas las variables y el tipo de red se procede a la llamada a la creación y entrenamiento de la neuronal, dependiendo del tipo de conjunto utilizado se ejecutan unas acciones u otras.

- Para el tipo de conjunto 1 se utiliza “Crear_Conjuntos_Por_Participantes_V2” para crear el archivo “Conjunto.mat” y la llamada a la red es mediante “RedNeuronalArtificial”, ya que en este caso no se utilizan datos para Test.
- En el caso de tipo de conjunto 2 los programas que se utilizan son “Crear_Conjunto_Entero_Solo1Bloque” y “RedNeuronalArtificialConTest”.

La última acción que el usuario puede ejecutar desde este panel es a través del botón “Iniciar Entrenamiento”, este botón ejecuta el entrenamiento propiamente dicho gracias a una ToolBox de MATLAB para redes neuronales mediante “nntraintool”.

Cuando se inicia el entrenamiento aparece una nueva GUI con el siguiente aspecto:

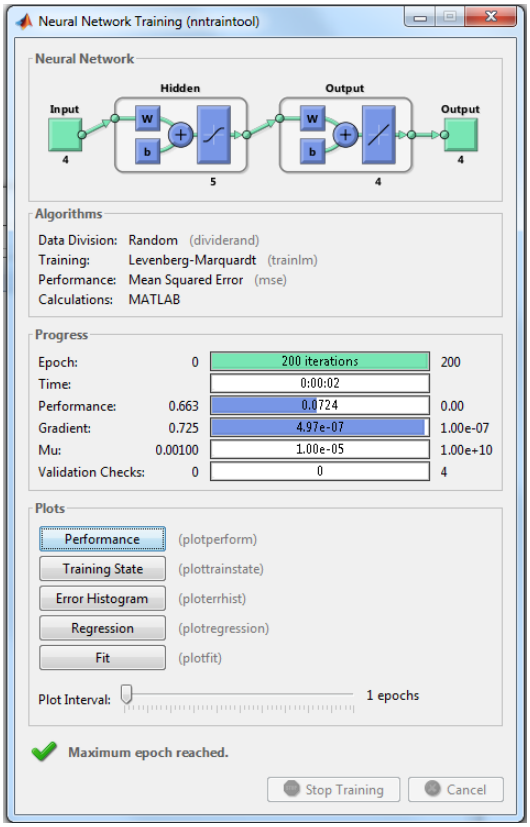


Ilustración 21. Neural Network Training. Realiza el seguimiento del entrenamiento.

A través de esta segunda interfaz se puede visualizar de manera esquemática el progreso del entrenamiento gracias las barras horizontales que hay,

Existe la posibilidad también de ver en diferentes plots otros aspectos del entrenamiento.

Acabado el entrenamiento quedan en la memoria de MATLAB los archivos de datos “Red.mat” y “Conjunto.mat” con los resultados del entrenamiento y los datos ordenados para esto, respectivamente.

8.2.3 Módulo 3. Mostrar resultados.

El último panel de la aplicación permite guardar la información de las redes con los porcentajes de acierto y mostrar los resultados sobre la interfaz.

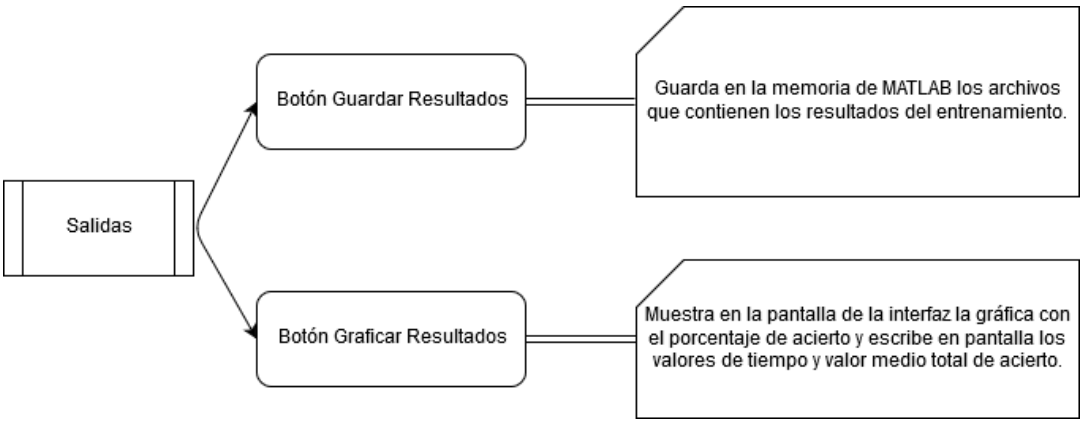


Ilustración 22. Diagrama de flujo del módulo 3.

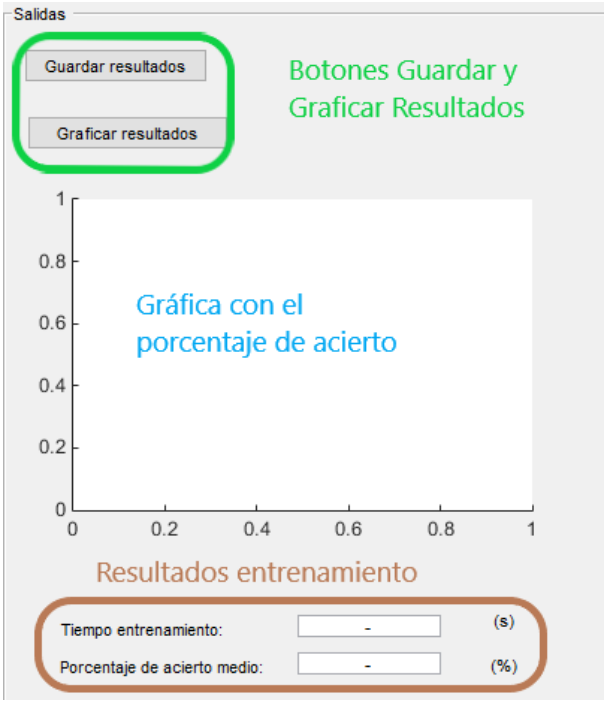


Ilustración 23. Panel 3-Resultados

La forma de actuar de este panel es únicamente por botones, sirve para guardar los resultados, los mejores porcentajes de entrenamiento y mostrar una gráfica con el porcentaje de acierto

global, que es la más significativa del entrenamiento. Además muestra el tiempo que ha necesitado el programa para realizar el entrenamiento y el valor de acierto global medio.

Para el cálculo del porcentaje de acierto global se deben tener en cuenta que las salidas de la simulación de la red no dan valores exactos, los resultados son valores entre 0 y 1 que se deben redondear. En esta aplicación se ha optado por redondear en el punto medio, los valores menos de 0.5 se entienden como 0 y los mayores como 1. Una vez los resultados se han redondeado al extremo más próximo se compara la salida en la simulación con el estado real. Una variable llamada AciertosGlobales va incrementando su valor cada vez que al recorrer los conjuntos de validación coinciden los valores reales con los obtenidos. Una vez terminados de comparar los conjuntos se puede calcular el PorcentajeDeAciertoGlobal dividiendo los AciertosGlobales entre el número de datos utilizados y multiplicando por 100 para obtener el valor porcentual.

El **Botón Guardar Resultados** para guardado de resultados sirve para crear y guardar una nueva variable con nombre específico y que contiene valores significativos del entrenamiento.

El **Botón Graficar Resultados** dibuja sobre el elemento axes el porcentaje de acierto global en ordenadas frente a la variables número de iteraciones en abscisas. También escribe, si el resultado del entrenamiento lo permite, los valores de tiempo consumido y valor del porcentaje de acierto medio.

Los resultados del entrenamiento se guardan en diferentes variables, la estructura Red contiene toda la información que se genera durante la prueba, se almacena en una estructura con la siguiente distribución:

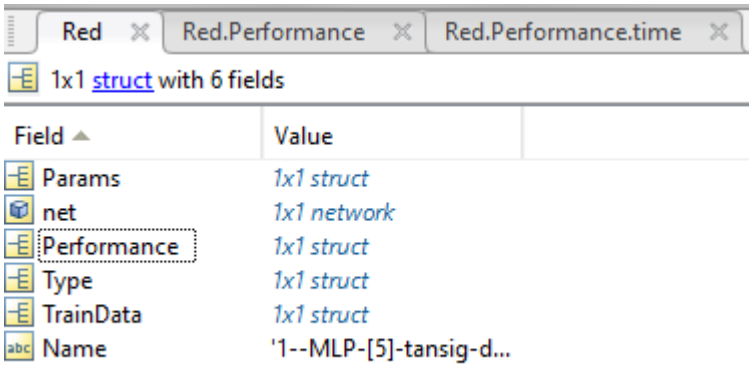


Ilustración 24. Estructura Red.mat

Dentro de "Red.Performance" se guardan muchas variables que aportan datos de cómo ha sido el entrenamiento:

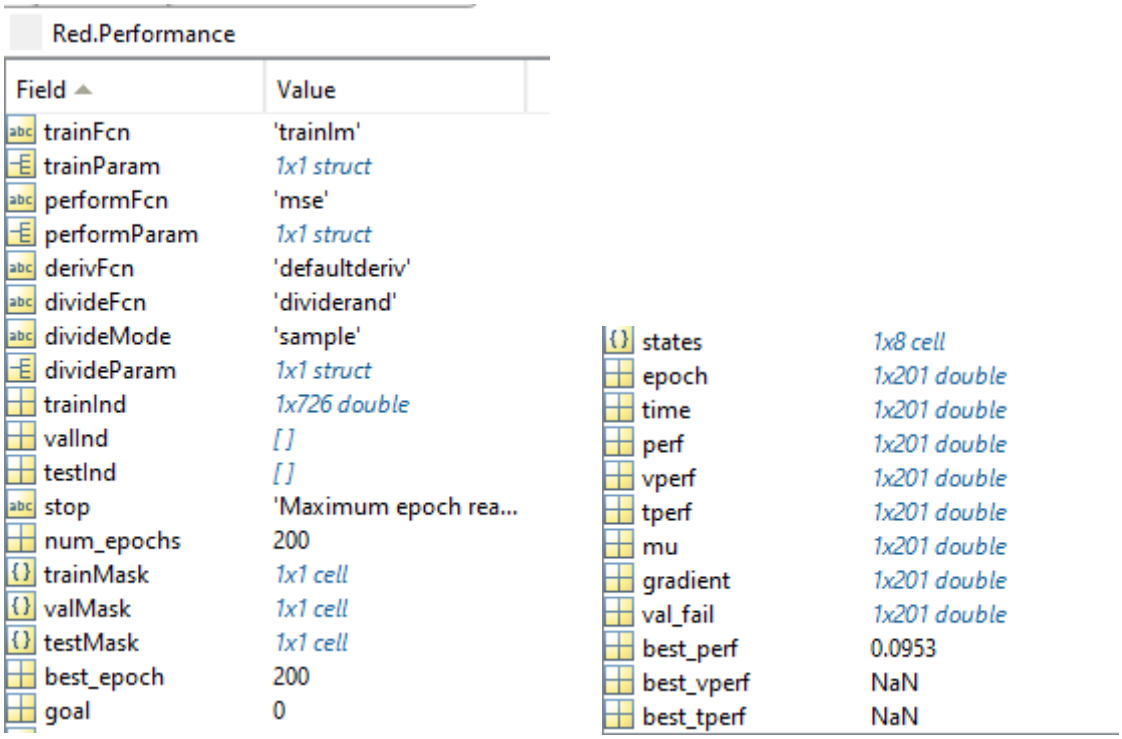


Ilustración 25. Estructura Red.Performance

A continuación se define que contienen los campos que son de utilidad para el estudio de redes de todos los que aparecen en la ilustración anterior:

- trainFcn. Función que actualiza los valores de peso. En este caso se utiliza el algortimo de Levenberg-Marquardt (lm para simplificar).
- performFcn. Tipo de función para medir el rendimiento de la red.
- performParam, derivFcn, divideFcn, divideMode. Contienen los nombres de las fórmulas utilizadas para derivar y dividir los datos de entrenamiento, validación y test.
- trainInd, valInd, testInd. Índices utilizados para entrenamiento, validación y test.
- stop. Define el motivo por el que el entrenamiento ha finalizado, puede contener avisos de un problema de ejecución o que se han alcanzado las iteraciones propuestas.
- num_epochs y best_epoch. Contienen el número de épocas y la que ha obtenido un mejor resultado.
- states. Contiene los nombres de las salidas que se van a utilizar: 'epoch', 'time', 'perf', 'vperf', 'tperf', 'mu', 'gradient', 'val_fail'
- epoch. Lista de números ordenados partiendo de 0 hasta el número de épocas utilizadas para realizar el entrenamiento.
- time. Tiempo empleado en cada época.
- perf. Rendimiento en cada época.

- mu. Parámetro controla los pesos de las neuronas en el proceso de (backpropagation), un nuevo entrenamiento dejará de ser útil si se detiene por alcanzar un valor máximo de mu.
- gradient. Gradiente entre cada entrenamiento,valor que toma la pendiente de la gráfica de entrenamiento.
- val_fail. Solo se utiliza en conjuntos con test, por lo que en esta app no otorga información.
- best_perf, best_vperf, best_tperf. Contienen las actuaciones de la mejor de las redes.

Para mostrar el tiempo total consumido durante el entrenamiento de la red se han utilizado los valores contenidos en "Red.Performance.time", aquí se encuentra en un array el valor de tiempo que durante cada época ha sido consumido, para sacar el valor total se escribe las siguientes líneas de código bajo el botón de Graficar Resultados:

```
TiempoEntrenamiento =num2str(sum(Red.Performance.time)) ;
set(handles.TiempoEntrenamiento,'String',TiempoEntrenamiento) ;
```

La primera línea asigna a la variable "TiempoEntrenamiento" el valor de la suma de todos los tiempo guardados en Red.Performance.time. Lo convierte en tipo "String" para mostrarlo en la interfaz en la casilla EditText nombrada como la variable donde se guarda, TiempoEntrenamiento.

Para obtener el valor medio del porcentaje de acierto se procede de manera similar, pero en lugar de buscar valores en la estructura Red, ahora están en PorcentajeDeAcierto.

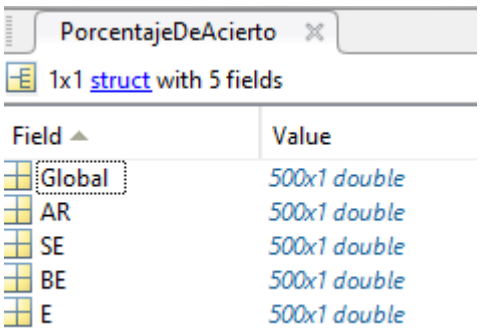


Ilustración 26. Estructura PorcentajeDeAcierto.

Aquí se guardan los porcentajes de acierto de la red en cada iteración, para el caso mostrado en la figura anterior se han utilizado los valores que aparecen en la aplicación por defecto, 500 iteraciones en este caso.

En esta ocasión la programación para obtener el valor se lleva a cabo de la siguiente manera:

```
PorcentajeFin=num2str(sum(PorcentajeDeAcierto.Global)/numel(Porcentaje
DeAcierto.Global));
set(handles.PorcentajeAcierto,'String',PorcentajeFin) ;
```

La única diferencia con el caso anterior es que en esta ocasión después de sumar todos los porcentajes obtenidos es necesario dividir entre el número total de datos que se han utilizado, para esto se utiliza el comando "numel" de MATLAB, que devuelve el número de elementos que contiene cierto array.

Las salida de un entrenamiento con los parámetros por defecto se muestran a continuación, la gráfica y valores de salida corresponden con los que se encuentren en ese momento en la memoria del programa.

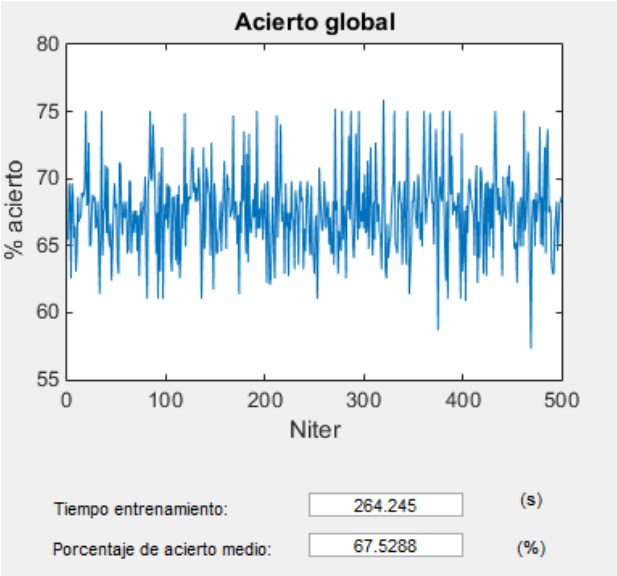


Ilustración 27. Pantalla de resultados tras el entrenamiento.

.

Una vez finalizado el entrenamiento se puede continuar escogiendo nuevos parámetros, el programa está creado para que las nuevas variables que se crean en "Samples", "Conjuntos" y "Red" se sobreescriban cada vez que se pulse un nuevo botón encargado de cargar nuevos datos.

8.3 Programas utilizados en redes neuronales

En este apartado se explican con más detalle los programas de los que la aplicación hace uso durante todo el proceso de ejecución. Todos ellos se utilizan en el módulo dos de la aplicación, son aquellos que tienen que ver con el uso de la red neuronal en sí. Como se ha comentado antes todos ellos han sido desarrollados y validados por otro estudiante. Mi trabajo con estos programas ha sido de comprensión y adaptación de pequeños fragmentos de código para que el funcionamiento sea el correcto.

En la creación de conjuntos de datos para el entrenamiento (estructuras Conjunto.mat) se utilizan dos programas:

- Crear_Conjuntos_Por_Participantes_V2: en caso de que la variable "TipoConjunto" escogida desde el menú fuese el número uno. En este caso se proporcionan a la red los datos divididos en tres tipos: entrenamiento, validación y test. (aunque en esta app solo se utilizan los dos primeros). Aproximadamente un 60% de los datos se utilizan para el entrenamiento y el 40% restante para validación. Dentro de cada variable se distinguen también dos tipos (In and Out), que se explican en el siguiente programa.

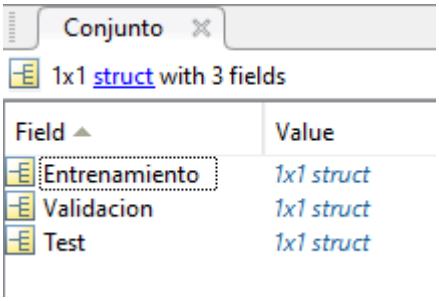


Ilustración 28. Estrutura de datos Conjunto.mat para la variable TipoConjunto igual a uno.

- Crear_Conjunto_Entero_Solo1Bloque_V2: corresponde al conjunto de datos tipo dos. En este caso a partir de la tabla de datos Samples crea una nueva estructura de datos de entrada a la red sin diferenciar entre entrenamiento y validación. Las entradas son los valores de las variables escogidas en el panel del módulo uno (en esta caso se han escogido dos variables estadísticas, de las cuales si dispone de 1319 datos por cada una). Las salidas son unos o ceros según el tipo de movimiento sea andar recto, subir o bajar escaleras y estático (cuatro clasificadores de movimiento equivalen a cuatro columnas en la salida).

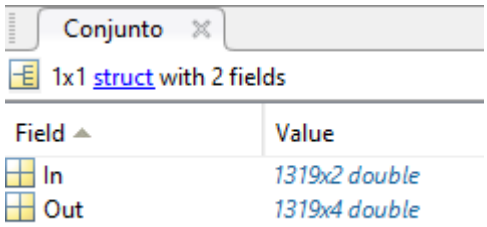


Ilustración 29. Estructura de datos Conjunto.mat para la variable TipoConjunto igual a dos.

Para el funcionamiento del tercer panel se utilizan los siguientes scripts:

- RedNeuronalArtificial: en este script se construye una red supervisada, que realiza el proceso de entrenamiento de manera iterativa para quedarse con la mejor salida (mejores porcentajes de acierto). Todos los datos que utiliza son para entrenar,
- RedNeuronalArtificalConTest: en este script también se construye una red neuronal que trabaja de manera supervisada, pero en esta ocasión tiene en cuenta el conjunto test.

Como puede verse los dos scripts solo se diferencian en la forma de utilizar los conjuntos de datos. Las llamadas a los entrenamientos de las dos redes neuronales se realizan igual, para los casos donde el tipo de red escogida sea MLP (perceptrón multicapa) el entrenamiento comienza una vez introducidas las variables con la funcion "train" de MATLAB.

```
[net,tr,Y,E,Pf,Af] = train(net,P,T);
```

Para los casos donde el tipo de red escogida sea RBF (redes función de base radial) el entrenamiento se realiza crando una red de base radial que unicamente pasa sobre los datos. En la variable P se guardan los datos de entrada y en T los de salida.

```
net = newrbe(P,T);
```

9 Pruebas

Se realizan un par de ejemplos de entrenamiento para comprobar que el programa funciona correctamente e ilustrar su funcionamiento.

9.1 Red MLP (Perceptrón multicapa)

Se utilizan catorce muestras de datos del archivo "Datos_Estadisticos.mat" para crear los conjuntos de entrenamiento, estos han sido :

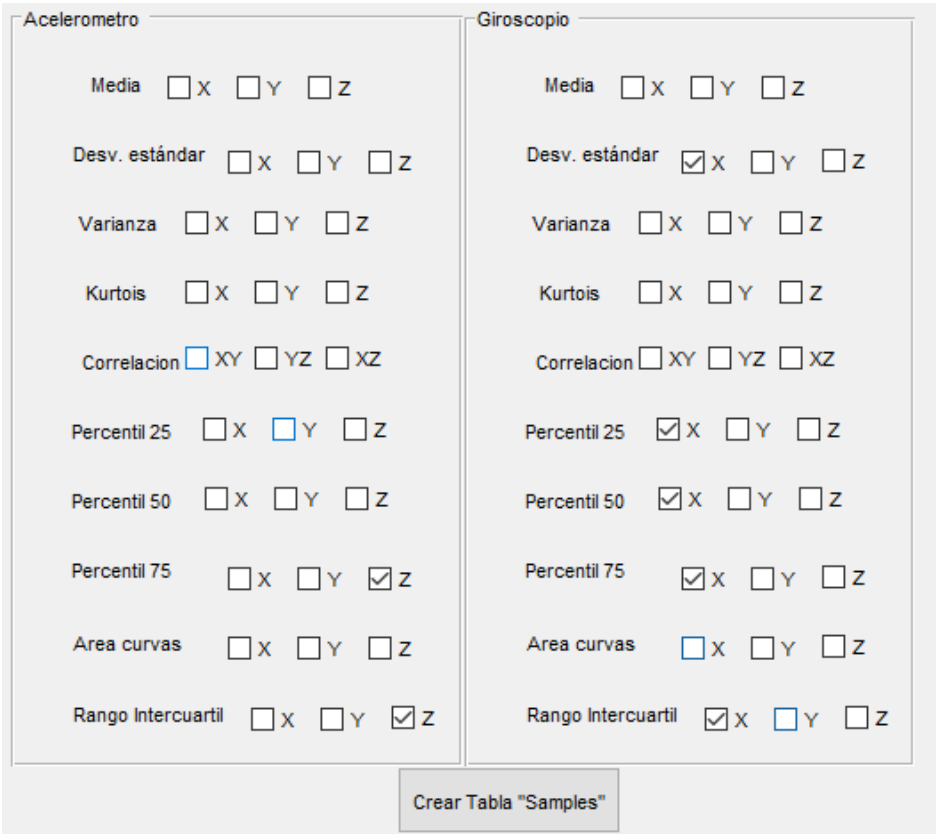


Tabla 11. Datos utilizados entrenamiento por defecto.

Mientras que los parámetros de la red son:

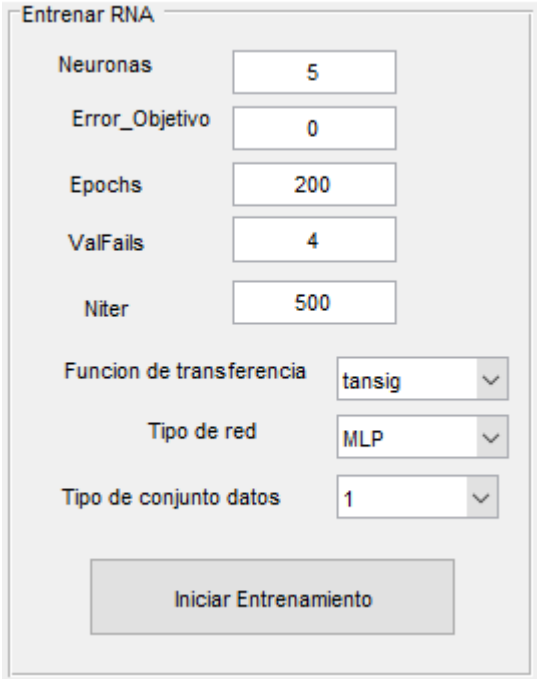


Tabla 12. Parámetros para entrenar red.

Con la pulsación del botón Iniciar Entrenamiento aparece en pantalla la ventana de MATLAB correspondiente a Neural Network Training (nntraintool). Esta ventana permite visualizar lo que está sucediendo en cada momento en diferentes gráficas, algunas de estas son:

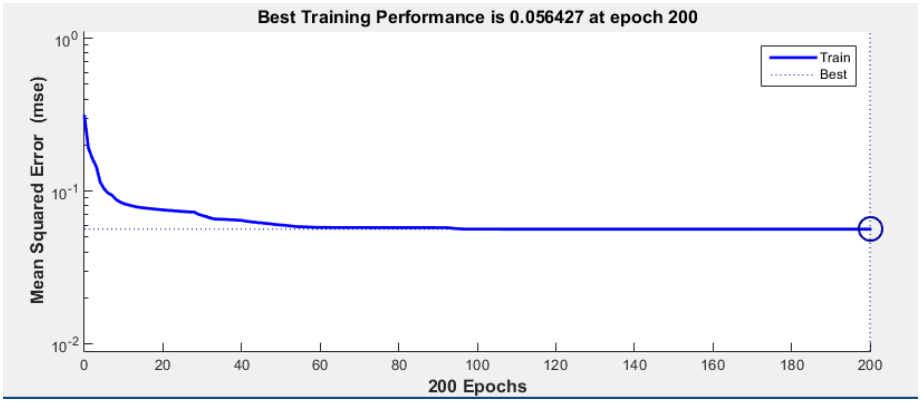


Ilustración 30. MPL Train Performance.

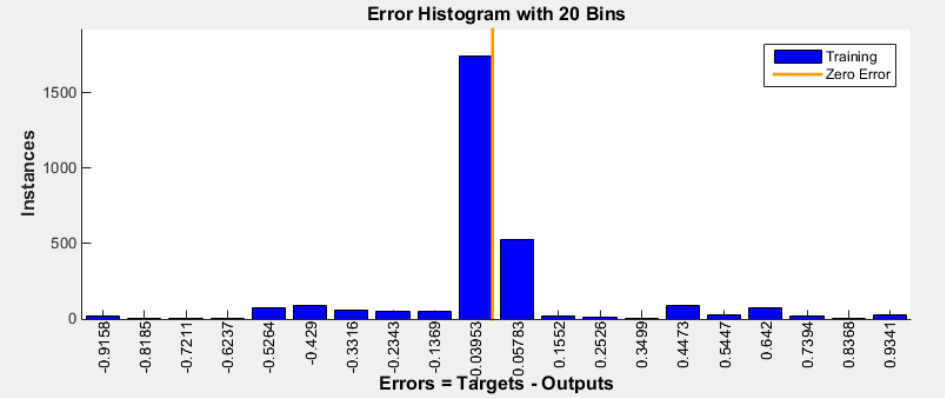


Ilustración 31. MPL Error Histogram.

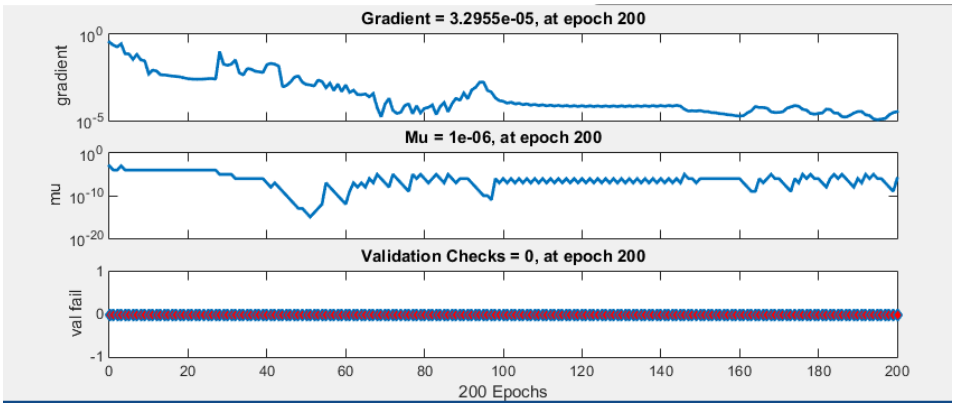


Ilustración 32. MPL Train State.

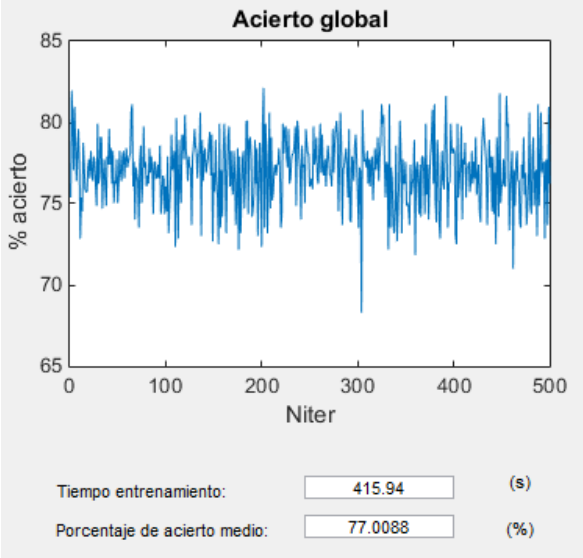


Ilustración 33. MPL Resultados.

9.2 Red MLP (Perceptrón multicapa) utilizada para comprobaciones.

Esta red ha sido utilizada mientras se desarrollaba el programa, para reducir el tiempo de espera se utilizaban los valores mínimos de configuración de la red.

Como puede verse los resultados de porcentaje de acierto se reducen de manera severa a costa de un tiempo de entrenamiento muy breve.

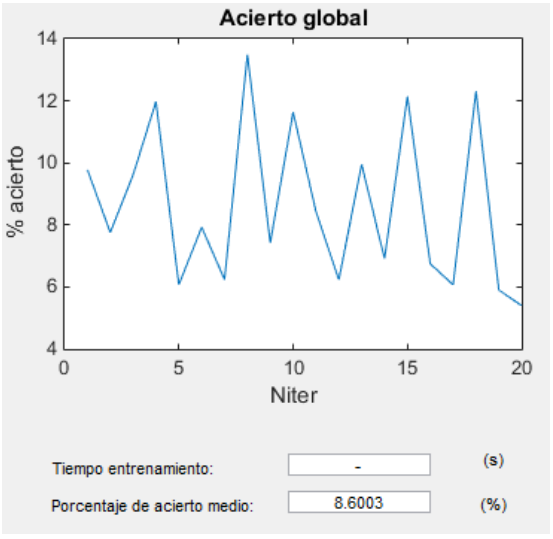


Ilustración 34. Resultados de una red MLP para comprobaciones.

9.3 Red RBF (función de base radial)

Para este tipo de red se obtiene un porcentaje de acierto constante.

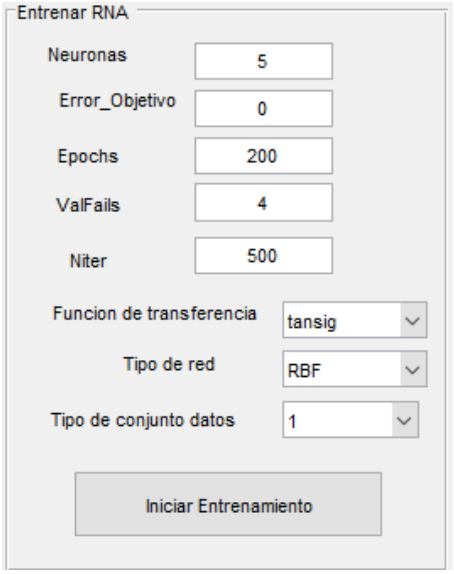


Ilustración 35. Parámetros RBF.

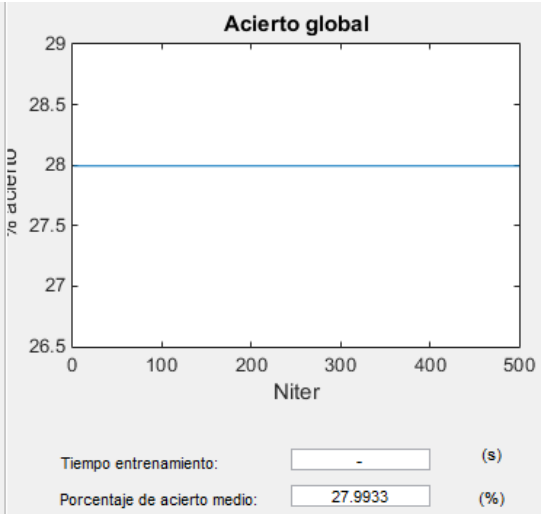


Ilustración 36. Resultados RBF.

10 Descripción de tareas

- Gestión del trabajo con el tutor y el grupo de investigación

El grupo de alumnos y profesores que trabajan desde hace unos años en el proyecto SMARTIP se reúne los martes de cada semana para comentar y poner en común los avances de cada persona con su trabajo. Desde que el trabajo de fin de grado es asignado al alumno hasta el final de este ha participado en dichas reuniones, con una duración entre una hora y dos por semana.

La duración total del proyecto ha sido de unos ocho meses aproximadamente. La asignación del trabajo se produjo a mediados de noviembre, y la entrega final a mitad del mes de julio.

- Conocimientos básicos sobre MATLAB y la herramienta GUIDE

Consulta de guías, documentación y foros online para asimilar conceptos que son necesarios para llevar a cabo el desarrollo de la aplicación.

Ocupa las dos primeras semanas una vez se tiene claro el tema principal del trabajo y en que va a consistir.

- Desarrollo de la aplicación

Para desarrollar este trabajo de fin de grado ha sido necesario trabajar en equipo, durante un tiempo se desarrolló la idea principal de la forma que debía tener la aplicación. Incluyendo la forma de extracción de datos de los archivos de variables estadísticas que ya se han obtenido y la implementación futura de la red.

Esta tarea puede considerarse que se extiende temporalmente durante todo el trabajo, porque hay pequeños cambios según avanza el trabajo y el desarrollo de la aplicación.

- Implantación de la red neuronal en la app

La red debe ser puesta en funcionamiento y validada por otro estudiante. Es necesario modificar algunos parámetros de la propia red o de la parte de la GUI que había sido creada hasta entonces para hacer las llamadas a esta desde la interfaz y asegurar un correcto funcionamiento.

- Programación resultados

Se realiza un estudio sobre los resultados que ofrecen las redes, cuales pueden dar datos más significativos y útiles. Codificación de las tareas correspondientes a la última parte de la aplicación.

- Depuración código y validación del programa

Corregir errores que aparecen en el código y perfeccionar algunos aspectos para asegurar el funcionamiento correcto de la aplicación, en función de distintos parámetros y valores de las carácterísticas de entrada.

- Redacción memoria

Elaboración del documento informe del proyecto. Recopilación de información para completar el contexto y otros apartados de la memoria para mejorar la compresión de algunos aspectos que se llevan a cabo durante el trabajo.

Durante las vacaciones de primavera comienza la búsqueda y recopilación de información, pero la tarea del desarrollo de la memoria se lleva a cabo durante los meses de junio y julio, una vez está finalizando el proyecto.

Durante los meses de febrero a abril se calcula una media de ocho horas de trabajo a la semana. Durante el mes de mayo se trabaja la mitad del mes únicamente mientras que durante todo el mes de junio y mitad de julio se trabaja unas veinticinco horas a la semana, coincidiendo con la parte de redacción y finalización del programa.

11 Diagrama de Gantt

El diagrama de Gantt es una herramienta que permite ordenar en el tiempo unas determinadas actividades para coordinar los recursos temporales previstos y ajustar fechas en un proyecto. Incluye todas las actividades a realizar e hitos a alcanzar con la duración estimada de cada uno, pero no las relaciones entre tareas.

La siguiente tabla muestra las tareas y la duración en días que han consumido. Los intervalos de tiempos entre actividades que se encuentran vacíos corresponden a fechas de exámenes, donde no se ha avanzado en el proyecto (21 de diciembre-25 de enero y 21 de mayo-6 de junio).

Nombre de la tarea	Fecha de inicio	Fecha final	Duración (dias)
Gestión del proyecto	15/11/2018	15/07/2019	242
Aprendizaje básico MATLAB (GUIDE)	01/12/2019	21/12/2019	20
Desarrollo de la aplicación	28/01/2019	07/07/2019	160
Implantación de la red neuronal	29/04/2019	17/05/2019	18
Programación resultados	06/06/2019	28/06/2019	22
Depuración del código y validación del programa	06/06/2019	07/07/2019	31
Redacción de la memoria	18/04/2019	15/07/2019	88

Tabla 13. Tabla de actividades.

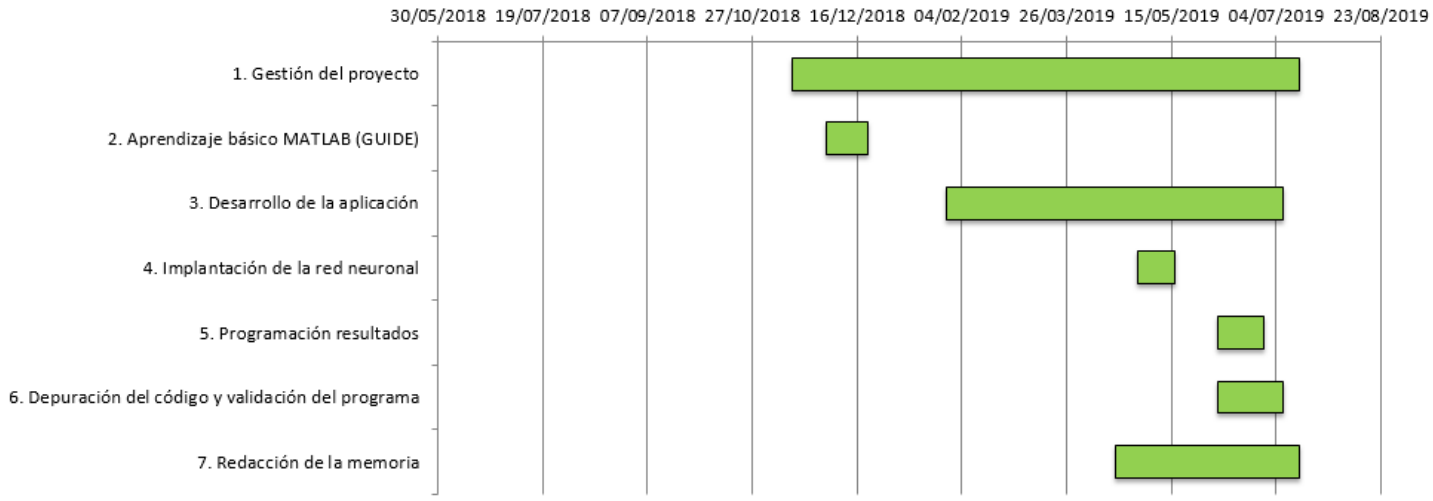


Ilustración 37. Diagrama de Gantt.

12 Presupuesto

12.1 Horas de trabajo

Horas totales (alumno más profesor)

Se estiman 200 horas de Ingeniero Junior necesarias para terminar el proyecto, empleadas en programación y redacción. Las horas de Ingeniero Senior, ejerciendo de Directo del TFG, empleadas en ayudar con la programación, redacción y corregir errores se estiman en unas 20 horas. Se fija las tasas horarias de trabajo en la tabla inferior.

Horas internas	Horas totales	Tasa horaria	Coste total
Ingeniero Junior	200	30 €	6.000 €
Ingeniero Senior	20	35 €	700 €
TOTAL			6.700 €

Tabla 14. Horas internas.

12.2 Amortizaciones

Ordenadores

Para el desarrollo de la aplicación y elaboración de la memoria se ha empleado un ordenador del Laboratorio de Automática y Control valorado en unos 1.000 € junto con el portátil del alumno, valorado en 700 €.

Licencia MATLAB

El software MATLAB precisa de una licencia de pago con un precio de 4.500 €.

Amortizaciones	Coste unitario	Vida útil	Horas utilización	Coste amortización
PC laboratorio	1.000 €	5.000 h	170 h	34 €
PC alumno	700 €	4.000 h	30 h	5.25 €
Licencia MATLAB	4.500 €	4.000 h	170 h	191.25 €
TOTAL				230.5 €

Tabla 15. Amortizaciones.

12.3 Costes indirectos

Engloba los gastos que no se pueden asociar a ningún proyecto: departamentos no productivos, subcontrataciones (seguridad, limpieza) y gastos generales (luz y agua).

Se calculan como un porcentaje de los costes totales.

Costes indirectos	Coste total	Porcentaje costes indirectos	Coste total
Gastos indirectos	6.930,5 €	5 %	346.5 €
TOTAL			346.5 €

Tabla 16. Costes indirectos.

12.4 Presupuesto total

Se muestra debajo una tabla con el presupuesto total, con todas las partidas y el porcentaje sobre el gasto global que conlleva cada una.

Concepto	Coste total	Porcentaje
Horas internas	6.700 €	92.07 %
Amortizaciones	230,5 €	3.17 %
Costes indirectos	346.5 €	4.76 %
TOTAL	7277 €	100%

Tabla 17. Presupuesto final.

El presupuesto total del proyecto asciende a 7277 €.

13 Conclusiones y tareas futuras

Las conclusiones finales que se extraen de este trabajo son: la implementación de la red neuronal validada es correcta y permite acelerar y facilitar el proceso de entrenamiento. La creación de nuevas estructuras de datos funciona y se lleva a cabo de manera ordenada. La interfaz cumple con su objetivo de simplificar una tarea determinada, reduciendo la problemática a una interacción persona-aplicación.

Llevar a cabo este trabajo de fin de grado ha sido una experiencia positiva para el alumno, permitiendo obtener conocimientos sobre un software muy extendido en el ámbito ingenieril como MATLAB.

También ha sido positiva la posibilidad de visitar los laboratorios de captura de movimiento VICON situados en Leioa. La experiencia de trabajar en equipo en estos ensayos resulta enriquecedora.

El objetivo es ayudar a realizar el tratamiento inteligente de los datos para buscar un diagnóstico correcto de los pacientes con EM, por lo que se debe seguir investigando para conseguir el objetivo final.

13.1 Líneas futuras

Un gran contra de MATLAB es que la gestión de la memoria que utiliza no es buena, se podría desarrollar la misma aplicación en un lenguaje como Python. Esto permitiría una mejor gestión de la memoria, lo que se traduce en velocidades más rápidas de entrenamiento y mayor productividad.

Se puede implementar un menú de ayuda, con ventanas emergentes que expliquen que significa cada parámetro y como afecta su variación a los resultados finales.

14 Bibliografía

1. Diego Orlando Barragán Guerrero. Manual de interfaz gráfica de usuario en MATLAB. 2008.
2. Gonzalo Fernández de Córdoba Martos. Creación de Interfaces Gráficas de Usuario (GUI) con MatLab. 2007.
3. Curso básico de GUI MATLAB. https://www.youtube.com/watch?v=S4xR2DjedG8&list=PLjApqg2Zsw31HHJONnV9IQkxTQfUrW_9F
4. Foro de MATLAB. https://www.lawebdelprogramador.com/foros/Matlab/index1.html
5. MATLAB Answers. MathWorks. https://es.mathworks.com/matlabcentral/answers/
6. ESTUDIO SOBRE LAS ENFERMEDADES NEURODEGENERATIVAS EN ESPAÑA Y SU IMPACTO ECONÓMICO Y SOCIAL. Universidad Complutense de Madrid con Neuroalianza. Madrid, Febrero de 2016.
7. Día mundial de la Esclerosis Múltiple. https://www.esclerosismultiple.com/dia-mundial-esclerosis-multiple-2019-cifras-reivindicaciones-actividades-programa/
8. Cifras y datos de la EM. https://www.esclerosismultiple.com/dia-mundial-esclerosis-multiple-2019-cifras-reivindicaciones-actividades-programa/
9. Iñigo Sesar, Aitziber Mancisidor, Asier Brull, Asier Zubizarreta, Itziar Cabanes. Departamento de Ingeniería de Sistemas y Automática. Escuela de Ingeniería de Bilbao (UPV/EHU). Desarrollo de una muleta sensorizada para medir la inclinación y el peso descargado. 2018.
10. Asier Zubizarreta Pico. Memoria SMARTIP. 2017.
11. Damián Jorge Matich. Redes Neuronales: Conceptos Básicos y Aplicaciones. Marzo de 2001.
12. Ivan Edward Sutherland. Sketchpad: A man-machine graphical communication system. 1980.

15 Anexo I. Manual de usuario

En esta sección se prentende ayudar al usuario de la aplicación a interactuar con esta, cumpliendo el objetivo de que pueda ser utilizada por personas sin conocimiento en el software MATLAB.

15.1 Lanzamiento de la aplicación.

Dentro de la carpeta donde se encuentran todos los archivos y programas necesarios para el funcionamiento correcto hay dos archivos con el nombre de "Aplicacion". Haciendo click sobre el archivo cuyo tipo es "MATLAB Figure" se abre el software MATLAB, y posteriormente aparece en pantalla la interfaz preparada para ser utilizada (Ilustración 14).

En caso de querer lanzarla desde la pantalla de MATLAB se debe abrir el archivo "Aplicacion" con tipo "MATLAB Code" y pulsar el botón **Run** que se encuentra en la parte superior de la ventana.

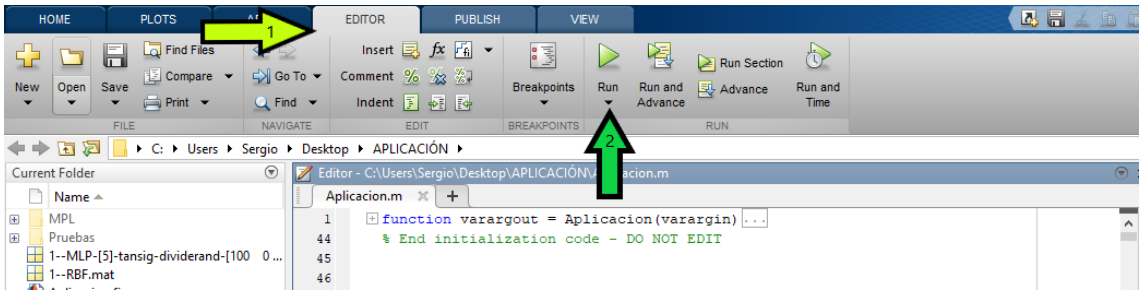


Ilustración 38. Ubicación botón Run.

15.2 Selección datos estadísticos.

Dentro del panel 1 (Ilustración 16) se seleccionan los datos estadísticos que se quieren introducir en la red. Los datos se encuentran seleccionables según sensor (acelerómetro o giróscopo) y eje en el que se ha tomado la medida (X-Y-Z o XY-YZ-XZ para el caso de las correlaciones).

Una vez seleccionados los datos deseados se debe pulsar el **Botón "Crear Tabla Samples"**, la aplicación genera el archivo Samples.mat en la carpeta donde se está ejecutando MATLAB. Esto se puede ir visualizando con la ventana del programa abierta.

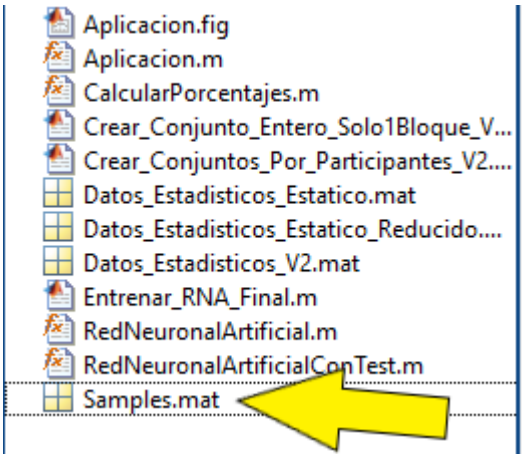


Ilustración 39. Archivos existentes tras pulsar el Botón "Crear Tabla Samples".

15.3 Configuración de red y entrenamiento.

El panel 2 (Ilustración 19) está divido en dos partes. En la primera el usuario debe introducir desde teclado los valores para las variables propias de la red y en la segunda la tipología que se quiere utilizar. En el primer caso los valores se encuentran limitados, si no son adecuados aparece una ventana emergente de aviso, por lo que el entrenamiento no inicia hasta que estos valores sean aptos. La selección de tipología se realiza desde un menú desplegable, todas las opciones seleccionables son válidas para comenzar un entrenamiento.

Si el tipo de red escogida es MLP al pulsar el **Botón Iniciar Entrenamiento** aparece una nueva ventana emergente que permite visualizar en todo momento el estado actual del entrenamiento (Ilustración 21). Para el caso de red RBF no existe tal ventana, los avisos sobre el estado aparecen en la Command Window de MATLAB.

Terminado este proceso deben aparecer en la ubicación de ejecución dos nuevos archivos: “Red.mat” y “Conjunto.mat”.

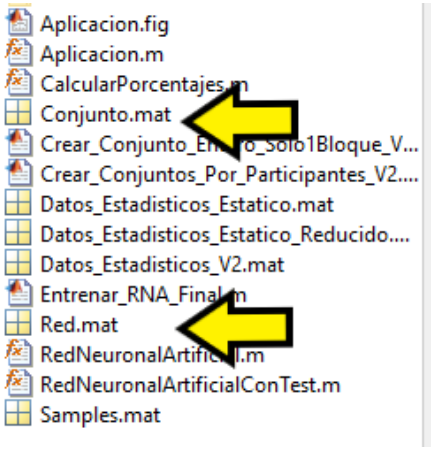


Ilustración 40. Archivos Red.mat y Conjunto.mat

15.4 Visualización de resultados

Una vez finalizado el tratamiento con la red se pueden realizar dos últimas tareas desde el panel 3 (Ilustración 23).

Mediante el **Botón Guardar Resultados** se guarda el archivo Red.mat con un nuevo nombre asignado por el programa, se consigue guardar la información del entrenamiento ya que la variable Red.mat se sobreescribe en cada proceso.

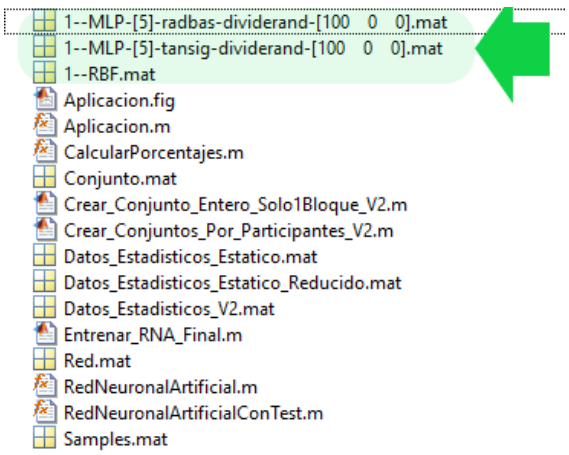


Ilustración 41. Ejemplos de redes guardadas.

La última opción ejecutable se realiza desde el **Botón Graficar Resultados**, muestra en la interfaz datos significativos del entrenamiento igual que aparecen en la Ilustracion 27.